L4Re – L4 Runtime Environment

Generated by Doxygen 1.9.1

1 Overview 1

2 Introduction 3

2.1 Fiasco.OC . . . 3

2.1.1 Communication . . . 3

2.1.2 Kernel Objects . . . 4

2.2 L4Re System Structure . . . 4

2.3 L4 Runtime Environment (L4Re) . . . 6

3 Tutorial 7

3.1 Customizations . . . 8

4 Programming for L4Re 9

4.1 Capabilities and Naming . . . 10

4.2 Spaces and Mappings . . . 11

4.3 Initial Environment and Application Bootstrapping . . . 11

4.3.1 Configuring an application before startup . . . 12

4.3.2 Connecting clients and servers . . . 12

4.4 Memory management - Data Spaces and the Region Map . . . 13

4.4.1 User-level paging . . . 13

4.4.1.1 Data spaces . . . 13

4.4.1.2 Virtual Memory Handling . . . 13

4.4.1.3 Memory Allocation . . . 13

4.5 Program Input and Output . . . 14

4.6 Initial Memory Allocator and Factory . . . 14

4.7 Application and Server Building Blocks . . . 14

4.7.1 Creating Additional Application Threads . . . 14

4.7.2 Providing a Service . . . 15

4.8 Pthread Support . . . 15

4.9 Interface Definition Language . . . 16

4.9.1 Parameter types for RPC . . . 17

4.9.2 RPC Return Types . . . 18

4.9.3 RPC Method Declaration . . . 18

4.10 L4Re Build System . . . 19

4.10.1 Building L4Re . . . 19

4.10.2 Writing BID Make Files . . . 20

4.10.3 prog.mk - Application Role . . . 20

4.10.4 include.mk - Header File Role . . . 23

4.10.5 test.mk - Test Application Role . . . 24

4.11 Kernel Factory . . . 30

4.11.1 Passing parameters for the create stream . . . 31

5 L4Re Servers 33

5.1 Sigma0, the Root-Pager . . . 35

5.1.1 Factory . . . 35
5.2 Moe, the Root-Task . . . 35
5.2.1 Moe objects . . . 35
5.2.1.1 Factory . . . 36
5.2.1.2 Namespace . . . 37
5.2.1.3 Dataspace . . . 37
5.2.1.4 Log Subsystem . . . 38
5.2.1.5 DMA Space . . . 38
5.2.1.6 Scheduler subsystem . . . 38
5.2.1.7 Region Map . . . 38
5.2.2 Command Line Options . . . 38
5.3 Ned, the Init Process . . . 39
5.3.1 Lua Bindings for L4Re . . . 39
5.3.1.1 Capabilities in Lua . . . 39
5.3.1.2 Access to L4Re::Env Capabilities . . . 39
5.3.1.3 Constants . . . 40
5.3.1.4 Application Startup Details . . . 40
5.3.1.5 Access to the kernel debugger . . . 41
5.3.1.6 Using the interactive ned prompt . . . 41
5.3.2 Command Line Options . . . 42
5.4 Io, the Io Server . . . 42
5.5 l4vio_net_p2p, a virtual network point-to-point link . . . 47
5.6 Uvmm, the virtual machine monitor . . . 49
5.7 Mag, the GUI Multiplexer . . . 56
5.8 Cons, the Console Multiplexer . . . 58

6 Bootstrap, the L4 kernel bootstrapper **61**

7 Deprecated List **65**

8 Module Index **67**
8.1 Modules . . . 67

9 Namespace Index **71**
9.1 Namespace List . . . 71

10 Hierarchical Index **73**
10.1 Class Hierarchy . . . 73

11 Data Structure Index **83**
11.1 Data Structures . . . 83

12 File Index **97**
12.1 File List . . . 97

13 Module Documentation 109
13.1 ARM Virtual Registers (UTCB) . . . 109
13.1.1 Detailed Description . . . 110
13.2 Atomic Instructions . . . 110
13.2.1 Detailed Description . . . 112
13.2.2 Function Documentation . . . 112
13.2.2.1 l4util_add16() . . . 112
13.2.2.2 l4util_add16_res() . . . 112
13.2.2.3 l4util_add32() . . . 112
13.2.2.4 l4util_add32_res() . . . 113
13.2.2.5 l4util_add8() . . . 113
13.2.2.6 l4util_add8_res() . . . 113
13.2.2.7 l4util_and16() . . . 114
13.2.2.8 l4util_and16_res() . . . 114
13.2.2.9 l4util_and32() . . . 115
13.2.2.10 l4util_and32_res() . . . 115
13.2.2.11 l4util_and8() . . . 115
13.2.2.12 l4util_and8_res() . . . 116
13.2.2.13 l4util_atomic_add() . . . 116
13.2.2.14 l4util_atomic_inc() . . . 116
13.2.2.15 l4util_cmpxchg() . . . 117
13.2.2.16 l4util_cmpxchg16() . . . 117
13.2.2.17 l4util_cmpxchg32() . . . 118
13.2.2.18 l4util_cmpxchg64() . . . 118
13.2.2.19 l4util_cmpxchg8() . . . 119
13.2.2.20 l4util_dec16() . . . 119
13.2.2.21 l4util_dec16_res() . . . 120
13.2.2.22 l4util_dec32() . . . 120
13.2.2.23 l4util_dec32_res() . . . 120
13.2.2.24 l4util_dec8() . . . 121
13.2.2.25 l4util_dec8_res() . . . 121
13.2.2.26 l4util_inc16() . . . 121
13.2.2.27 l4util_inc16_res() . . . 121
13.2.2.28 l4util_inc32() . . . 122
13.2.2.29 l4util_inc32_res() . . . 122
13.2.2.30 l4util_inc8() . . . 122
13.2.2.31 l4util_inc8_res() . . . 123
13.2.2.32 l4util_or16() . . . 123
13.2.2.33 l4util_or16_res() . . . 123
13.2.2.34 l4util_or32() . . . 124
13.2.2.35 l4util_or32_res() . . . 124
13.2.2.36 l4util_or8() . . . 125

13.2.2.37 l4util_or8_res() . . . 125
13.2.2.38 l4util_sub16() . . . 125
13.2.2.39 l4util_sub16_res() . . . 126
13.2.2.40 l4util_sub32() . . . 126
13.2.2.41 l4util_sub32_res() . . . 126
13.2.2.42 l4util_sub8() . . . 127
13.2.2.43 l4util_sub8_res() . . . 127
13.2.2.44 l4util_xchg() . . . 127
13.2.2.45 l4util_xchg16() . . . 128
13.2.2.46 l4util_xchg32() . . . 128
13.2.2.47 l4util_xchg8() . . . 129
13.3 Auxiliary data . . . 129
13.3.1 Detailed Description . . . 130
13.4 Base API . . . 130
13.4.1 Detailed Description . . . 132
13.5 Basic Macros . . . 132
13.5.1 Detailed Description . . . 134
13.5.2 Macro Definition Documentation . . . 134
13.5.2.1 L4_ALWAYS_INLINE . . . 134
13.5.2.2 L4_HIDDEN . . . 134
13.5.2.3 L4_NOTHROW . . . 134
13.5.3 Function Documentation . . . 135
13.5.3.1 l4_align_stack_for_direct_fncall() . . . 135
13.6 Bit Manipulation . . . 135
13.6.1 Detailed Description . . . 136
13.6.2 Function Documentation . . . 136
13.6.2.1 l4util_bsf() . . . 136
13.6.2.2 l4util_bsr() . . . 137
13.6.2.3 l4util_btc() . . . 138
13.6.2.4 l4util_btr() . . . 138
13.6.2.5 l4util_bts() . . . 139
13.6.2.6 l4util_clear_bit() . . . 140
13.6.2.7 l4util_complement_bit() . . . 140
13.6.2.8 l4util_find_first_set_bit() . . . 140
13.6.2.9 l4util_find_first_zero_bit() . . . 141
13.6.2.10 l4util_next_power2() . . . 141
13.6.2.11 l4util_set_bit() . . . 142
13.6.2.12 l4util_test_bit() . . . 142
13.7 Bitmap graphics and fonts . . . 143
13.7.1 Detailed Description . . . 143
13.8 Buffer Registers (BRs) . . . 143
13.8.1 Detailed Description . . . 144

13.8.2 Enumeration Type Documentation . . . 144
13.8.2.1 l4_buffer_desc_consts_t . . . 144
13.9 C++ Exceptions . . . 145
13.9.1 Detailed Description . . . 146
13.10 C++ IPC Interface Definition. . . . 146
13.10.1 Detailed Description . . . 146
13.11 CPU related functions . . . 146
13.11.1 Detailed Description . . . 147
13.11.2 Function Documentation . . . 147
13.11.2.1 l4util_cpu_capabilities() . . . 147
13.11.2.2 l4util_cpu_capabilities_nocheck() . . . 147
13.11.2.3 l4util_cpu_has_cpuid() . . . 148
13.12 Cache Consistency . . . 149
13.12.1 Detailed Description . . . 150
13.12.2 Function Documentation . . . 150
13.12.2.1 l4_cache_clean_data() . . . 150
13.12.2.2 l4_cache_coherent() . . . 150
13.12.2.3 l4_cache_dma_coherent() . . . 151
13.12.2.4 l4_cache_flush_data() . . . 151
13.12.2.5 l4_cache_inv_data() . . . 152
13.13 Capabilities . . . 152
13.13.1 Detailed Description . . . 153
13.13.2 Enumeration Type Documentation . . . 153
13.13.2.1 l4_cap_consts_t . . . 153
13.13.2.2 l4_default_caps_t . . . 154
13.13.3 Function Documentation . . . 155
13.13.3.1 l4_capability_equal() . . . 155
13.13.3.2 l4_is_invalid_cap() . . . 155
13.13.3.3 l4_is_valid_cap() . . . 156
13.14 Capability allocator . . . 156
13.14.1 Detailed Description . . . 157
13.14.2 Function Documentation . . . 157
13.14.2.1 l4re_util_cap_last() . . . 157
13.15 Chunks . . . 157
13.15.1 Detailed Description . . . 158
13.15.2 Function Documentation . . . 158
13.15.2.1 l4shmc_add_chunk() . . . 158
13.15.2.2 l4shmc_chunk_capacity() . . . 159
13.15.2.3 l4shmc_chunk_ptr() . . . 159
13.15.2.4 l4shmc_chunk_signal() . . . 160
13.15.2.5 l4shmc_get_chunk() . . . 160
13.15.2.6 l4shmc_get_chunk_to() . . . 161

13.15.2.7 l4shmc_iterate_chunk() . . . 161
13.16 Comfortable Command Line Parsing . . . 162
13.16.1 Detailed Description . . . 162
13.16.2 Function Documentation . . . 162
13.16.2.1 parse_cmdline() . . . 163
13.17 Console API . . . 164
13.17.1 Detailed Description . . . 164
13.18 Consumer . . . 165
13.18.1 Detailed Description . . . 165
13.18.2 Function Documentation . . . 165
13.18.2.1 l4shmc_chunk_consumed() . . . 165
13.18.2.2 l4shmc_chunk_size() . . . 166
13.18.2.3 l4shmc_enable_chunk() . . . 166
13.18.2.4 l4shmc_is_chunk_ready() . . . 167
13.18.2.5 l4shmc_wait_chunk() . . . 167
13.18.2.6 l4shmc_wait_chunk_to() . . . 167
13.18.2.7 l4shmc_wait_chunk_try() . . . 168
13.19 Consumer . . . 168
13.19.1 Detailed Description . . . 169
13.19.2 Function Documentation . . . 169
13.19.2.1 l4shmc_enable_signal() . . . 169
13.19.2.2 l4shmc_wait_any() . . . 169
13.19.2.3 l4shmc_wait_any_to() . . . 170
13.19.2.4 l4shmc_wait_any_try() . . . 170
13.19.2.5 l4shmc_wait_signal() . . . 171
13.19.2.6 l4shmc_wait_signal_to() . . . 171
13.19.2.7 l4shmc_wait_signal_try() . . . 171
13.20 DMA Space Interface . . . 172
13.20.1 Detailed Description . . . 173
13.20.2 Typedef Documentation . . . 173
13.20.2.1 l4re_dma_space_t . . . 173
13.20.3 Function Documentation . . . 173
13.20.3.1 l4re_dma_space_associate() . . . 173
13.20.3.2 l4re_dma_space_disassociate() . . . 174
13.20.3.3 l4re_dma_space_map() . . . 174
13.20.3.4 l4re_dma_space_unmap() . . . 175
13.21 DMA space . . . 176
13.22 Dataspace interface . . . 176
13.22.1 Detailed Description . . . 177
13.22.2 Enumeration Type Documentation . . . 177
13.22.2.1 l4re_ds_map_flags . . . 177
13.22.3 Function Documentation . . . 178

13.22.3.1 l4re_ds_allocate() . . . 178
13.22.3.2 l4re_ds_clear() . . . 178
13.22.3.3 l4re_ds_copy_in() . . . 178
13.22.3.4 l4re_ds_flags() . . . 179
13.22.3.5 l4re_ds_info() . . . 179
13.22.3.6 l4re_ds_size() . . . 179
13.23 Debug interface . . . 179
13.23.1 Detailed Description . . . 180
13.23.2 Function Documentation . . . 180
13.23.2.1 l4re_debug_obj_debug() . . . 180
13.24 Debugging API . . . 180
13.24.1 Detailed Description . . . 181
13.25 EDID parsing functionality . . . 181
13.25.1 Detailed Description . . . 181
13.25.2 Enumeration Type Documentation . . . 181
13.25.2.1 Libedid_consts . . . 181
13.25.3 Function Documentation . . . 182
13.25.3.1 libedid_check_header() . . . 182
13.25.3.2 libedid_checksum() . . . 182
13.25.3.3 libedid_dump() . . . 182
13.25.3.4 libedid_dump_standard_timings() . . . 183
13.25.3.5 libedid_num_ext_blocks() . . . 183
13.25.3.6 libedid_pnp_id() . . . 183
13.25.3.7 libedid_prefered_resolution() . . . 184
13.25.3.8 libedid_revision() . . . 184
13.25.3.9 libedid_version() . . . 185
13.26 ELF binary format . . . 185
13.26.1 Detailed Description . . . 191
13.26.2 Macro Definition Documentation . . . 191
13.26.2.1 ELF32_R_TYPE . . . 191
13.26.2.2 ELF32_ST_BIND . . . 191
13.26.2.3 ELF32_ST_TYPE . . . 191
13.26.2.4 ELF64_R_TYPE . . . 192
13.26.2.5 ELF64_ST_BIND . . . 192
13.26.2.6 ELF64_ST_TYPE . . . 192
13.26.3 Enumeration Type Documentation . . . 192
13.26.3.1 anonymous enum . . . 192
13.26.3.2 Elf_ARM_SBs . . . 193
13.26.3.3 Elf_ATs . . . 193
13.26.3.4 Elf_CIASSs . . . 194
13.26.3.5 Elf_DATAs . . . 194
13.26.3.6 Elf_DF_1s . . . 194

13.26.3.7 Elf_DF_P1s . . . 195
13.26.3.8 Elf_DFs . . . 195
13.26.3.9 Elf_DTs . . . 196
13.26.3.10 Elf_EF_ARM_s . . . 197
13.26.3.11 Elf_EIs . . . 197
13.26.3.12 Elf_EMs . . . 198
13.26.3.13 Elf_ETs . . . 200
13.26.3.14 Elf_EVs . . . 200
13.26.3.15 Elf_MAGs . . . 201
13.26.3.16 Elf_NTs_core . . . 201
13.26.3.17 Elf_NTs_obj . . . 201
13.26.3.18 Elf_OSABIs . . . 202
13.26.3.19 ELF_PFs . . . 202
13.26.3.20 Elf_PTs . . . 203
13.26.3.21 Elf_R_386_s . . . 203
13.26.3.22 Elf_R_AARCH64_s . . . 205
13.26.3.23 Elf_R_ARM_s . . . 206
13.26.3.24 Elf_R_X86_64_s . . . 206
13.26.3.25 Elf_SHF_s_ARM . . . 207
13.26.3.26 Elf_SHFs . . . 207
13.26.3.27 Elf_SHNs . . . 208
13.26.3.28 Elf_SHTs . . . 208
13.26.3.29 Elf_STBs . . . 209
13.26.3.30 Elf_STTs . . . 210
13.27 Error Handling . . . 210
13.27.1 Detailed Description . . . 211
13.27.2 Enumeration Type Documentation . . . 211
13.27.2.1 l4_ipc_tcr_error_t . . . 211
13.27.3 Function Documentation . . . 212
13.27.3.1 l4_error() . . . 212
13.27.3.2 l4_ipc_error() . . . 213
13.27.3.3 l4_ipc_error_code() . . . 214
13.27.3.4 l4_ipc_is_rcv_error() . . . 215
13.27.3.5 l4_ipc_is_snd_error() . . . 215
13.28 Error codes . . . 216
13.28.1 Detailed Description . . . 216
13.28.2 Enumeration Type Documentation . . . 216
13.28.2.1 l4_error_code_t . . . 216
13.29 Event API . . . 217
13.29.1 Detailed Description . . . 218
13.30 Event interface . . . 218
13.30.1 Detailed Description . . . 219

13.30.2 Function Documentation . . . 219
13.30.2.1 l4re_event_get_axis_info() . . . 219
13.30.2.2 l4re_event_get_buffer() . . . 219
13.30.2.3 l4re_event_get_num_streams() . . . 220
13.30.2.4 l4re_event_get_stream_info() . . . 220
13.30.2.5 l4re_event_get_stream_info_for_id() . . . 221
13.31 Exception registers . . . 221
13.31.1 Detailed Description . . . 222
13.31.2 Function Documentation . . . 222
13.31.2.1 l4_utcb_exc() . . . 222
13.31.2.2 l4_utcb_exc_is_ex_regs_exception() . . . 223
13.31.2.3 l4_utcb_exc_is_pf() . . . 223
13.31.2.4 l4_utcb_exc_pc() . . . 223
13.31.2.5 l4_utcb_exc_pc_set() . . . 224
13.32 Extended vCPU support . . . 224
13.32.1 Detailed Description . . . 225
13.32.2 Function Documentation . . . 225
13.32.2.1 l4vcpu_ext_alloc() . . . 225
13.33 Factory . . . 225
13.33.1 Detailed Description . . . 226
13.33.2 Function Documentation . . . 227
13.33.2.1 l4_factory_create() . . . 227
13.33.2.2 l4_factory_create_factory() . . . 227
13.33.2.3 l4_factory_create_gate() . . . 228
13.33.2.4 l4_factory_create_irq() . . . 229
13.33.2.5 l4_factory_create_task() . . . 229
13.33.2.6 l4_factory_create_thread() . . . 230
13.33.2.7 l4_factory_create_vm() . . . 230
13.34 Fiasco extensions . . . 231
13.34.1 Detailed Description . . . 232
13.34.2 Function Documentation . . . 232
13.34.2.1 fiasco_gdt_get_entry_offset() . . . 232
13.34.2.2 fiasco_gdt_set() . . . 233
13.34.2.3 fiasco_ldt_set() . . . 233
13.34.2.4 fiasco_tbuf_log() . . . 234
13.34.2.5 fiasco_tbuf_log_3val() . . . 235
13.34.2.6 fiasco_tbuf_log_binary() . . . 235
13.35 Fiasco real time scheduling extensions . . . 236
13.36 Fiasco-UX Virtual devices . . . 236
13.36.1 Detailed Description . . . 237
13.36.2 Enumeration Type Documentation . . . 237
13.36.2.1 l4_vhw_entry_type . . . 237

13.37 Flex pages . . . 237
13.37.1 Detailed Description . . . 240
13.37.2 Enumeration Type Documentation . . . 240
13.37.2.1 anonymous enum . . . 240
13.37.2.2 anonymous enum . . . 240
13.37.2.3 L4_cap_fpage_rights . . . 241
13.37.2.4 l4_fpage_cacheability_opt_t . . . 242
13.37.2.5 l4_fpage_consts . . . 242
13.37.2.6 L4_fpage_control . . . 243
13.37.2.7 L4_fpage_rights . . . 243
13.37.2.8 L4_fpage_type . . . 244
13.37.2.9 L4_obj_fpage_ctl . . . 244
13.37.3 Function Documentation . . . 244
13.37.3.1 l4_fpage() . . . 245
13.37.3.2 l4_fpage_all() . . . 245
13.37.3.3 l4_fpage_contains() . . . 245
13.37.3.4 l4_fpage_invalid() . . . 246
13.37.3.5 l4_fpage_ioport() . . . 246
13.37.3.6 l4_fpage_max_order() . . . 247
13.37.3.7 l4_fpage_memaddr() . . . 248
13.37.3.8 l4_fpage_obj() . . . 249
13.37.3.9 l4_fpage_page() . . . 249
13.37.3.10 l4_fpage_rights() . . . 250
13.37.3.11 l4_fpage_set_rights() . . . 250
13.37.3.12 l4_fpage_size() . . . 251
13.37.3.13 l4_fpage_type() . . . 251
13.37.3.14 l4_iofpage() . . . 252
13.37.3.15 l4_is_fpage_writable() . . . 253
13.37.3.16 l4_obj_fpage() . . . 253
13.38 Functions for rendering bitmap data in frame buffers . . . 254
13.39 Functions for rendering bitmap fonts to frame buffers . . . 254
13.40 Functions to manipulate the local IDT . . . 255
13.40.1 Detailed Description . . . 255
13.41 IA32 Port I/O API . . . 255
13.41.1 Detailed Description . . . 256
13.41.2 Function Documentation . . . 256
13.41.2.1 l4util_in16() . . . 256
13.41.2.2 l4util_in32() . . . 256
13.41.2.3 l4util_in8() . . . 257
13.41.2.4 l4util_ins16() . . . 257
13.41.2.5 l4util_ins32() . . . 258
13.41.2.6 l4util_ins8() . . . 258

13.41.2.7 l4util_out16() . . . 259
13.41.2.8 l4util_out32() . . . 259
13.41.2.9 l4util_out8() . . . 259
13.41.2.10 l4util_outs16() . . . 260
13.41.2.11 l4util_outs32() . . . 260
13.41.2.12 l4util_outs8() . . . 261
13.42 IO interface . . . 261
13.42.1 Detailed Description . . . 262
13.42.2 Typedef Documentation . . . 262
13.42.2.1 l4io_resource_t . . . 262
13.42.3 Enumeration Type Documentation . . . 262
13.42.3.1 l4io_device_types_t . . . 262
13.42.3.2 l4io_iomem_flags_t . . . 263
13.42.3.3 l4io_resource_types_t . . . 263
13.42.4 Function Documentation . . . 264
13.42.4.1 l4io_has_resource() . . . 264
13.42.4.2 l4io_lookup_device() . . . 264
13.42.4.3 l4io_lookup_resource() . . . 265
13.42.4.4 l4io_release_iomem() . . . 265
13.42.4.5 l4io_release_ioport() . . . 265
13.42.4.6 l4io_request_iomem() . . . 266
13.42.4.7 l4io_request_iomem_region() . . . 267
13.42.4.8 l4io_request_ioport() . . . 267
13.42.4.9 l4io_request_resource_iomem() . . . 268
13.43 IPC-Gate API . . . 268
13.43.1 Detailed Description . . . 269
13.43.2 Function Documentation . . . 269
13.43.2.1 l4_ipc_gate_get_infos() . . . 269
13.43.2.2 l4_rcv_ep_bind_thread() . . . 270
13.44 IRQ handling library . . . 271
13.44.1 Detailed Description . . . 271
13.45 IRQs . . . 271
13.45.1 Detailed Description . . . 272
13.45.2 Enumeration Type Documentation . . . 273
13.45.2.1 L4_irq_mode . . . 273
13.45.3 Function Documentation . . . 274
13.45.3.1 l4_irq_detach() . . . 274
13.45.3.2 l4_irq_detach_u() . . . 275
13.45.3.3 l4_irq_mux_chain() . . . 276
13.45.3.4 l4_irq_mux_chain_u() . . . 276
13.45.3.5 l4_irq_receive() . . . 277
13.45.3.6 l4_irq_receive_u() . . . 278

13.45.3.7 l4_irq_trigger() . . . 279
13.45.3.8 l4_irq_trigger_u() . . . 280
13.45.3.9 l4_irq_unmask() . . . 281
13.45.3.10 l4_irq_unmask_u() . . . 282
13.45.3.11 l4_irq_wait() . . . 283
13.45.3.12 l4_irq_wait_u() . . . 284
13.46 Initial Environment . . . 284
13.46.1 Detailed Description . . . 285
13.46.2 Function Documentation . . . 285
13.46.2.1 l4re_env() . . . 286
13.46.2.2 l4re_env_get_cap() . . . 286
13.46.2.3 l4re_env_get_cap_e() . . . 287
13.46.2.4 l4re_env_get_cap_l() . . . 288
13.46.2.5 l4re_kip() . . . 289
13.47 Integer Types . . . 289
13.47.1 Detailed Description . . . 291
13.48 Interface for asynchronous ISR handlers with a given IRQ capability. . . . 291
13.48.1 Detailed Description . . . 291
13.48.2 Function Documentation . . . 291
13.48.2.1 l4irq_request_cap() . . . 291
13.49 Interface for asynchronous ISR handlers. . . . 292
13.49.1 Detailed Description . . . 292
13.49.2 Function Documentation . . . 293
13.49.2.1 l4irq_release() . . . 293
13.49.2.2 l4irq_request() . . . 293
13.50 Interface using direct functionality. . . . 294
13.50.1 Detailed Description . . . 294
13.50.2 Function Documentation . . . 294
13.50.2.1 l4irq_attach_cap() . . . 294
13.50.2.2 l4irq_attach_cap_ft() . . . 295
13.50.2.3 l4irq_attach_thread_cap() . . . 295
13.50.2.4 l4irq_attach_thread_cap_ft() . . . 296
13.51 Interface using direct functionality. . . . 296
13.51.1 Detailed Description . . . 297
13.51.2 Function Documentation . . . 297
13.51.2.1 l4irq_attach() . . . 297
13.51.2.2 l4irq_attach_ft() . . . 298
13.51.2.3 l4irq_attach_thread() . . . 298
13.51.2.4 l4irq_attach_thread_ft() . . . 298
13.51.2.5 l4irq_detach() . . . 299
13.51.2.6 l4irq_unmask() . . . 299
13.51.2.7 l4irq_unmask_and_wait_any() . . . 300

13.51.2.8 l4irq_wait() . . . 300
13.51.2.9 l4irq_wait_any() . . . 300
13.52 Internal Helpers . . . 301
13.52.1 Detailed Description . . . 301
13.53 Internal constants . . . 301
13.53.1 Detailed Description . . . 302
13.54 Internal functions . . . 303
13.54.1 Detailed Description . . . 303
13.55 Interrupt controller . . . 303
13.55.1 Detailed Description . . . 305
13.55.2 Typedef Documentation . . . 305
13.55.2.1 l4_icu_info_t . . . 305
13.55.3 Enumeration Type Documentation . . . 305
13.55.3.1 L4_icu_flags . . . 305
13.55.4 Function Documentation . . . 306
13.55.4.1 l4_icu_bind() . . . 306
13.55.4.2 l4_icu_bind_u() . . . 307
13.55.4.3 l4_icu_info() . . . 308
13.55.4.4 l4_icu_info_u() . . . 309
13.55.4.5 l4_icu_mask() . . . 309
13.55.4.6 l4_icu_mask_u() . . . 310
13.55.4.7 l4_icu_msi_info() . . . 311
13.55.4.8 l4_icu_msi_info_u() . . . 312
13.55.4.9 l4_icu_set_mode() . . . 312
13.55.4.10 l4_icu_set_mode_u() . . . 313
13.55.4.11 l4_icu_unbind() . . . 314
13.55.4.12 l4_icu_unbind_u() . . . 315
13.55.4.13 l4_icu_unmask() . . . 316
13.55.4.14 l4_icu_unmask_u() . . . 316
13.56 Kernel Debugger . . . 317
13.56.1 Detailed Description . . . 318
13.56.2 Function Documentation . . . 318
13.56.2.1 l4_debugger_global_id() . . . 318
13.56.2.2 l4_debugger_kobj_to_id() . . . 318
13.56.2.3 l4_debugger_set_object_name() . . . 319
13.57 Kernel Interface Page . . . 319
13.57.1 Detailed Description . . . 321
13.57.2 Enumeration Type Documentation . . . 321
13.57.2.1 anonymous enum . . . 321
13.57.3 Function Documentation . . . 321
13.57.3.1 l4_kernel_info_version_offset() . . . 322
13.57.3.2 l4_kip_clock() . . . 322

13.57.3.3 l4_kip_clock_lw() . . . 323
13.57.3.4 l4_kip_clock_ns() . . . 324
13.57.3.5 l4_kip_version() . . . 324
13.57.3.6 l4_kip_version_string() . . . 325
13.58 Kernel Interface Page API . . . 326
13.58.1 Detailed Description . . . 326
13.58.2 Macro Definition Documentation . . . 326
13.58.2.1 l4util_kip_for_each_feature . . . 326
13.58.3 Function Documentation . . . 327
13.58.3.1 l4util_kip_kernel_abi_version() . . . 327
13.58.3.2 l4util_kip_kernel_has_feature() . . . 327
13.58.3.3 l4util_kip_kernel_is_ux() . . . 327
13.59 Kernel Objects . . . 328
13.59.1 Detailed Description . . . 329
13.60 Kernel-provided semaphore . . . 329
13.60.1 Detailed Description . . . 330
13.60.2 Function Documentation . . . 330
13.60.2.1 l4_semaphore_down() . . . 330
13.60.2.2 l4_semaphore_up() . . . 331
13.61 Kumem allocator utility . . . 331
13.62 Kumem utiltities . . . 332
13.62.1 Detailed Description . . . 332
13.62.2 Function Documentation . . . 332
13.62.2.1 kumem_alloc() . . . 332
13.63 L4 IPC Opcodes . . . 333
13.63.1 Detailed Description . . . 333
13.63.2 Enumeration Type Documentation . . . 333
13.63.2.1 L4_icu_opcode . . . 334
13.63.2.2 L4_ipc_gate_ops . . . 334
13.63.2.3 L4_platform_ctl_ops . . . 335
13.63.2.4 L4_task_ops . . . 335
13.63.2.5 L4_thread_ops . . . 336
13.63.2.6 L4_vcon_ops . . . 336
13.64 L4 VIRTIO Block Device . . . 337
13.64.1 Detailed Description . . . 337
13.64.2 Enumeration Type Documentation . . . 337
13.64.2.1 L4virtio_block_operations . . . 337
13.64.2.2 L4virtio_block_status . . . 338
13.65 L4 VIRTIO Input Device . . . 338
13.65.1 Detailed Description . . . 339
13.66 L4 VIRTIO Interface . . . 339
13.66.1 Detailed Description . . . 340

13.67 L4 VIRTIO Transport Layer . . . 340
13.67.1 Detailed Description . . . 341
13.67.2 Typedef Documentation . . . 342
13.67.2.1 l4virtio_config_queue_t . . . 342
13.67.3 Enumeration Type Documentation . . . 342
13.67.3.1 L4_virtio_cmd . . . 342
13.67.3.2 L4_virtio_irq_status . . . 342
13.67.3.3 L4_virtio_opcodes . . . 343
13.67.3.4 L4virtio_device_ids . . . 343
13.67.3.5 L4virtio_device_status . . . 344
13.67.3.6 L4virtio_feature_bits . . . 344
13.67.4 Function Documentation . . . 345
13.67.4.1 l4virtio_config_queue() . . . 345
13.67.4.2 l4virtio_config_queues() . . . 345
13.67.4.3 l4virtio_device_config() . . . 346
13.67.4.4 l4virtio_device_config_ds() . . . 346
13.67.4.5 l4virtio_device_notification_irq() . . . 346
13.67.4.6 l4virtio_register_ds() . . . 347
13.67.4.7 l4virtio_register_iface() . . . 347
13.67.4.8 l4virtio_set_status() . . . 348
13.68 L4 Vbus functions . . . 348
13.68.1 Detailed Description . . . 350
13.68.2 Enumeration Type Documentation . . . 350
13.68.2.1 L4vbus_dma_domain_assign_flags . . . 350
13.68.3 Function Documentation . . . 350
13.68.3.1 l4vbus_assign_dma_domain() . . . 351
13.68.3.2 l4vbus_get_device() . . . 351
13.68.3.3 l4vbus_get_device_by_hid() . . . 352
13.68.3.4 l4vbus_get_hid() . . . 353
13.68.3.5 l4vbus_get_next_device() . . . 353
13.68.3.6 l4vbus_get_resource() . . . 355
13.68.3.7 l4vbus_is_compatible() . . . 356
13.68.3.8 l4vbus_release_ioport() . . . 357
13.68.3.9 l4vbus_release_resource() . . . 357
13.68.3.10 l4vbus_request_ioport() . . . 358
13.68.3.11 l4vbus_request_resource() . . . 359
13.68.3.12 l4vbus_vicu_get_cap() . . . 359
13.69 L4 kernel object type information . . . 360
13.69.1 Detailed Description . . . 361
13.69.2 Function Documentation . . . 361
13.69.2.1 kobject_typeid() . . . 361
13.70 L4Re C Interface . . . 361

13.70.1 Detailed Description . . . 363
13.70.2 Function Documentation . . . 363
13.70.2.1 l4re_inhibitor_acquire() . . . 363
13.71 L4Re C++ Interface . . . 364
13.71.1 Detailed Description . . . 366
13.72 L4Re Capability API . . . 366
13.72.1 Detailed Description . . . 367
13.72.2 Function Documentation . . . 367
13.72.2.1 make_ref_cap() . . . 367
13.72.2.2 make_ref_del_cap() . . . 368
13.72.3 Variable Documentation . . . 368
13.72.3.1 cap_alloc . . . 368
13.73 L4Re ELF Auxiliary Information . . . 369
13.73.1 Detailed Description . . . 370
13.73.2 Macro Definition Documentation . . . 370
13.73.2.1 L4RE_ELF_AUX_ELEM . . . 370
13.73.2.2 L4RE_ELF_AUX_ELEM_T . . . 370
13.73.3 Enumeration Type Documentation . . . 371
13.73.3.1 anonymous enum . . . 371
13.74 L4Re Protocol identifiers . . . 371
13.74.1 Detailed Description . . . 372
13.75 L4Re Util C Interface . . . 372
13.76 L4Re Util C++ Interface . . . 372
13.76.1 Detailed Description . . . 373
13.77 L4vbus GPIO functions . . . 373
13.77.1 Detailed Description . . . 374
13.77.2 Enumeration Type Documentation . . . 374
13.77.2.1 L4vbus_gpio_generic_func . . . 374
13.77.2.2 L4vbus_gpio_pull_modes . . . 375
13.77.3 Function Documentation . . . 375
13.77.3.1 l4vbus_gpio_config_get() . . . 375
13.77.3.2 l4vbus_gpio_config_pad() . . . 376
13.77.3.3 l4vbus_gpio_config_pull() . . . 377
13.77.3.4 l4vbus_gpio_get() . . . 377
13.77.3.5 l4vbus_gpio_multi_config_pad() . . . 378
13.77.3.6 l4vbus_gpio_multi_get() . . . 379
13.77.3.7 l4vbus_gpio_multi_set() . . . 380
13.77.3.8 l4vbus_gpio_multi_setup() . . . 380
13.77.3.9 l4vbus_gpio_set() . . . 381
13.77.3.10 l4vbus_gpio_setup() . . . 382
13.77.3.11 l4vbus_gpio_to_irq() . . . 383
13.78 L4vbus PCI functions . . . 384

13.78.1 Detailed Description . . . 384
13.78.2 Function Documentation . . . 384
13.78.2.1 l4vbus_pci_cfg_read() . . . 384
13.78.2.2 l4vbus_pci_cfg_write() . . . 385
13.78.2.3 l4vbus_pci_irq_enable() . . . 386
13.78.2.4 l4vbus_pcidev_cfg_read() . . . 387
13.78.2.5 l4vbus_pcidev_cfg_write() . . . 388
13.78.2.6 l4vbus_pcidev_irq_enable() . . . 388
13.79 Log interface . . . 389
13.79.1 Detailed Description . . . 390
13.79.2 Function Documentation . . . 390
13.79.2.1 l4re_log_print() . . . 390
13.79.2.2 l4re_log_print_srv() . . . 390
13.79.2.3 l4re_log_printn() . . . 391
13.79.2.4 l4re_log_printn_srv() . . . 392
13.80 Logging interface . . . 393
13.80.1 Detailed Description . . . 393
13.81 Low-Level Thread Functions . . . 393
13.82 Memory allocator . . . 394
13.82.1 Detailed Description . . . 394
13.82.2 Enumeration Type Documentation . . . 394
13.82.2.1 l4re_ma_flags . . . 394
13.82.3 Function Documentation . . . 395
13.82.3.1 l4re_ma_alloc() . . . 395
13.82.3.2 l4re_ma_alloc_align() . . . 396
13.82.3.3 l4re_ma_alloc_align_srv() . . . 397
13.83 Memory descriptors (C version) . . . 398
13.83.1 Detailed Description . . . 399
13.83.2 Typedef Documentation . . . 399
13.83.2.1 l4_kernel_info_mem_desc_t . . . 399
13.83.3 Enumeration Type Documentation . . . 399
13.83.3.1 l4_mem_info_sub_type_t . . . 399
13.83.3.2 l4_mem_type_t . . . 400
13.83.4 Function Documentation . . . 400
13.83.4.1 l4_kernel_info_get_mem_desc_end() . . . 400
13.83.4.2 l4_kernel_info_get_mem_desc_is_virtual() . . . 401
13.83.4.3 l4_kernel_info_get_mem_desc_start() . . . 401
13.83.4.4 l4_kernel_info_get_mem_desc_subtype() . . . 401
13.83.4.5 l4_kernel_info_get_mem_desc_type() . . . 402
13.83.4.6 l4_kernel_info_get_num_mem_descs() . . . 402
13.83.4.7 l4_kernel_info_set_mem_desc() . . . 402
13.84 Memory operations. . . . 403

13.84.1 Detailed Description . . . 403
13.84.2 Enumeration Type Documentation . . . 403
13.84.2.1 L4_mem_op_widths . . . 403
13.84.3 Function Documentation . . . 404
13.84.3.1 l4_mem_read() . . . 404
13.84.3.2 l4_mem_write() . . . 405
13.85 Memory related . . . 405
13.85.1 Detailed Description . . . 406
13.85.2 Macro Definition Documentation . . . 407
13.85.2.1 L4_LOG2_PAGESIZE . . . 407
13.85.2.2 L4_LOG2_SUPERPAGESIZE . . . 407
13.85.2.3 L4_PAGEMASK . . . 407
13.85.2.4 L4_SUPERPAGEMASK . . . 407
13.85.2.5 L4_SUPERPAGESIZE . . . 408
13.85.3 Enumeration Type Documentation . . . 408
13.85.3.1 l4_addr_consts_t . . . 408
13.85.4 Function Documentation . . . 408
13.85.4.1 l4_bytes_to_mwords() . . . 408
13.85.4.2 l4_round_page() . . . 409
13.85.4.3 l4_round_size() . . . 409
13.85.4.4 l4_trunc_page() . . . 410
13.85.4.5 l4_trunc_size() . . . 411
13.86 Message Items . . . 411
13.86.1 Detailed Description . . . 412
13.86.2 Enumeration Type Documentation . . . 412
13.86.2.1 l4_msg_item_consts_t . . . 412
13.86.3 Function Documentation . . . 413
13.86.3.1 l4_map_control() . . . 413
13.86.3.2 l4_map_obj_control() . . . 414
13.87 Message Registers (MRs) . . . 415
13.87.1 Detailed Description . . . 415
13.88 Message Tag . . . 415
13.88.1 Detailed Description . . . 417
13.88.2 Typedef Documentation . . . 417
13.88.2.1 l4_msgtag_t . . . 417
13.88.3 Enumeration Type Documentation . . . 417
13.88.3.1 l4_msgtag_flags . . . 417
13.88.3.2 l4_msgtag_protocol . . . 418
13.88.3.3 L4_platform_ctl_proto . . . 418
13.88.4 Function Documentation . . . 419
13.88.4.1 l4_msgtag() . . . 419
13.88.4.2 l4_msgtag_flags() . . . 420

13.88.4.3 l4_msgtag_has_error() . . . 420
13.88.4.4 l4_msgtag_is_exception() . . . 421
13.88.4.5 l4_msgtag_is_io_page_fault() . . . 422
13.88.4.6 l4_msgtag_is_page_fault() . . . 422
13.88.4.7 l4_msgtag_is_preemption() . . . 423
13.88.4.8 l4_msgtag_is_sigma0() . . . 424
13.88.4.9 l4_msgtag_is_sys_exception() . . . 424
13.88.4.10 l4_msgtag_items() . . . 425
13.88.4.11 l4_msgtag_label() . . . 426
13.88.4.12 l4_msgtag_words() . . . 427
13.89 Name-space API . . . 428
13.89.1 Detailed Description . . . 428
13.90 Namespace interface . . . 428
13.90.1 Detailed Description . . . 429
13.90.2 Enumeration Type Documentation . . . 429
13.90.2.1 l4re_ns_register_flags . . . 429
13.90.3 Function Documentation . . . 429
13.90.3.1 l4re_ns_query_srv() . . . 429
13.90.3.2 l4re_ns_query_to_srv() . . . 430
13.90.3.3 l4re_ns_register_obj_srv() . . . 431
13.91 Object Invocation . . . 432
13.91.1 Detailed Description . . . 433
13.91.2 Enumeration Type Documentation . . . 434
13.91.2.1 l4_syscall_flags_t . . . 434
13.91.3 Function Documentation . . . 435
13.91.3.1 l4_ipc() . . . 435
13.91.3.2 l4_ipc_call() . . . 436
13.91.3.3 l4_ipc_receive() . . . 437
13.91.3.4 l4_ipc_reply_and_wait() . . . 439
13.91.3.5 l4_ipc_send() . . . 440
13.91.3.6 l4_ipc_send_and_wait() . . . 441
13.91.3.7 l4_ipc_sleep() . . . 442
13.91.3.8 l4_ipc_wait() . . . 443
13.91.3.9 l4_sndfpage_add() . . . 444
13.92 Parent API . . . 445
13.92.1 Detailed Description . . . 445
13.93 Platform Control C API . . . 446
13.93.1 Detailed Description . . . 446
13.93.2 Function Documentation . . . 446
13.93.2.1 l4_platform_ctl_cpu_allow_shutdown() . . . 446
13.93.2.2 l4_platform_ctl_cpu_disable() . . . 447
13.93.2.3 l4_platform_ctl_cpu_enable() . . . 447

13.93.2.4 l4_platform_ctl_system_shutdown() . . . 448
13.93.2.5 l4_platform_ctl_system_suspend() . . . 448
13.94 Producer . . . 449
13.94.1 Detailed Description . . . 449
13.94.2 Function Documentation . . . 449
13.94.2.1 l4shmc_chunk_ready() . . . 449
13.94.2.2 l4shmc_chunk_ready_sig() . . . 450
13.94.2.3 l4shmc_chunk_try_to_take() . . . 450
13.94.2.4 l4shmc_is_chunk_clear() . . . 451
13.95 Producer . . . 451
13.95.1 Detailed Description . . . 451
13.95.2 Function Documentation . . . 451
13.95.2.1 l4shmc_trigger() . . . 452
13.96 Random number support . . . 452
13.96.1 Detailed Description . . . 452
13.96.2 Function Documentation . . . 452
13.96.2.1 l4util_rand() . . . 453
13.96.2.2 l4util_srand() . . . 453
13.97 Realtime API . . . 453
13.98 Region map API . . . 453
13.98.1 Detailed Description . . . 454
13.99 Region map interface . . . 454
13.99.1 Detailed Description . . . 455
13.99.2 Enumeration Type Documentation . . . 456
13.99.2.1 l4re_rm_flags_values . . . 456
13.99.3 Function Documentation . . . 456
13.99.3.1 l4re_rm_attach() . . . 456
13.99.3.2 l4re_rm_attach_srv() . . . 458
13.99.3.3 l4re_rm_detach() . . . 458
13.99.3.4 l4re_rm_detach_ds() . . . 459
13.99.3.5 l4re_rm_detach_ds_unmap() . . . 460
13.99.3.6 l4re_rm_detach_srv() . . . 461
13.99.3.7 l4re_rm_detach_unmap() . . . 462
13.99.3.8 l4re_rm_find() . . . 463
13.99.3.9 l4re_rm_find_srv() . . . 464
13.99.3.10 l4re_rm_free_area() . . . 465
13.99.3.11 l4re_rm_free_area_srv() . . . 466
13.99.3.12 l4re_rm_reserve_area() . . . 466
13.99.3.13 l4re_rm_reserve_area_srv() . . . 468
13.99.3.14 l4re_rm_show_lists() . . . 468
13.100 Scheduler . . . 469
13.100.1 Detailed Description . . . 470

13.100.2 Enumeration Type Documentation . . . 470
13.100.2.1 L4_scheduler_classes . . . 470
13.100.2.2 L4_scheduler_ops . . . 470
13.100.3 Function Documentation . . . 471
13.100.3.1 l4_sched_cpu_set() . . . 471
13.100.3.2 l4_scheduler_idle_time() . . . 472
13.100.3.3 l4_scheduler_info() . . . 472
13.100.3.4 l4_scheduler_is_online() . . . 473
13.100.3.5 l4_scheduler_run_thread() . . . 473
13.101 Server-Side IPC framework . . . 474
13.101.1 Detailed Description . . . 475
13.101.2 Enumeration Type Documentation . . . 475
13.101.2.1 Reply_mode . . . 475
13.102 Shared Memory Library . . . 476
13.102.1 Detailed Description . . . 477
13.102.2 Function Documentation . . . 477
13.102.2.1 l4shmc_area_overhead() . . . 477
13.102.2.2 l4shmc_area_size() . . . 477
13.102.2.3 l4shmc_area_size_free() . . . 478
13.102.2.4 l4shmc_attach() . . . 478
13.102.2.5 l4shmc_attach_to() . . . 479
13.102.2.6 l4shmc_chunk_overhead() . . . 479
13.102.2.7 l4shmc_connect_chunk_signal() . . . 480
13.102.2.8 l4shmc_create() . . . 481
13.103 Sigma0 API . . . 481
13.103.1 Detailed Description . . . 483
13.103.2 Enumeration Type Documentation . . . 483
13.103.2.1 l4sigma0_return_flags_t . . . 483
13.103.3 Function Documentation . . . 483
13.103.3.1 l4sigma0_debug_dump() . . . 483
13.103.3.2 l4sigma0_map_anypage() . . . 484
13.103.3.3 l4sigma0_map_errstr() . . . 484
13.103.3.4 l4sigma0_map_iomem() . . . 485
13.103.3.5 l4sigma0_map_kip() . . . 485
13.103.3.6 l4sigma0_map_mem() . . . 486
13.103.3.7 l4sigma0_new_client() . . . 486
13.104 Signals . . . 487
13.104.1 Detailed Description . . . 488
13.104.2 Function Documentation . . . 488
13.104.2.1 l4shmc_add_signal() . . . 488
13.104.2.2 l4shmc_attach_signal() . . . 488
13.104.2.3 l4shmc_attach_signal_to() . . . 489

13.104.2.4 l4shmc_check_magic() . . . 489
13.104.2.5 l4shmc_get_signal_to() . . . 490
13.104.2.6 l4shmc_signal_cap() . . . 490
13.105 Small C++ Template Library . . . 491
13.105.1 Detailed Description . . . 492
13.105.2 Function Documentation . . . 492
13.105.2.1 max() . . . 492
13.105.2.2 min() . . . 493
13.105.2.3 operator new() . . . 494
13.106 Task . . . 494
13.106.1 Detailed Description . . . 495
13.106.2 Enumeration Type Documentation . . . 495
13.106.2.1 l4_unmap_flags_t . . . 495
13.106.3 Function Documentation . . . 496
13.106.3.1 l4_task_add_ku_mem() . . . 496
13.106.3.2 l4_task_cap_equal() . . . 497
13.106.3.3 l4_task_cap_valid() . . . 497
13.106.3.4 l4_task_delete_obj() . . . 498
13.106.3.5 l4_task_map() . . . 498
13.106.3.6 l4_task_release_cap() . . . 499
13.106.3.7 l4_task_unmap() . . . 500
13.106.3.8 l4_task_unmap_batch() . . . 501
13.106.3.9 l4_task_vgicc_map() . . . 501
13.107 Thread . . . 502
13.107.1 Detailed Description . . . 504
13.107.2 Enumeration Type Documentation . . . 504
13.107.2.1 L4_thread_control_flags . . . 504
13.107.2.2 L4_thread_control_mr_indices . . . 505
13.107.2.3 L4_thread_ex_regs_flags . . . 505
13.107.3 Function Documentation . . . 505
13.107.3.1 l4_thread_arm_set_tpidruro() . . . 505
13.107.3.2 l4_thread_ex_regs() . . . 507
13.107.3.3 l4_thread_ex_regs_ret() . . . 508
13.107.3.4 l4_thread_ex_regs_ret_u() . . . 509
13.107.3.5 l4_thread_ex_regs_u() . . . 510
13.107.3.6 l4_thread_modify_sender_add() . . . 511
13.107.3.7 l4_thread_modify_sender_commit() . . . 512
13.107.3.8 l4_thread_modify_sender_start() . . . 512
13.107.3.9 l4_thread_register_del_irq() . . . 513
13.107.3.10 l4_thread_stats_time() . . . 513
13.107.3.11 l4_thread_switch() . . . 514
13.107.3.12 l4_thread_vcpu_control() . . . 514

13.107.3.13 l4_thread_vcpu_control_ext() . . . 515
13.107.3.14 l4_thread_vcpu_control_ext_u() . . . 516
13.107.3.15 l4_thread_vcpu_control_u() . . . 517
13.107.3.16 l4_thread_vcpu_resume_commit() . . . 518
13.107.3.17 l4_thread_vcpu_resume_start() . . . 519
13.107.3.18 l4_thread_yield() . . . 519
13.108 Thread Control Registers (TCRs) . . . 520
13.108.1 Detailed Description . . . 520
13.109 Thread control . . . 520
13.109.1 Detailed Description . . . 521
13.109.2 Function Documentation . . . 521
13.109.2.1 l4_thread_control_alien() . . . 521
13.109.2.2 l4_thread_control_bind() . . . 522
13.109.2.3 l4_thread_control_commit() . . . 523
13.109.2.4 l4_thread_control_exc_handler() . . . 523
13.109.2.5 l4_thread_control_pager() . . . 524
13.109.2.6 l4_thread_control_start() . . . 524
13.109.2.7 l4_thread_control_ux_host_syscall() . . . 525
13.110 Timeouts . . . 525
13.110.1 Detailed Description . . . 527
13.110.2 Macro Definition Documentation . . . 527
13.110.2.1 L4_IPC_TIMEOUT_0 . . . 527
13.110.3 Typedef Documentation . . . 527
13.110.3.1 l4_timeout_s . . . 527
13.110.3.2 l4_timeout_t . . . 527
13.110.4 Function Documentation . . . 527
13.110.4.1 l4_ipc_timeout() . . . 527
13.110.4.2 l4_rcv_timeout() . . . 528
13.110.4.3 l4_snd_timeout() . . . 528
13.110.4.4 l4_timeout() . . . 528
13.110.4.5 l4_timeout_abs() . . . 529
13.110.4.6 l4_timeout_get() . . . 530
13.110.4.7 l4_timeout_is_absolute() . . . 531
13.110.4.8 l4_timeout_rel() . . . 532
13.110.4.9 l4_timeout_rel_get() . . . 532
13.110.4.10 l4_utcb_mr64_idx() . . . 533
13.111 Timestamp Counter . . . 533
13.111.1 Detailed Description . . . 534
13.111.2 Function Documentation . . . 534
13.111.2.1 l4_busy_wait_ns() . . . 534
13.111.2.2 l4_busy_wait_us() . . . 535
13.111.2.3 l4_calibrate_tsc() . . . 536

13.111.2.4 l4_get_hz() . . . 536
13.111.2.5 l4_ns_to_tsc() . . . 536
13.111.2.6 l4_rdpmc() . . . 537
13.111.2.7 l4_rdpmc_32() . . . 538
13.111.2.8 l4_rdtsc() . . . 538
13.111.2.9 l4_rdtsc_32() . . . 539
13.111.2.10 l4_tsc_init() . . . 539
13.111.2.11 l4_tsc_to_ns() . . . 539
13.111.2.12 l4_tsc_to_s_and_ns() . . . 540
13.111.2.13 l4_tsc_to_us() . . . 540
13.112 Utility Functions . . . 541
13.112.1 Detailed Description . . . 542
13.112.2 Function Documentation . . . 542
13.112.2.1 l4_sleep() . . . 543
13.112.2.2 l4_touch_ro() . . . 544
13.112.2.3 l4_touch_rw() . . . 544
13.112.2.4 l4_usleep() . . . 545
13.112.2.5 l4util_micros2l4to() . . . 546
13.112.2.6 l4util_splitlog2_hdl() . . . 546
13.112.2.7 l4util_splitlog2_size() . . . 547
13.113 VM API for SVM . . . 548
13.113.1 Detailed Description . . . 549
13.114 VM API for TZ . . . 549
13.114.1 Detailed Description . . . 549
13.115 VM API for VMX . . . 549
13.115.1 Detailed Description . . . 551
13.115.2 Enumeration Type Documentation . . . 551
13.115.2.1 anonymous enum . . . 551
13.115.2.2 L4_vm_vmx_caps_regs . . . 552
13.115.2.3 L4_vm_vmx_dfl1_regs . . . 552
13.115.3 Function Documentation . . . 552
13.115.3.1 l4_vm_vmx_clear() . . . 553
13.115.3.2 l4_vm_vmx_field_len() . . . 553
13.115.3.3 l4_vm_vmx_field_order() . . . 554
13.115.3.4 l4_vm_vmx_get_caps() . . . 554
13.115.3.5 l4_vm_vmx_get_caps_default1() . . . 555
13.115.3.6 l4_vm_vmx_get_cr2_index() . . . 555
13.115.3.7 l4_vm_vmx_ptr_load() . . . 556
13.115.3.8 l4_vm_vmx_read() . . . 556
13.115.3.9 l4_vm_vmx_read_16() . . . 557
13.115.3.10 l4_vm_vmx_read_32() . . . 558
13.115.3.11 l4_vm_vmx_read_64() . . . 559

13.115.3.12 l4_vm_vmx_read_nat() . . . 559
13.115.3.13 l4_vm_vmx_write() . . . 560
13.115.3.14 l4_vm_vmx_write_16() . . . 561
13.115.3.15 l4_vm_vmx_write_32() . . . 562
13.115.3.16 l4_vm_vmx_write_64() . . . 562
13.115.3.17 l4_vm_vmx_write_nat() . . . 563
13.116 Vbus API . . . 563
13.116.1 Detailed Description . . . 564
13.117 Video API . . . 565
13.117.1 Detailed Description . . . 566
13.117.2 Typedef Documentation . . . 566
13.117.2.1 l4re_video_view_t . . . 566
13.117.3 Enumeration Type Documentation . . . 567
13.117.3.1 l4re_video_goos_info_flags_t . . . 567
13.117.3.2 l4re_video_view_info_flags_t . . . 567
13.117.4 Function Documentation . . . 567
13.117.4.1 l4re_video_goos_create_buffer() . . . 568
13.117.4.2 l4re_video_goos_create_view() . . . 568
13.117.4.3 l4re_video_goos_delete_buffer() . . . 568
13.117.4.4 l4re_video_goos_delete_view() . . . 570
13.117.4.5 l4re_video_goos_get_static_buffer() . . . 570
13.117.4.6 l4re_video_goos_get_view() . . . 570
13.117.4.7 l4re_video_goos_info() . . . 571
13.117.4.8 l4re_video_goos_refresh() . . . 571
13.117.4.9 l4re_video_view_get_info() . . . 572
13.117.4.10 l4re_video_view_refresh() . . . 572
13.117.4.11 l4re_video_view_set_info() . . . 573
13.117.4.12 l4re_video_view_set_viewport() . . . 573
13.117.4.13 l4re_video_view_stack() . . . 573
13.118 Virtual Console . . . 574
13.118.1 Detailed Description . . . 575
13.118.2 Enumeration Type Documentation . . . 576
13.118.2.1 L4_vcon_i_flags . . . 576
13.118.2.2 L4_vcon_l_flags . . . 576
13.118.2.3 L4_vcon_o_flags . . . 576
13.118.2.4 L4_vcon_size_consts . . . 577
13.118.3 Function Documentation . . . 577
13.118.3.1 l4_vcon_get_attr() . . . 577
13.118.3.2 l4_vcon_get_attr_u() . . . 578
13.118.3.3 l4_vcon_read() . . . 579
13.118.3.4 l4_vcon_read_u() . . . 580
13.118.3.5 l4_vcon_read_with_flags() . . . 580

13.118.3.6 l4_vcon_send() . . . 581
13.118.3.7 l4_vcon_send_u() . . . 582
13.118.3.8 l4_vcon_set_attr() . . . 583
13.118.3.9 l4_vcon_set_attr_raw() . . . 584
13.118.3.10 l4_vcon_set_attr_u() . . . 585
13.118.3.11 l4_vcon_write() . . . 585
13.118.3.12 l4_vcon_write_u() . . . 586
13.119 Virtual Machines . . . 587
13.119.1 Detailed Description . . . 588
13.120 Virtual Registers (UTCBs) . . . 588
13.120.1 Detailed Description . . . 589
13.120.2 Typedef Documentation . . . 590
13.120.2.1 l4_utcb_t . . . 590
13.120.3 Function Documentation . . . 590
13.120.3.1 l4_utcb_br() . . . 590
13.120.3.2 l4_utcb_mr() . . . 591
13.120.3.3 l4_utcb_tcr() . . . 591
13.121 amd64 Virtual Registers (UTCB) . . . 592
13.121.1 Detailed Description . . . 592
13.122 vCPU API . . . 592
13.122.1 Detailed Description . . . 593
13.122.2 Enumeration Type Documentation . . . 594
13.122.2.1 L4_vcpu_state_flags . . . 594
13.122.2.2 L4_vcpu_state_offset . . . 596
13.122.2.3 L4_vcpu_sticky_flags . . . 596
13.123 vCPU Support Library . . . 596
13.123.1 Detailed Description . . . 597
13.123.2 Function Documentation . . . 597
13.123.2.1 l4vcpu_irq_disable() . . . 597
13.123.2.2 l4vcpu_irq_disable_save() . . . 598
13.123.2.3 l4vcpu_irq_enable() . . . 599
13.123.2.4 l4vcpu_irq_restore() . . . 600
13.123.2.5 l4vcpu_is_irq_entry() . . . 601
13.123.2.6 l4vcpu_is_page_fault_entry() . . . 602
13.123.2.7 l4vcpu_print_state() . . . 602
13.123.2.8 l4vcpu_wait_for_event() . . . 603
13.124 x86 Virtual Registers (UTCB) . . . 604
13.124.1 Detailed Description . . . 604
13.124.2 Enumeration Type Documentation . . . 604
13.124.2.1 L4_utcb_consts_x86 . . . 604

14 Namespace Documentation **607**

14.1 cxx Namespace Reference . . . 607
14.1.1 Detailed Description . . . 609
14.2 cxx::Bits Namespace Reference . . . 609
14.2.1 Detailed Description . . . 610
14.3 L4 Namespace Reference . . . 610
14.3.1 Detailed Description . . . 613
14.3.2 Enumeration Type Documentation . . . 614
14.3.2.1 anonymous enum . . . 614
14.3.3 Function Documentation . . . 614
14.3.3.1 cap_cast() [1/2] . . . 614
14.3.3.2 cap_cast() [2/2] . . . 615
14.3.3.3 cap_dynamic_cast() . . . 615
14.3.3.4 cap_reinterpret_cast() [1/2] . . . 616
14.3.3.5 cap_reinterpret_cast() [2/2] . . . 617
14.3.3.6 round_order() . . . 617
14.3.3.7 throw_ipc_exception() [1/2] . . . 618
14.3.3.8 throw_ipc_exception() [2/2] . . . 619
14.3.3.9 trunc_order() . . . 620
14.4 L4::Ipc Namespace Reference . . . 620
14.4.1 Detailed Description . . . 622
14.4.2 Function Documentation . . . 622
14.4.2.1 buf_cp_in() . . . 623
14.4.2.2 buf_cp_out() . . . 623
14.4.2.3 buf_in() . . . 624
14.4.2.4 make_cap() . . . 624
14.4.2.5 make_cap_full() . . . 625
14.4.2.6 make_cap_rw() . . . 625
14.4.2.7 make_cap_rws() . . . 626
14.4.2.8 msg_ptr() . . . 627
14.4.2.9 read() . . . 627
14.4.2.10 str_cp_in() . . . 627
14.5 L4::Ipc::Msg Namespace Reference . . . 628
14.5.1 Detailed Description . . . 629
14.5.2 Enumeration Type Documentation . . . 629
14.5.2.1 anonymous enum . . . 629
14.5.3 Function Documentation . . . 630
14.5.3.1 align_to() [1/2] . . . 630
14.5.3.2 align_to() [2/2] . . . 630
14.5.3.3 check_size() [1/2] . . . 631
14.5.3.4 check_size() [2/2] . . . 632
14.5.3.5 msg_add() . . . 632
14.5.3.6 msg_get() . . . 633

14.6 L4::Ipc_svr Namespace Reference . . . 633
14.6.1 Detailed Description . . . 634
14.7 L4::Typeid Namespace Reference . . . 634
14.7.1 Detailed Description . . . 635
14.8 L4::Types Namespace Reference . . . 635
14.8.1 Detailed Description . . . 636
14.9 L4Re Namespace Reference . . . 636
14.9.1 Detailed Description . . . 638
14.9.2 Typedef Documentation . . . 638
14.9.2.1 Shared_cap . . . 638
14.9.2.2 shared_cap . . . 638
14.9.2.3 Shared_del_cap . . . 639
14.9.2.4 shared_del_cap . . . 639
14.9.2.5 Unique_cap . . . 640
14.9.2.6 unique_cap . . . 640
14.9.2.7 Unique_del_cap . . . 641
14.9.2.8 unique_del_cap . . . 641
14.9.3 Function Documentation . . . 641
14.9.3.1 chkcap() . . . 642
14.9.3.2 chkipc() . . . 643
14.9.3.3 chksys() [1/3] . . . 644
14.9.3.4 chksys() [2/3] . . . 645
14.9.3.5 chksys() [3/3] . . . 646
14.9.3.6 make_shared_cap() . . . 647
14.9.3.7 make_shared_del_cap() . . . 648
14.9.3.8 make_unique_cap() . . . 649
14.9.3.9 make_unique_del_cap() . . . 650
14.9.3.10 throw_error() . . . 650
14.10 L4Re::Util Namespace Reference . . . 651
14.10.1 Detailed Description . . . 653
14.10.2 Typedef Documentation . . . 653
14.10.2.1 Shared_cap . . . 654
14.10.2.2 shared_cap . . . 655
14.10.2.3 Shared_del_cap . . . 656
14.10.2.4 shared_del_cap . . . 656
14.10.2.5 Unique_cap . . . 657
14.10.2.6 unique_cap . . . 658
14.10.2.7 Unique_del_cap . . . 658
14.10.2.8 unique_del_cap . . . 659
14.10.3 Function Documentation . . . 660
14.10.3.1 make_shared_cap() . . . 660
14.10.3.2 make_shared_del_cap() . . . 660

14.10.3.3 make_unique_cap() . . . 661
14.10.3.4 make_unique_del_cap() . . . 661
14.11 L4Re::Vfs Namespace Reference . . . 662
14.11.1 Detailed Description . . . 663
14.12 L4vbus Namespace Reference . . . 663
14.12.1 Detailed Description . . . 663
14.13 L4virtio Namespace Reference . . . 664
14.13.1 Detailed Description . . . 664

15 Data Structure Documentation **665**
15.1 cxx::Auto_ptr< T > Class Template Reference . . . 665
15.1.1 Detailed Description . . . 666
15.1.2 Constructor & Destructor Documentation . . . 666
15.1.2.1 Auto_ptr() [1/2] . . . 666
15.1.2.2 Auto_ptr() [2/2] . . . 667
15.1.3 Member Function Documentation . . . 667
15.1.3.1 get() . . . 667
15.1.3.2 operator=() . . . 667
15.1.3.3 release() . . . 668
15.2 cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > Class Template Reference . . . 669
15.2.1 Detailed Description . . . 671
15.2.2 Constructor & Destructor Documentation . . . 671
15.2.2.1 Avl_map() . . . 671
15.2.3 Member Function Documentation . . . 672
15.2.3.1 insert() . . . 672
15.2.3.2 operator[]() [1/2] . . . 673
15.2.3.3 operator[]() [2/2] . . . 674
15.3 cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > Class Template Reference . . . 674
15.3.1 Detailed Description . . . 676
15.4 cxx::Avl_tree< Node, Get_key, Compare > Class Template Reference . . . 677
15.4.1 Detailed Description . . . 680
15.4.2 Member Typedef Documentation . . . 681
15.4.2.1 Iterator . . . 681
15.4.3 Member Function Documentation . . . 681
15.4.3.1 insert() . . . 681
15.4.3.2 remove() . . . 682
15.5 cxx::Avl_tree_node Class Reference . . . 682
15.5.1 Detailed Description . . . 684
15.6 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference . . . 685
15.6.1 Detailed Description . . . 686
15.6.2 Member Enumeration Documentation . . . 687
15.6.2.1 anonymous enum . . . 687

15.6.3 Member Function Documentation . . . 687
15.6.3.1 alloc() . . . 687
15.6.3.2 free() . . . 688
15.6.3.3 free_objects() . . . 689
15.6.3.4 total_objects() . . . 690
15.7 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i Struct Reference . . . 691
15.7.1 Detailed Description . . . 692
15.8 cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference . . . 692
15.8.1 Detailed Description . . . 694
15.8.2 Member Enumeration Documentation . . . 695
15.8.2.1 anonymous enum . . . 695
15.8.3 Member Function Documentation . . . 695
15.8.3.1 alloc() . . . 695
15.8.3.2 free() . . . 696
15.8.3.3 free_objects() . . . 696
15.8.3.4 total_objects() . . . 697
15.9 cxx::Bitfield< T, LSB, MSB > Class Template Reference . . . 698
15.9.1 Detailed Description . . . 699
15.9.2 Member Typedef Documentation . . . 699
15.9.2.1 Bits_type . . . 699
15.9.2.2 Shift_type . . . 700
15.9.3 Member Enumeration Documentation . . . 700
15.9.3.1 anonymous enum . . . 700
15.9.3.2 Masks . . . 700
15.9.4 Member Function Documentation . . . 700
15.9.4.1 get() . . . 701
15.9.4.2 get_unshifted() . . . 701
15.9.4.3 set() . . . 702
15.9.4.4 set_dirty() . . . 703
15.9.4.5 set_unshifted() . . . 704
15.9.4.6 set_unshifted_dirty() . . . 704
15.9.4.7 val() . . . 705
15.9.4.8 val_dirty() . . . 707
15.9.4.9 val_unshifted() . . . 708
15.10 cxx::Bitfield< T, LSB, MSB >::Value< TT > Class Template Reference . . . 709
15.10.1 Detailed Description . . . 710
15.11 cxx::Bitfield< T, LSB, MSB >::Value_base< TT > Class Template Reference . . . 710
15.11.1 Detailed Description . . . 711
15.12 cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Class Template Reference . . . 712
15.12.1 Detailed Description . . . 713
15.13 cxx::Bitmap< BITS > Class Template Reference . . . 713
15.13.1 Detailed Description . . . 715

15.13.2 Member Function Documentation . . . 716
15.13.2.1 scan_zero() . . . 716
15.14 cxx::Bitmap_base Class Reference . . . 717
15.14.1 Detailed Description . . . 719
15.14.2 Member Enumeration Documentation . . . 719
15.14.2.1 anonymous enum . . . 719
15.14.3 Member Function Documentation . . . 720
15.14.3.1 bit() [1/2] . . . 720
15.14.3.2 bit() [2/2] . . . 720
15.14.3.3 bit_index() . . . 721
15.14.3.4 clear_bit() . . . 722
15.14.3.5 operator[]() [1/2] . . . 722
15.14.3.6 operator[]() [2/2] . . . 722
15.14.3.7 scan_zero() . . . 723
15.14.3.8 set_bit() . . . 724
15.14.3.9 word_index() . . . 724
15.15 cxx::Bitmap_base::Bit Class Reference . . . 725
15.15.1 Detailed Description . . . 725
15.16 cxx::Bitmap_base::Char< BITS > Class Template Reference . . . 726
15.16.1 Detailed Description . . . 726
15.17 cxx::Bitmap_base::Word< BITS > Class Template Reference . . . 726
15.17.1 Detailed Description . . . 727
15.18 cxx::Bits::Avl_map_get_key< KEY_TYPE > Struct Template Reference . . . 728
15.18.1 Detailed Description . . . 728
15.19 cxx::Bits::Avl_set_get_key< KEY_TYPE > Struct Template Reference . . . 728
15.19.1 Detailed Description . . . 729
15.20 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > Class Template Reference 729
15.20.1 Detailed Description . . . 732
15.20.2 Member Enumeration Documentation . . . 733
15.20.2.1 anonymous enum . . . 733
15.20.3 Constructor & Destructor Documentation . . . 733
15.20.3.1 Base_avl_set() [1/2] . . . 733
15.20.3.2 Base_avl_set() [2/2] . . . 734
15.20.4 Member Function Documentation . . . 734
15.20.4.1 begin() [1/2] . . . 734
15.20.4.2 begin() [2/2] . . . 735
15.20.4.3 end() [1/2] . . . 736
15.20.4.4 end() [2/2] . . . 737
15.20.4.5 erase() . . . 737
15.20.4.6 find_node() . . . 738
15.20.4.7 insert() . . . 739
15.20.4.8 lower_bound_node() . . . 740

15.20.4.9 rbegin() [1/2] . . . 740
15.20.4.10 rbegin() [2/2] . . . 741
15.20.4.11 remove() . . . 741
15.20.4.12 rend() [1/2] . . . 742
15.20.4.13 rend() [2/2] . . . 743
15.21 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node Class Reference 743
15.21.1 Detailed Description . . . 744
15.21.2 Member Function Documentation . . . 744
15.21.2.1 operator*() . . . 744
15.21.2.2 operator->() . . . 745
15.21.2.3 valid() . . . 745
15.22 cxx::Bits::Basic_list< POLICY > Class Template Reference . . . 745
15.22.1 Detailed Description . . . 748
15.22.2 Member Function Documentation . . . 748
15.22.2.1 clear() . . . 748
15.22.2.2 iter() . . . 748
15.23 cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference . . . 749
15.23.1 Detailed Description . . . 752
15.23.2 Member Function Documentation . . . 752
15.23.2.1 begin() [1/2] . . . 753
15.23.2.2 begin() [2/2] . . . 753
15.23.2.3 dir() [1/2] . . . 754
15.23.2.4 dir() [2/2] . . . 755
15.23.2.5 end() [1/2] . . . 755
15.23.2.6 end() [2/2] . . . 756
15.23.2.7 find() . . . 756
15.23.2.8 find_node() . . . 757
15.23.2.9 lower_bound_node() . . . 758
15.23.2.10 rbegin() [1/2] . . . 759
15.23.2.11 rbegin() [2/2] . . . 760
15.23.2.12 remove_all() . . . 760
15.23.2.13 remove_tree() . . . 761
15.23.2.14 rend() [1/2] . . . 762
15.23.2.15 rend() [2/2] . . . 762
15.24 cxx::Bits::Bst_node Class Reference . . . 763
15.24.1 Detailed Description . . . 765
15.25 cxx::Bits::Direction Struct Reference . . . 765
15.25.1 Detailed Description . . . 766
15.25.2 Member Enumeration Documentation . . . 767
15.25.2.1 Direction_e . . . 767
15.25.3 Member Function Documentation . . . 767
15.25.3.1 operator"!() . . . 767

15.26 cxx::Bits::Smart_ptr_list< ITEM > Class Template Reference . . . 768
15.26.1 Detailed Description . . . 769
15.26.2 Member Function Documentation . . . 770
15.26.2.1 pop_front() . . . 770
15.27 cxx::Bits::Smart_ptr_list_item< T, STORE_T > Class Template Reference . . . 770
15.27.1 Detailed Description . . . 771
15.28 cxx::H_list< T, POLICY > Class Template Reference . . . 772
15.28.1 Detailed Description . . . 774
15.28.2 Member Function Documentation . . . 774
15.28.2.1 erase() . . . 774
15.28.2.2 insert() . . . 775
15.28.2.3 insert_after() . . . 776
15.28.2.4 insert_before() . . . 777
15.28.2.5 iter() . . . 777
15.28.2.6 pop_front() . . . 778
15.28.2.7 remove() . . . 779
15.28.2.8 replace() . . . 780
15.29 cxx::H_list_item_t< ELEM_TYPE > Class Template Reference . . . 780
15.29.1 Detailed Description . . . 782
15.29.2 Constructor & Destructor Documentation . . . 782
15.29.2.1 H_list_item_t() . . . 782
15.29.2.2 ~H_list_item_t() . . . 783
15.30 cxx::H_list_t< T > Struct Template Reference . . . 783
15.30.1 Detailed Description . . . 785
15.31 cxx::List< D, Alloc > Class Template Reference . . . 786
15.31.1 Detailed Description . . . 787
15.31.2 Member Function Documentation . . . 787
15.31.2.1 operator[]() [1/2] . . . 787
15.31.2.2 operator[]() [2/2] . . . 787
15.32 cxx::List< D, Alloc >::Iter Class Reference . . . 788
15.32.1 Detailed Description . . . 788
15.33 cxx::List_alloc Class Reference . . . 789
15.33.1 Detailed Description . . . 789
15.33.2 Constructor & Destructor Documentation . . . 789
15.33.2.1 List_alloc() . . . 790
15.33.3 Member Function Documentation . . . 790
15.33.3.1 alloc() . . . 790
15.33.3.2 alloc_max() . . . 790
15.33.3.3 avail() . . . 791
15.33.3.4 free() . . . 792
15.34 cxx::List_item Class Reference . . . 792
15.34.1 Detailed Description . . . 793

15.34.2 Member Function Documentation . . . 793
15.34.2.1 push_back() . . . 793
15.34.2.2 push_front() . . . 794
15.34.2.3 remove() . . . 794
15.35 cxx::List_item::Iter Class Reference . . . 795
15.35.1 Detailed Description . . . 797
15.36 cxx::List_item::T_iter< T, Poly > Class Template Reference . . . 797
15.36.1 Detailed Description . . . 799
15.37 cxx::Lt_functor< Obj > Struct Template Reference . . . 800
15.37.1 Detailed Description . . . 800
15.38 cxx::New_allocator< _Type > Class Template Reference . . . 800
15.38.1 Detailed Description . . . 801
15.39 cxx::Nothrow Class Reference . . . 802
15.39.1 Detailed Description . . . 802
15.40 cxx::Pair< First, Second > Struct Template Reference . . . 802
15.40.1 Detailed Description . . . 803
15.40.2 Constructor & Destructor Documentation . . . 803
15.40.2.1 Pair() . . . 803
15.41 cxx::Pair_first_compare< Cmp, Typ > Class Template Reference . . . 804
15.41.1 Detailed Description . . . 804
15.41.2 Constructor & Destructor Documentation . . . 805
15.41.2.1 Pair_first_compare() . . . 805
15.41.3 Member Function Documentation . . . 805
15.41.3.1 operator()() . . . 805
15.42 cxx::Ref_obj_list_item< T > Struct Template Reference . . . 806
15.42.1 Detailed Description . . . 807
15.43 cxx::Ref_ptr< T, CNT > Class Template Reference . . . 807
15.43.1 Detailed Description . . . 808
15.43.2 Constructor & Destructor Documentation . . . 809
15.43.2.1 Ref_ptr() [1/3] . . . 809
15.43.2.2 Ref_ptr() [2/3] . . . 810
15.43.2.3 Ref_ptr() [3/3] . . . 810
15.43.3 Member Function Documentation . . . 810
15.43.3.1 get() . . . 810
15.43.3.2 ptr() . . . 811
15.43.3.3 release() . . . 811
15.44 cxx::S_list< T, POLICY > Class Template Reference . . . 811
15.44.1 Detailed Description . . . 814
15.44.2 Member Function Documentation . . . 814
15.44.2.1 pop_front() . . . 814
15.45 cxx::Slab< Type, Slab_size, Max_free, Alloc > Class Template Reference . . . 815
15.45.1 Detailed Description . . . 817

15.45.2 Member Function Documentation . . . 817
15.45.2.1 alloc() . . . 817
15.45.2.2 free() . . . 818
15.46 cxx::Slab_static< Type, Slab_size, Max_free, Alloc > Class Template Reference . . . 818
15.46.1 Detailed Description . . . 820
15.46.2 Member Function Documentation . . . 822
15.46.2.1 alloc() . . . 822
15.47 cxx::static_vector< T, IDX > Class Template Reference . . . 823
15.47.1 Detailed Description . . . 823
15.48 cxx::String Class Reference . . . 824
15.48.1 Detailed Description . . . 826
15.48.2 Constructor & Destructor Documentation . . . 827
15.48.2.1 String() . . . 827
15.48.3 Member Function Documentation . . . 827
15.48.3.1 find() . . . 827
15.48.3.2 from_dec() . . . 828
15.48.3.3 from_hex() . . . 829
15.48.3.4 starts_with() . . . 830
15.49 cxx::Weak_ref< T > Class Template Reference . . . 831
15.49.1 Detailed Description . . . 832
15.50 cxx::Weak_ref_base Class Reference . . . 833
15.50.1 Detailed Description . . . 835
15.51 Elf32_Auxv Struct Reference . . . 836
15.51.1 Detailed Description . . . 836
15.51.2 Field Documentation . . . 836
15.51.2.1 atype . . . 836
15.52 Elf32_Dyn Struct Reference . . . 837
15.52.1 Detailed Description . . . 837
15.52.2 Field Documentation . . . 837
15.52.2.1 d_tag . . . 837
15.53 Elf32_Ehdr Struct Reference . . . 838
15.53.1 Detailed Description . . . 839
15.53.2 Field Documentation . . . 839
15.53.2.1 e_flags . . . 839
15.53.2.2 e_machine . . . 839
15.53.2.3 e_type . . . 840
15.53.2.4 e_version . . . 840
15.54 Elf32_Phdr Struct Reference . . . 840
15.54.1 Detailed Description . . . 841
15.54.2 Field Documentation . . . 841
15.54.2.1 p_flags . . . 841
15.54.2.2 p_type . . . 842

15.55 Elf32_Rel Struct Reference . . . 842
15.55.1 Detailed Description . . . 842
15.56 Elf32_Rela Struct Reference . . . 843
15.56.1 Detailed Description . . . 843
15.57 Elf32_Shdr Struct Reference . . . 843
15.57.1 Detailed Description . . . 844
15.57.2 Field Documentation . . . 845
15.57.2.1 sh_flags . . . 845
15.57.2.2 sh_type . . . 845
15.58 Elf32_Sym Struct Reference . . . 845
15.58.1 Detailed Description . . . 846
15.59 Elf64_Auxv Struct Reference . . . 846
15.59.1 Detailed Description . . . 847
15.59.2 Field Documentation . . . 847
15.59.2.1 atype . . . 847
15.60 Elf64_Dyn Struct Reference . . . 847
15.60.1 Detailed Description . . . 848
15.60.2 Field Documentation . . . 848
15.60.2.1 d_tag . . . 848
15.61 Elf64_Ehdr Struct Reference . . . 848
15.61.1 Detailed Description . . . 850
15.61.2 Field Documentation . . . 850
15.61.2.1 e_flags . . . 850
15.61.2.2 e_machine . . . 850
15.61.2.3 e_type . . . 851
15.61.2.4 e_version . . . 851
15.62 Elf64_Phdr Struct Reference . . . 851
15.62.1 Detailed Description . . . 852
15.62.2 Field Documentation . . . 852
15.62.2.1 p_flags . . . 852
15.62.2.2 p_type . . . 853
15.63 Elf64_Rel Struct Reference . . . 853
15.63.1 Detailed Description . . . 853
15.64 Elf64_Rela Struct Reference . . . 854
15.64.1 Detailed Description . . . 854
15.65 Elf64_Shdr Struct Reference . . . 854
15.65.1 Detailed Description . . . 855
15.65.2 Field Documentation . . . 856
15.65.2.1 sh_flags . . . 856
15.65.2.2 sh_type . . . 856
15.66 Elf64_Sym Struct Reference . . . 856
15.66.1 Detailed Description . . . 857

15.67 gfxbitmap_offset Struct Reference . . . 857
15.67.1 Detailed Description . . . 858
15.68 L4::Alloc_list Class Reference . . . 858
15.68.1 Detailed Description . . . 858
15.69 L4::Arm_smccc Class Reference . . . 859
15.69.1 Detailed Description . . . 859
15.69.2 Member Function Documentation . . . 859
15.69.2.1 call() . . . 860
15.70 L4::Base_exception Class Reference . . . 861
15.70.1 Detailed Description . . . 862
15.71 L4::Basic_registry Class Reference . . . 863
15.71.1 Detailed Description . . . 864
15.71.2 Member Function Documentation . . . 864
15.71.2.1 dispatch() . . . 864
15.71.2.2 find() . . . 864
15.72 L4::Bounds_error Class Reference . . . 865
15.72.1 Detailed Description . . . 867
15.73 L4::Cap< T > Class Template Reference . . . 868
15.73.1 Detailed Description . . . 870
15.73.2 Constructor & Destructor Documentation . . . 870
15.73.2.1 Cap() [1/4] . . . 870
15.73.2.2 Cap() [2/4] . . . 871
15.73.2.3 Cap() [3/4] . . . 871
15.73.2.4 Cap() [4/4] . . . 871
15.73.3 Member Function Documentation . . . 872
15.73.3.1 copy() . . . 872
15.73.3.2 move() . . . 872
15.74 L4::Cap_base Class Reference . . . 873
15.74.1 Detailed Description . . . 875
15.74.2 Member Enumeration Documentation . . . 875
15.74.2.1 Cap_type . . . 875
15.74.2.2 No_init_type . . . 876
15.74.3 Constructor & Destructor Documentation . . . 876
15.74.3.1 Cap_base() [1/2] . . . 876
15.74.3.2 Cap_base() [2/2] . . . 876
15.74.4 Member Function Documentation . . . 877
15.74.4.1 cap() . . . 877
15.74.4.2 copy() . . . 878
15.74.4.3 fpage() . . . 878
15.74.4.4 is_valid() . . . 879
15.74.4.5 move() . . . 880
15.74.4.6 snd_base() . . . 881

15.74.4.7 validate() [1/2] . . . 882
15.74.4.8 validate() [2/2] . . . 882
15.75 L4::Com_error Class Reference . . . 883
15.75.1 Detailed Description . . . 885
15.75.2 Constructor & Destructor Documentation . . . 886
15.75.2.1 Com_error() . . . 886
15.76 L4::Debugger Class Reference . . . 886
15.76.1 Detailed Description . . . 889
15.76.2 Member Function Documentation . . . 889
15.76.2.1 get_object_name() . . . 889
15.76.2.2 global_id() . . . 890
15.76.2.3 kobj_to_id() . . . 890
15.76.2.4 query_log_name() . . . 891
15.76.2.5 query_log_typeid() . . . 891
15.76.2.6 set_object_name() . . . 892
15.76.2.7 switch_log() . . . 892
15.77 L4::Element_already_exists Class Reference . . . 893
15.77.1 Detailed Description . . . 894
15.78 L4::Element_not_found Class Reference . . . 895
15.78.1 Detailed Description . . . 896
15.79 L4::Epiface Struct Reference . . . 897
15.79.1 Detailed Description . . . 898
15.79.2 Member Function Documentation . . . 898
15.79.2.1 dispatch() . . . 898
15.79.2.2 get_buffer_demand() . . . 899
15.79.2.3 obj_cap() . . . 899
15.79.2.4 server_iface() . . . 900
15.79.2.5 set_server() . . . 900
15.80 L4::Epiface_t< Derived, IFACE, BASE, bool > Struct Template Reference . . . 901
15.80.1 Detailed Description . . . 903
15.80.2 Member Function Documentation . . . 903
15.80.2.1 dispatch() . . . 903
15.81 L4::Epiface_t0< RPC_IFACE, BASE > Struct Template Reference . . . 904
15.81.1 Detailed Description . . . 905
15.81.2 Member Function Documentation . . . 906
15.81.2.1 obj_cap() . . . 906
15.82 L4::Exception Class Reference . . . 906
15.82.1 Detailed Description . . . 907
15.82.2 Member Typedef Documentation . . . 907
15.82.2.1 Rpcs . . . 907
15.83 L4::Exception_tracer Class Reference . . . 907
15.83.1 Detailed Description . . . 909

15.84 L4::Factory Class Reference . . . 909
15.84.1 Detailed Description . . . 912
15.84.2 Member Function Documentation . . . 912
15.84.2.1 create() [1/2] . . . 912
15.84.2.2 create() [2/2] . . . 913
15.84.2.3 create_factory() . . . 914
15.84.2.4 create_gate() . . . 915
15.84.2.5 create_task() . . . 916
15.85 L4::Factory::Lstr Struct Reference . . . 917
15.85.1 Detailed Description . . . 917
15.85.2 Constructor & Destructor Documentation . . . 918
15.85.2.1 Lstr() . . . 918
15.86 L4::Factory::Nil Struct Reference . . . 919
15.86.1 Detailed Description . . . 919
15.87 L4::Factory::S Class Reference . . . 920
15.87.1 Detailed Description . . . 921
15.87.2 Constructor & Destructor Documentation . . . 921
15.87.2.1 S() [1/2] . . . 921
15.87.2.2 S() [2/2] . . . 921
15.87.3 Member Function Documentation . . . 922
15.87.3.1 operator l4_msgtag_t() . . . 922
15.87.3.2 put() [1/6] . . . 922
15.87.3.3 put() [2/6] . . . 922
15.87.3.4 put() [3/6] . . . 923
15.87.3.5 put() [4/6] . . . 923
15.87.3.6 put() [5/6] . . . 923
15.87.3.7 put() [6/6] . . . 924
15.88 L4::Icu Class Reference . . . 924
15.88.1 Detailed Description . . . 927
15.88.2 Member Function Documentation . . . 927
15.88.2.1 bind() . . . 927
15.88.2.2 info() . . . 929
15.88.2.3 mask() . . . 930
15.88.2.4 msi_info() . . . 930
15.88.2.5 set_mode() . . . 931
15.88.2.6 unbind() . . . 932
15.89 L4::Icu::Info Class Reference . . . 932
15.89.1 Detailed Description . . . 934
15.90 L4::Invalid_capability Class Reference . . . 934
15.90.1 Detailed Description . . . 935
15.90.2 Constructor & Destructor Documentation . . . 936
15.90.2.1 Invalid_capability() . . . 936

15.90.3 Member Function Documentation . . . 936
15.90.3.1 cap() . . . 936
15.91 L4::Io_pager Class Reference . . . 937
15.91.1 Detailed Description . . . 938
15.91.2 Member Function Documentation . . . 939
15.91.2.1 io_page_fault() . . . 939
15.92 L4::Iommu Class Reference . . . 939
15.92.1 Detailed Description . . . 941
15.92.2 Member Function Documentation . . . 942
15.92.2.1 bind() . . . 942
15.92.2.2 unbind() . . . 942
15.93 L4::IOModifier Class Reference . . . 942
15.93.1 Detailed Description . . . 943
15.94 L4::Ipc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference . . . 943
15.94.1 Detailed Description . . . 946
15.95 L4::Ipc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference . . . 946
15.95.1 Detailed Description . . . 947
15.96 L4::Ipc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference . . . 947
15.96.1 Detailed Description . . . 949
15.97 L4::Ipc::As_value< T > Struct Template Reference . . . 949
15.97.1 Detailed Description . . . 950
15.98 L4::Ipc::Buf_item Class Reference . . . 950
15.98.1 Detailed Description . . . 951
15.99 L4::Ipc::Call Struct Reference . . . 951
15.99.1 Detailed Description . . . 952
15.100 L4::Ipc::Call_t< RIGHTS > Struct Template Reference . . . 952
15.100.1 Detailed Description . . . 953
15.101 L4::Ipc::Call_zero_send_timeout Struct Reference . . . 953
15.101.1 Detailed Description . . . 954
15.102 L4::Ipc::Cap< T > Class Template Reference . . . 955
15.102.1 Detailed Description . . . 956
15.102.2 Member Enumeration Documentation . . . 956
15.102.2.1 anonymous enum . . . 956
15.102.3 Constructor & Destructor Documentation . . . 956
15.102.3.1 Cap() . . . 957
15.102.4 Member Function Documentation . . . 957
15.102.4.1 from_ci() . . . 957
15.103 L4::Ipc::Gen_fpage< T > Class Template Reference . . . 957
15.103.1 Detailed Description . . . 959
15.103.2 Member Function Documentation . . . 959
15.103.2.1 cap_received() . . . 959
15.103.2.2 id_received() . . . 959

15.103.2.3 is_compound() . . . 960
15.103.2.4 local_id_received() . . . 960
15.104 L4::Ipc::In_out< T > Struct Template Reference . . . 960
15.104.1 Detailed Description . . . 960
15.105 L4::Ipc::Iostream Class Reference . . . 961
15.105.1 Detailed Description . . . 964
15.105.2 Constructor & Destructor Documentation . . . 965
15.105.2.1 Iostream() . . . 965
15.105.3 Member Function Documentation . . . 965
15.105.3.1 call() . . . 965
15.105.3.2 get() [1/3] . . . 966
15.105.3.3 get() [2/3] . . . 966
15.105.3.4 get() [3/3] . . . 967
15.105.3.5 put() [1/2] . . . 967
15.105.3.6 put() [2/2] . . . 968
15.105.3.7 reply_and_wait() [1/2] . . . 968
15.105.3.8 reply_and_wait() [2/2] . . . 969
15.105.3.9 reset() . . . 969
15.106 L4::Ipc::Istream Class Reference . . . 970
15.106.1 Detailed Description . . . 973
15.106.2 Constructor & Destructor Documentation . . . 973
15.106.2.1 Istream() . . . 974
15.106.3 Member Function Documentation . . . 974
15.106.3.1 get() [1/3] . . . 974
15.106.3.2 get() [2/3] . . . 975
15.106.3.3 get() [3/3] . . . 975
15.106.3.4 receive() . . . 976
15.106.3.5 reset() . . . 976
15.106.3.6 skip() . . . 977
15.106.3.7 tag() [1/2] . . . 977
15.106.3.8 tag() [2/2] . . . 978
15.106.3.9 wait() [1/2] . . . 978
15.106.3.10 wait() [2/2] . . . 979
15.107 L4::Ipc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference . . . 980
15.107.1 Detailed Description . . . 980
15.108 L4::Ipc::Msg::Cls_buffer Struct Reference . . . 982
15.108.1 Detailed Description . . . 983
15.109 L4::Ipc::Msg::Cls_data Struct Reference . . . 983
15.109.1 Detailed Description . . . 984
15.110 L4::Ipc::Msg::Cls_item Struct Reference . . . 984
15.110.1 Detailed Description . . . 985
15.111 L4::Ipc::Msg::Dir_in Struct Reference . . . 985

15.111.1 Detailed Description . . . 985
15.112 L4::Ipc::Msg::Dir_out Struct Reference . . . 986
15.112.1 Detailed Description . . . 986
15.113 L4::Ipc::Msg::Do_in_data Struct Reference . . . 987
15.113.1 Detailed Description . . . 987
15.114 L4::Ipc::Msg::Do_in_items Struct Reference . . . 988
15.114.1 Detailed Description . . . 988
15.115 L4::Ipc::Msg::Do_out_data Struct Reference . . . 989
15.115.1 Detailed Description . . . 990
15.116 L4::Ipc::Msg::Do_out_items Struct Reference . . . 990
15.116.1 Detailed Description . . . 991
15.117 L4::Ipc::Msg::Do_rcv_buffers Struct Reference . . . 991
15.117.1 Detailed Description . . . 992
15.118 L4::Ipc::Msg::Elem< Array< A, LEN > & > Struct Template Reference . . . 993
15.118.1 Detailed Description . . . 993
15.119 L4::Ipc::Msg::Elem< Array< A, LEN > > Struct Template Reference . . . 994
15.119.1 Detailed Description . . . 994
15.120 L4::Ipc::Msg::Elem< Array_ref< A, LEN > & > Struct Template Reference . . . 995
15.120.1 Detailed Description . . . 995
15.121 L4::Ipc::Msg::Is_valid_rpc_type< T > Struct Template Reference . . . 996
15.121.1 Detailed Description . . . 997
15.122 L4::Ipc::Msg::Svr_arg_pack< IPC_TYPE > Struct Template Reference . . . 998
15.122.1 Detailed Description . . . 998
15.123 L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference . . . 998
15.123.1 Detailed Description . . . 999
15.124 L4::Ipc::Msg_ptr< T > Class Template Reference . . . 1000
15.124.1 Detailed Description . . . 1000
15.124.2 Constructor & Destructor Documentation . . . 1000
15.124.2.1 Msg_ptr() . . . 1000
15.125 L4::Ipc::Opt< T > Struct Template Reference . . . 1001
15.125.1 Detailed Description . . . 1002
15.126 L4::Ipc::Ostream Class Reference . . . 1002
15.126.1 Detailed Description . . . 1005
15.126.2 Member Function Documentation . . . 1005
15.126.2.1 put() [1/2] . . . 1005
15.126.2.2 put() [2/2] . . . 1006
15.126.2.3 send() . . . 1006
15.126.2.4 tag() [1/2] . . . 1007
15.126.2.5 tag() [2/2] . . . 1008
15.127 L4::Ipc::Out< T > Struct Template Reference . . . 1008
15.127.1 Detailed Description . . . 1008
15.128 L4::Ipc::Ret_array< T > Struct Template Reference . . . 1009

15.128.1 Detailed Description . . . 1009
15.129 L4::Ipc::Send_only Struct Reference . . . 1010
15.129.1 Detailed Description . . . 1010
15.130 L4::Ipc::Small_buf Class Reference . . . 1011
15.130.1 Detailed Description . . . 1011
15.130.2 Constructor & Destructor Documentation . . . 1011
15.130.2.1 Small_buf() [1/2] . . . 1011
15.130.2.2 Small_buf() [2/2] . . . 1012
15.131 L4::Ipc::Snd_item Class Reference . . . 1012
15.131.1 Detailed Description . . . 1013
15.132 L4::Ipc::Str_cp_in< T > Class Template Reference . . . 1013
15.132.1 Detailed Description . . . 1013
15.132.2 Constructor & Destructor Documentation . . . 1014
15.132.2.1 Str_cp_in() . . . 1014
15.133 L4::Ipc::Varg Class Reference . . . 1014
15.133.1 Detailed Description . . . 1015
15.133.2 Member Function Documentation . . . 1016
15.133.2.1 data() . . . 1016
15.133.2.2 get_value() . . . 1016
15.133.2.3 is_nil() . . . 1017
15.133.2.4 is_of() . . . 1017
15.133.2.5 is_of_int() . . . 1018
15.133.2.6 length() . . . 1019
15.133.2.7 tag() . . . 1019
15.133.2.8 type() . . . 1019
15.133.2.9 value() . . . 1019
15.134 L4::Ipc::Varg_list< MAX > Class Template Reference . . . 1020
15.134.1 Detailed Description . . . 1021
15.135 L4::Ipc::Varg_list_ref Class Reference . . . 1022
15.135.1 Detailed Description . . . 1023
15.135.2 Constructor & Destructor Documentation . . . 1023
15.135.2.1 Varg_list_ref() . . . 1023
15.136 L4::Ipc::Varg_list_ref::Iterator Class Reference . . . 1024
15.136.1 Detailed Description . . . 1024
15.137 L4::Ipc_gate Class Reference . . . 1025
15.137.1 Detailed Description . . . 1027
15.137.2 Member Function Documentation . . . 1027
15.137.2.1 get_infos() . . . 1027
15.138 L4::Ipc_svr::Br_manager_no_buffers Class Reference . . . 1028
15.138.1 Detailed Description . . . 1031
15.138.2 Member Function Documentation . . . 1031
15.138.2.1 alloc_buffer_demand() . . . 1031

15.139 L4::Ipc_svr::Compound_reply Struct Reference . . . 1032
15.139.1 Detailed Description . . . 1033
15.140 L4::Ipc_svr::Default_loop_hooks Struct Reference . . . 1033
15.140.1 Detailed Description . . . 1035
15.141 L4::Ipc_svr::Default_setup_wait Struct Reference . . . 1035
15.141.1 Detailed Description . . . 1036
15.142 L4::Ipc_svr::Default_timeout Struct Reference . . . 1036
15.142.1 Detailed Description . . . 1038
15.143 L4::Ipc_svr::Direct_dispatch< R > Struct Template Reference . . . 1038
15.143.1 Detailed Description . . . 1039
15.144 L4::Ipc_svr::Direct_dispatch< R * > Struct Template Reference . . . 1040
15.144.1 Detailed Description . . . 1040
15.145 L4::Ipc_svr::Exc_dispatch< R, Exc > Struct Template Reference . . . 1041
15.145.1 Detailed Description . . . 1042
15.146 L4::Ipc_svr::Ignore_errors Struct Reference . . . 1042
15.146.1 Detailed Description . . . 1044
15.147 L4::Ipc_svr::Server_iface Class Reference . . . 1044
15.147.1 Detailed Description . . . 1047
15.147.2 Member Function Documentation . . . 1047
15.147.2.1 add_timeout() . . . 1047
15.147.2.2 alloc_buffer_demand() . . . 1047
15.147.2.3 get_rcv_cap() . . . 1048
15.147.2.4 rcv_cap() [1/2] . . . 1049
15.147.2.5 rcv_cap() [2/2] . . . 1049
15.147.2.6 realloc_rcv_cap() . . . 1050
15.147.2.7 remove_timeout() . . . 1051
15.148 L4::Ipc_svr::Timeout Class Reference . . . 1051
15.148.1 Detailed Description . . . 1053
15.148.2 Member Function Documentation . . . 1053
15.148.2.1 expired() . . . 1053
15.148.2.2 timeout() . . . 1054
15.149 L4::Ipc_svr::Timeout_queue Class Reference . . . 1054
15.149.1 Detailed Description . . . 1055
15.149.2 Member Function Documentation . . . 1055
15.149.2.1 add() . . . 1055
15.149.2.2 handle_expired_timeouts() . . . 1056
15.149.2.3 next_timeout() . . . 1057
15.149.2.4 remove() . . . 1058
15.149.2.5 timeout_expired() . . . 1059
15.150 L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference . . . 1060
15.150.1 Detailed Description . . . 1063
15.150.2 Member Function Documentation . . . 1063

15.150.2.1 add_timeout() . . . 1063
15.150.2.2 remove_timeout() . . . 1064
15.151 L4::Irq Class Reference . . . 1065
15.151.1 Detailed Description . . . 1068
15.151.2 Member Function Documentation . . . 1068
15.151.2.1 detach() . . . 1068
15.151.2.2 receive() . . . 1069
15.151.2.3 unmask() . . . 1070
15.151.2.4 wait() . . . 1071
15.152 L4::Irq_eoi Class Reference . . . 1071
15.152.1 Detailed Description . . . 1072
15.152.2 Member Function Documentation . . . 1073
15.152.2.1 unmask() . . . 1073
15.153 L4::Irq_handler_object Struct Reference . . . 1074
15.153.1 Detailed Description . . . 1075
15.154 L4::Irq_mux Struct Reference . . . 1076
15.154.1 Detailed Description . . . 1079
15.154.2 Member Function Documentation . . . 1079
15.154.2.1 chain() . . . 1079
15.155 L4::Irqep_t< Derived, BASE, bool > Struct Template Reference . . . 1080
15.155.1 Detailed Description . . . 1083
15.155.2 Member Function Documentation . . . 1083
15.155.2.1 dispatch() . . . 1083
15.155.2.2 obj_cap() . . . 1084
15.156 L4::Kip::Mem_desc Class Reference . . . 1084
15.156.1 Detailed Description . . . 1086
15.156.2 Member Enumeration Documentation . . . 1086
15.156.2.1 Info_sub_type . . . 1086
15.156.2.2 Mem_type . . . 1086
15.156.3 Constructor & Destructor Documentation . . . 1087
15.156.3.1 Mem_desc() . . . 1087
15.156.4 Member Function Documentation . . . 1087
15.156.4.1 all() [1/2] . . . 1087
15.156.4.2 all() [2/2] . . . 1088
15.156.4.3 count() [1/2] . . . 1089
15.156.4.4 count() [2/2] . . . 1090
15.156.4.5 end() . . . 1090
15.156.4.6 first() . . . 1091
15.156.4.7 is_virtual() . . . 1092
15.156.4.8 set() . . . 1092
15.156.4.9 size() . . . 1093
15.156.4.10 start() . . . 1093

15.156.4.11 sub_type() . . . 1094
15.156.4.12 type() . . . 1094
15.157 L4::Kobject Class Reference . . . 1094
15.157.1 Detailed Description . . . 1095
15.157.2 Member Function Documentation . . . 1095
15.157.2.1 cap() . . . 1095
15.157.2.2 dec_refcnt() . . . 1096
15.158 L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference . . . 1097
15.158.1 Detailed Description . . . 1098
15.158.2 Member Typedef Documentation . . . 1099
15.158.2.1 __Iface . . . 1099
15.158.2.2 __Iface_list . . . 1099
15.158.2.3 Class . . . 1100
15.158.3 Member Function Documentation . . . 1100
15.158.3.1 __check_protocols__() . . . 1100
15.158.3.2 c() . . . 1100
15.159 L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference . . . 1101
15.159.1 Detailed Description . . . 1102
15.159.2 Member Typedef Documentation . . . 1103
15.159.2.1 __Iface . . . 1103
15.159.2.2 __Iface_list . . . 1103
15.159.2.3 Class . . . 1104
15.159.3 Member Function Documentation . . . 1104
15.159.3.1 __check_protocols__() . . . 1104
15.159.3.2 c() . . . 1104
15.160 L4::Kobject_demand< T > Struct Template Reference . . . 1105
15.160.1 Detailed Description . . . 1105
15.161 L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference . . . 1105
15.161.1 Detailed Description . . . 1107
15.162 L4::Kobject_typeid< T > Struct Template Reference . . . 1107
15.162.1 Detailed Description . . . 1108
15.162.2 Member Typedef Documentation . . . 1108
15.162.2.1 Demand . . . 1108
15.162.3 Member Function Documentation . . . 1108
15.162.3.1 demand() . . . 1109
15.162.3.2 id() . . . 1109
15.162.3.3 proto_dispatch() . . . 1109
15.163 L4::Kobject_typeid< void > Struct Reference . . . 1110
15.163.1 Detailed Description . . . 1111
15.164 L4::Kobject_x< Derived, ARGS > Struct Template Reference . . . 1111
15.164.1 Detailed Description . . . 1112
15.165 L4::Meta Class Reference . . . 1112

15.165.1 Detailed Description . . . 1114
15.165.2 Member Function Documentation . . . 1115
15.165.2.1 interface() . . . 1115
15.165.2.2 num_interfaces() . . . 1115
15.165.2.3 supports() . . . 1115
15.166 L4::Out_of_memory Class Reference . . . 1116
15.166.1 Detailed Description . . . 1119
15.167 L4::Pager Class Reference . . . 1119
15.167.1 Detailed Description . . . 1121
15.167.2 Member Function Documentation . . . 1122
15.167.2.1 page_fault() . . . 1122
15.168 L4::Platform_control Class Reference . . . 1123
15.168.1 Detailed Description . . . 1124
15.168.2 Member Function Documentation . . . 1125
15.168.2.1 cpu_allow_shutdown() . . . 1125
15.168.2.2 cpu_disable() . . . 1125
15.168.2.3 cpu_enable() . . . 1126
15.168.2.4 system_shutdown() . . . 1126
15.168.2.5 system_suspend() . . . 1126
15.169 L4::Poll_timeout_kipclock Class Reference . . . 1127
15.169.1 Detailed Description . . . 1127
15.169.2 Constructor & Destructor Documentation . . . 1127
15.169.2.1 Poll_timeout_kipclock() . . . 1128
15.169.3 Member Function Documentation . . . 1128
15.169.3.1 set() . . . 1128
15.169.3.2 test() . . . 1129
15.169.3.3 timed_out() . . . 1130
15.170 L4::Proto_t< P > Struct Template Reference . . . 1130
15.170.1 Detailed Description . . . 1131
15.171 L4::Rcv_endpoint Class Reference . . . 1131
15.171.1 Detailed Description . . . 1133
15.171.2 Member Function Documentation . . . 1134
15.171.2.1 bind_thread() . . . 1134
15.172 L4::Registry_iface Class Reference . . . 1135
15.172.1 Detailed Description . . . 1136
15.172.2 Member Function Documentation . . . 1136
15.172.2.1 register_irq_obj() . . . 1136
15.172.2.2 register_obj() [1/3] . . . 1137
15.172.2.3 register_obj() [2/3] . . . 1137
15.172.2.4 register_obj() [3/3] . . . 1138
15.172.2.5 unregister_obj() . . . 1138
15.173 L4::Runtime_error Class Reference . . . 1139

15.173.1 Detailed Description . . . 1140
15.173.2 Constructor & Destructor Documentation . . . 1141
15.173.2.1 Runtime_error() . . . 1141
15.173.3 Member Function Documentation . . . 1141
15.173.3.1 err_no() . . . 1141
15.173.3.2 extra_str() . . . 1141
15.174 L4::Scheduler Class Reference . . . 1142
15.174.1 Detailed Description . . . 1144
15.174.2 Member Function Documentation . . . 1144
15.174.2.1 idle_time() . . . 1144
15.174.2.2 info() . . . 1145
15.174.2.3 is_online() . . . 1146
15.174.2.4 run_thread() . . . 1146
15.175 L4::Semaphore Struct Reference . . . 1147
15.175.1 Detailed Description . . . 1150
15.175.2 Member Function Documentation . . . 1150
15.175.2.1 down() . . . 1150
15.175.2.2 up() . . . 1151
15.176 L4::Server< LOOP_HOOKS > Class Template Reference . . . 1151
15.176.1 Detailed Description . . . 1154
15.176.2 Constructor & Destructor Documentation . . . 1154
15.176.2.1 Server() . . . 1154
15.176.3 Member Function Documentation . . . 1154
15.176.3.1 internal_loop() . . . 1155
15.176.3.2 loop() . . . 1156
15.177 L4::Server_object Class Reference . . . 1156
15.177.1 Detailed Description . . . 1158
15.177.2 Member Function Documentation . . . 1159
15.177.2.1 dispatch() [1/2] . . . 1159
15.177.2.2 dispatch() [2/2] . . . 1159
15.178 L4::Server_object_t< IFACE, BASE > Struct Template Reference . . . 1161
15.178.1 Detailed Description . . . 1163
15.178.2 Member Function Documentation . . . 1163
15.178.2.1 dispatch_meta_request() . . . 1163
15.178.2.2 get_buffer_demand() . . . 1164
15.178.2.3 proto_dispatch() . . . 1164
15.179 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference . . . 1165
15.179.1 Detailed Description . . . 1168
15.180 L4::Smart_cap< T, SMART > Class Template Reference . . . 1168
15.180.1 Detailed Description . . . 1171
15.180.2 Constructor & Destructor Documentation . . . 1171
15.180.2.1 Smart_cap() . . . 1171

15.181 L4::String Class Reference . . . 1171
15.181.1 Detailed Description . . . 1172
15.182 L4::Task Class Reference . . . 1172
15.182.1 Detailed Description . . . 1175
15.182.2 Member Function Documentation . . . 1175
15.182.2.1 add_ku_mem() . . . 1175
15.182.2.2 cap_equal() . . . 1176
15.182.2.3 cap_valid() . . . 1176
15.182.2.4 delete_obj() . . . 1177
15.182.2.5 map() . . . 1177
15.182.2.6 release_cap() . . . 1178
15.182.2.7 unmap() . . . 1179
15.182.2.8 unmap_batch() . . . 1179
15.183 L4::Thread Class Reference . . . 1180
15.183.1 Detailed Description . . . 1183
15.183.2 Member Function Documentation . . . 1184
15.183.2.1 control() . . . 1184
15.183.2.2 ex_regs() [1/2] . . . 1184
15.183.2.3 ex_regs() [2/2] . . . 1185
15.183.2.4 modify_senders() . . . 1186
15.183.2.5 register_del_irq() . . . 1187
15.183.2.6 stats_time() . . . 1188
15.183.2.7 switch_to() . . . 1188
15.183.2.8 vcpu_control() . . . 1189
15.183.2.9 vcpu_control_ext() . . . 1190
15.183.2.10 vcpu_resume_commit() . . . 1191
15.183.2.11 vcpu_resume_start() . . . 1191
15.184 L4::Thread::Attr Class Reference . . . 1191
15.184.1 Detailed Description . . . 1192
15.184.2 Constructor & Destructor Documentation . . . 1192
15.184.2.1 Attr() . . . 1192
15.184.3 Member Function Documentation . . . 1193
15.184.3.1 alien() . . . 1193
15.184.3.2 bind() . . . 1193
15.184.3.3 exc_handler() [1/2] . . . 1194
15.184.3.4 exc_handler() [2/2] . . . 1194
15.184.3.5 pager() [1/2] . . . 1195
15.184.3.6 pager() [2/2] . . . 1195
15.184.3.7 ux_host_syscall() . . . 1195
15.185 L4::Thread::Modify_senders Class Reference . . . 1196
15.185.1 Detailed Description . . . 1196
15.185.2 Member Function Documentation . . . 1196

15.185.2.1 add() . . . 1197
15.186 L4::Triggerable Struct Reference . . . 1198
15.186.1 Detailed Description . . . 1199
15.186.2 Member Function Documentation . . . 1200
15.186.2.1 trigger() . . . 1200
15.187 L4::Type_info Struct Reference . . . 1201
15.187.1 Detailed Description . . . 1201
15.188 L4::Type_info::Demand Class Reference . . . 1202
15.188.1 Detailed Description . . . 1203
15.188.2 Constructor & Destructor Documentation . . . 1203
15.188.2.1 Demand() . . . 1204
15.188.3 Member Function Documentation . . . 1204
15.188.3.1 no_demand() . . . 1204
15.189 L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference . . . 1205
15.189.1 Detailed Description . . . 1206
15.189.2 Member Enumeration Documentation . . . 1207
15.189.2.1 anonymous enum . . . 1207
15.190 L4::Type_info::Demand_union_t< D1, D2 > Struct Template Reference . . . 1207
15.190.1 Detailed Description . . . 1209
15.191 L4::Typeid::Detail::_Rpcs< OPCODE, O, X > Struct Template Reference . . . 1210
15.191.1 Detailed Description . . . 1211
15.192 L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference . . . 1211
15.192.1 Detailed Description . . . 1212
15.193 L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... > Struct Template Reference . . . 1212
15.193.1 Detailed Description . . . 1213
15.194 L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference . . . 1213
15.194.1 Detailed Description . . . 1214
15.195 L4::Typeid::Detail::Rpcs_end Struct Reference . . . 1214
15.195.1 Detailed Description . . . 1215
15.196 L4::Typeid::P_dispatch< LIST > Struct Template Reference . . . 1215
15.196.1 Detailed Description . . . 1215
15.197 L4::Typeid::Raw_ipc< CLASS > Struct Template Reference . . . 1216
15.197.1 Detailed Description . . . 1216
15.198 L4::Typeid::Rpc_nocode< OPERATION > Struct Template Reference . . . 1216
15.198.1 Detailed Description . . . 1218
15.199 L4::Typeid::Rpcs< RPCS > Struct Template Reference . . . 1219
15.199.1 Detailed Description . . . 1220
15.200 L4::Typeid::Rpcs_code< OPCODE_TYPE > Struct Template Reference . . . 1221
15.200.1 Detailed Description . . . 1221
15.201 L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference . . . 1222
15.201.1 Detailed Description . . . 1223
15.202 L4::Typeid::Rpcs_sys< ARG > Struct Template Reference . . . 1224

15.202.1 Detailed Description . . . 1225
15.203 L4::Types::Bool< V > Struct Template Reference . . . 1226
15.203.1 Detailed Description . . . 1227
15.204 L4::Types::False Struct Reference . . . 1228
15.204.1 Detailed Description . . . 1229
15.205 L4::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference . . . 1229
15.205.1 Detailed Description . . . 1231
15.205.2 Member Enumeration Documentation . . . 1232
15.205.2.1 None_type . . . 1232
15.205.3 Constructor & Destructor Documentation . . . 1232
15.205.3.1 Flags() [1/2] . . . 1232
15.205.3.2 Flags() [2/2] . . . 1233
15.205.4 Member Function Documentation . . . 1233
15.205.4.1 clear() . . . 1233
15.205.4.2 from_raw() . . . 1234
15.206 L4::Types::Flags_ops_t< DT > Struct Template Reference . . . 1234
15.206.1 Detailed Description . . . 1235
15.207 L4::Types::Flags_t< DT, T > Struct Template Reference . . . 1236
15.207.1 Detailed Description . . . 1238
15.208 L4::Types::Int_for_size< SIZE, bool > Struct Template Reference . . . 1238
15.208.1 Detailed Description . . . 1238
15.209 L4::Types::Int_for_type< T > Struct Template Reference . . . 1239
15.209.1 Detailed Description . . . 1239
15.210 L4::Types::Same< A, B > Struct Template Reference . . . 1240
15.210.1 Detailed Description . . . 1241
15.211 L4::Types::True Struct Reference . . . 1242
15.211.1 Detailed Description . . . 1244
15.212 L4::Unknown_error Class Reference . . . 1244
15.212.1 Detailed Description . . . 1246
15.213 L4::Vcon Class Reference . . . 1247
15.213.1 Detailed Description . . . 1249
15.213.2 Member Function Documentation . . . 1249
15.213.2.1 get_attr() . . . 1249
15.213.2.2 read() . . . 1250
15.213.2.3 read_with_flags() . . . 1251
15.213.2.4 send() . . . 1252
15.213.2.5 set_attr() . . . 1253
15.213.2.6 write() . . . 1253
15.214 L4::Vm Class Reference . . . 1254
15.214.1 Detailed Description . . . 1256
15.215 l4_buf_regs_t Struct Reference . . . 1257
15.215.1 Detailed Description . . . 1257

15.216 l4_exc_regs_t Struct Reference . . . 1258
15.216.1 Detailed Description . . . 1260
15.216.2 Field Documentation . . . 1260
15.216.2.1 flags . . . 1260
15.216.2.2 ss . . . 1260
15.217 l4_fpage_t Union Reference . . . 1261
15.217.1 Detailed Description . . . 1261
15.218 l4_icu_info_t Struct Reference . . . 1261
15.218.1 Detailed Description . . . 1263
15.218.2 Field Documentation . . . 1263
15.218.2.1 features . . . 1263
15.219 l4_icu_msi_info_t Struct Reference . . . 1263
15.219.1 Detailed Description . . . 1264
15.220 l4_kernel_info_mem_desc_t Struct Reference . . . 1264
15.220.1 Detailed Description . . . 1264
15.221 l4_kernel_info_t Struct Reference . . . 1265
15.221.1 Detailed Description . . . 1267
15.222 l4_msg_regs_t Union Reference . . . 1267
15.222.1 Detailed Description . . . 1267
15.223 l4_msgtag_t Struct Reference . . . 1268
15.223.1 Detailed Description . . . 1269
15.223.2 Member Function Documentation . . . 1269
15.223.2.1 flags() . . . 1269
15.224 l4_sched_cpu_set_t Struct Reference . . . 1270
15.224.1 Detailed Description . . . 1270
15.224.2 Member Function Documentation . . . 1270
15.224.2.1 granularity() . . . 1271
15.224.2.2 offset() . . . 1271
15.224.2.3 set() . . . 1272
15.224.3 Field Documentation . . . 1272
15.224.3.1 gran_offset . . . 1272
15.225 l4_sched_param_t Struct Reference . . . 1273
15.225.1 Detailed Description . . . 1274
15.226 l4_snd_fpage_t Struct Reference . . . 1274
15.226.1 Detailed Description . . . 1275
15.227 l4_thread_regs_t Struct Reference . . . 1275
15.227.1 Detailed Description . . . 1276
15.228 l4_timeout_s Struct Reference . . . 1276
15.228.1 Detailed Description . . . 1277
15.229 l4_timeout_t Union Reference . . . 1277
15.229.1 Detailed Description . . . 1278
15.230 l4_vcon_attr_t Struct Reference . . . 1278

15.230.1 Detailed Description . . . 1279
15.230.2 Member Function Documentation . . . 1279
15.230.2.1 set_raw() . . . 1279
15.231 l4_vcpu_ipc_regs_t Struct Reference . . . 1280
15.231.1 Detailed Description . . . 1280
15.232 l4_vcpu_regs_t Struct Reference . . . 1281
15.232.1 Detailed Description . . . 1282
15.232.2 Field Documentation . . . 1282
15.232.2.1 ax . . . 1283
15.232.2.2 bp . . . 1283
15.232.2.3 bx . . . 1283
15.232.2.4 cx . . . 1283
15.232.2.5 di . . . 1284
15.232.2.6 dx . . . 1284
15.232.2.7 si . . . 1284
15.233 l4_vcpu_state_t Struct Reference . . . 1285
15.233.1 Detailed Description . . . 1287
15.234 l4_vhw_descriptor Struct Reference . . . 1287
15.234.1 Detailed Description . . . 1288
15.235 l4_vhw_entry Struct Reference . . . 1289
15.235.1 Detailed Description . . . 1290
15.236 l4_vm_svm_vmcb_control_area Struct Reference . . . 1290
15.236.1 Detailed Description . . . 1290
15.237 l4_vm_svm_vmcb_state_save_area Struct Reference . . . 1291
15.237.1 Detailed Description . . . 1291
15.238 l4_vm_svm_vmcb_state_save_area_seg Struct Reference . . . 1292
15.238.1 Detailed Description . . . 1292
15.239 l4_vm_svm_vmcb_t Struct Reference . . . 1293
15.239.1 Detailed Description . . . 1294
15.240 l4_vm_tz_state Struct Reference . . . 1294
15.240.1 Detailed Description . . . 1294
15.241 L4Re::Cap_alloc Class Reference . . . 1295
15.241.1 Detailed Description . . . 1295
15.241.2 Member Function Documentation . . . 1296
15.241.2.1 alloc() [1/2] . . . 1296
15.241.2.2 alloc() [2/2] . . . 1296
15.241.2.3 free() . . . 1297
15.241.2.4 get_cap_alloc() . . . 1298
15.242 L4Re::Console Class Reference . . . 1299
15.242.1 Detailed Description . . . 1300
15.243 L4Re::Dataspace Class Reference . . . 1301
15.243.1 Detailed Description . . . 1303

15.243.2 Member Function Documentation . . . 1303
15.243.2.1 allocate() . . . 1303
15.243.2.2 clear() . . . 1304
15.243.2.3 copy_in() . . . 1304
15.243.2.4 flags() . . . 1306
15.243.2.5 info() . . . 1307
15.243.2.6 map() . . . 1307
15.243.2.7 map_region() . . . 1308
15.243.2.8 size() . . . 1309
15.244 L4Re::Dataspace::F Struct Reference . . . 1309
15.244.1 Detailed Description . . . 1310
15.244.2 Member Enumeration Documentation . . . 1310
15.244.2.1 anonymous enum . . . 1310
15.244.2.2 Flags . . . 1310
15.245 L4Re::Dataspace::Stats Struct Reference . . . 1311
15.245.1 Detailed Description . . . 1311
15.246 L4Re::Debug_obj Class Reference . . . 1311
15.246.1 Detailed Description . . . 1313
15.246.2 Member Function Documentation . . . 1314
15.246.2.1 debug() . . . 1314
15.247 L4Re::Dma_space Class Reference . . . 1314
15.247.1 Detailed Description . . . 1315
15.247.2 Member Typedef Documentation . . . 1315
15.247.2.1 Attributes . . . 1316
15.247.3 Member Enumeration Documentation . . . 1316
15.247.3.1 Attribute . . . 1316
15.247.3.2 Direction . . . 1316
15.247.3.3 Space_attrib . . . 1317
15.247.4 Member Function Documentation . . . 1317
15.247.4.1 associate() . . . 1317
15.247.4.2 disassociate() . . . 1318
15.247.4.3 map() . . . 1318
15.247.4.4 unmap() . . . 1319
15.248 L4Re::Env Class Reference . . . 1319
15.248.1 Detailed Description . . . 1321
15.248.2 Member Function Documentation . . . 1322
15.248.2.1 env() . . . 1322
15.248.2.2 factory() [1/2] . . . 1323
15.248.2.3 factory() [2/2] . . . 1323
15.248.2.4 first_free_cap() [1/2] . . . 1323
15.248.2.5 first_free_cap() [2/2] . . . 1323
15.248.2.6 first_free_utcb() [1/2] . . . 1324

15.248.2.7 first_free_utcb() [2/2] . . . 1324
15.248.2.8 get() . . . 1324
15.248.2.9 get_cap() [1/2] . . . 1325
15.248.2.10 get_cap() [2/2] . . . 1326
15.248.2.11 initial_caps() [1/2] . . . 1326
15.248.2.12 initial_caps() [2/2] . . . 1326
15.248.2.13 log() [1/2] . . . 1327
15.248.2.14 log() [2/2] . . . 1327
15.248.2.15 main_thread() [1/2] . . . 1327
15.248.2.16 main_thread() [2/2] . . . 1328
15.248.2.17 mem_alloc() [1/2] . . . 1328
15.248.2.18 mem_alloc() [2/2] . . . 1328
15.248.2.19 parent() [1/2] . . . 1329
15.248.2.20 parent() [2/2] . . . 1329
15.248.2.21 rm() [1/2] . . . 1329
15.248.2.22 rm() [2/2] . . . 1330
15.248.2.23 scheduler() [1/2] . . . 1330
15.248.2.24 scheduler() [2/2] . . . 1330
15.248.2.25 task() . . . 1331
15.248.2.26 utcb_area() [1/2] . . . 1331
15.248.2.27 utcb_area() [2/2] . . . 1331
15.249 L4Re::Event Class Reference . . . 1332
15.249.1 Detailed Description . . . 1335
15.249.2 Member Function Documentation . . . 1335
15.249.2.1 get_buffer() . . . 1335
15.250 L4Re::Event_buffer_t< PAYLOAD > Class Template Reference . . . 1335
15.250.1 Detailed Description . . . 1337
15.250.2 Constructor & Destructor Documentation . . . 1337
15.250.2.1 Event_buffer_t() . . . 1337
15.250.3 Member Function Documentation . . . 1337
15.250.3.1 next() . . . 1337
15.250.3.2 put() . . . 1338
15.251 L4Re::Event_buffer_t< PAYLOAD >::Event Struct Reference . . . 1338
15.251.1 Detailed Description . . . 1339
15.252 L4Re::Inhibitor Class Reference . . . 1339
15.252.1 Detailed Description . . . 1342
15.252.2 Member Enumeration Documentation . . . 1342
15.252.2.1 anonymous enum . . . 1342
15.252.3 Member Function Documentation . . . 1342
15.252.3.1 acquire() . . . 1342
15.252.3.2 next_lock_info() . . . 1343
15.252.3.3 release() . . . 1344

15.253 L4Re::Log Class Reference . . . 1344
15.253.1 Detailed Description . . . 1347
15.253.2 Member Function Documentation . . . 1347
15.253.2.1 print() . . . 1347
15.253.2.2 printn() . . . 1347
15.254 L4Re::Mem_alloc Class Reference . . . 1347
15.254.1 Detailed Description . . . 1350
15.254.2 Member Enumeration Documentation . . . 1350
15.254.2.1 Mem_alloc_flags . . . 1350
15.254.3 Member Function Documentation . . . 1350
15.254.3.1 alloc() . . . 1351
15.255 L4Re::Mmio_space Struct Reference . . . 1352
15.255.1 Detailed Description . . . 1354
15.255.2 Member Enumeration Documentation . . . 1354
15.255.2.1 Access_width . . . 1354
15.255.3 Member Function Documentation . . . 1354
15.255.3.1 mmio_read() . . . 1355
15.255.3.2 mmio_write() . . . 1355
15.256 L4Re::Namespace Class Reference . . . 1356
15.256.1 Detailed Description . . . 1358
15.256.2 Member Enumeration Documentation . . . 1358
15.256.2.1 Query_result_flags . . . 1358
15.256.2.2 Register_flags . . . 1358
15.256.3 Member Function Documentation . . . 1359
15.256.3.1 query() [1/2] . . . 1359
15.256.3.2 query() [2/2] . . . 1360
15.256.3.3 register_obj() . . . 1361
15.256.3.4 unlink() . . . 1362
15.257 L4Re::Ned::Cmd_control Class Reference . . . 1362
15.257.1 Detailed Description . . . 1363
15.257.2 Member Function Documentation . . . 1363
15.257.2.1 execute() [1/2] . . . 1363
15.257.2.2 execute() [2/2] . . . 1363
15.258 L4Re::Parent Class Reference . . . 1364
15.258.1 Detailed Description . . . 1366
15.258.2 Member Function Documentation . . . 1367
15.258.2.1 signal() . . . 1367
15.259 L4Re::Random Struct Reference . . . 1367
15.259.1 Detailed Description . . . 1370
15.259.2 Member Function Documentation . . . 1370
15.259.2.1 get_random() . . . 1370
15.260 L4Re::Rm Class Reference . . . 1371

15.260.1 Detailed Description . . . 1374
15.260.2 Member Typedef Documentation . . . 1374
15.260.2.1 Area . . . 1374
15.260.2.2 Region . . . 1374
15.260.3 Member Enumeration Documentation . . . 1374
15.260.3.1 Detach_flags . . . 1374
15.260.3.2 Detach_result . . . 1375
15.260.3.3 Region_flag_shifts . . . 1375
15.260.4 Member Function Documentation . . . 1375
15.260.4.1 attach() [1/2] . . . 1375
15.260.4.2 attach() [2/2] . . . 1377
15.260.4.3 detach() [1/3] . . . 1378
15.260.4.4 detach() [2/3] . . . 1378
15.260.4.5 detach() [3/3] . . . 1379
15.260.4.6 find() . . . 1379
15.260.4.7 free_area() . . . 1380
15.260.4.8 get_areas() . . . 1381
15.260.4.9 get_regions() . . . 1381
15.260.4.10 reserve_area() [1/2] . . . 1382
15.260.4.11 reserve_area() [2/2] . . . 1383
15.261 L4Re::Rm::F Struct Reference . . . 1384
15.261.1 Detailed Description . . . 1384
15.261.2 Member Enumeration Documentation . . . 1384
15.261.2.1 Attach_flags . . . 1384
15.261.2.2 Region_flags . . . 1385
15.262 L4Re::Rm::Range Struct Reference . . . 1385
15.262.1 Detailed Description . . . 1386
15.263 L4Re::Smart_cap_auto< Unmap_flags > Class Template Reference . . . 1386
15.263.1 Detailed Description . . . 1387
15.264 L4Re::Smart_count_cap< Unmap_flags > Class Template Reference . . . 1387
15.264.1 Detailed Description . . . 1388
15.265 L4Re::Util::Br_manager Class Reference . . . 1388
15.265.1 Detailed Description . . . 1391
15.265.2 Member Function Documentation . . . 1391
15.265.2.1 alloc_buffer_demand() . . . 1391
15.265.2.2 get_rcv_cap() . . . 1392
15.265.2.3 realloc_rcv_cap() . . . 1392
15.265.2.4 set_rcv_cap_flags() . . . 1393
15.266 L4Re::Util::Br_manager_hooks Struct Reference . . . 1394
15.266.1 Detailed Description . . . 1395
15.267 L4Re::Util::Br_manager_timeout_hooks Struct Reference . . . 1395
15.267.1 Detailed Description . . . 1397

15.268 L4Re::Util::Cap_alloc_base Class Reference . . . 1398
15.268.1 Detailed Description . . . 1398
15.269 L4Re::Util::Counter< COUNTER > Struct Template Reference . . . 1399
15.269.1 Detailed Description . . . 1400
15.270 L4Re::Util::Counting_cap_alloc< COUNTERTYPE > Class Template Reference . . . 1400
15.270.1 Detailed Description . . . 1402
15.270.2 Constructor & Destructor Documentation . . . 1402
15.270.2.1 Counting_cap_alloc() . . . 1402
15.270.3 Member Function Documentation . . . 1402
15.270.3.1 alloc() [1/2] . . . 1403
15.270.3.2 alloc() [2/2] . . . 1404
15.270.3.3 free() . . . 1404
15.270.3.4 release() . . . 1406
15.270.3.5 setup() . . . 1407
15.270.3.6 take() . . . 1407
15.271 L4Re::Util::Dataspace_svr Class Reference . . . 1408
15.271.1 Detailed Description . . . 1409
15.271.2 Member Function Documentation . . . 1409
15.271.2.1 allocate() . . . 1409
15.271.2.2 clear() . . . 1410
15.271.2.3 copy() . . . 1410
15.271.2.4 is_static() . . . 1411
15.271.2.5 map() . . . 1411
15.271.2.6 map_hook() . . . 1412
15.271.2.7 page_shift() . . . 1413
15.271.2.8 release() . . . 1413
15.271.2.9 take() . . . 1413
15.272 L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference . . . 1414
15.272.1 Detailed Description . . . 1415
15.272.2 Member Function Documentation . . . 1416
15.272.2.1 foreach_available_event() . . . 1416
15.272.2.2 process() . . . 1416
15.273 L4Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference . . . 1417
15.273.1 Detailed Description . . . 1419
15.273.2 Member Function Documentation . . . 1419
15.273.2.1 attach() . . . 1420
15.273.2.2 buf() . . . 1420
15.273.2.3 detach() . . . 1420
15.274 L4Re::Util::Event_svr< SVR > Class Template Reference . . . 1421
15.274.1 Detailed Description . . . 1423
15.275 L4Re::Util::Event_t< PAYLOAD > Class Template Reference . . . 1423
15.275.1 Detailed Description . . . 1424

15.275.2 Member Enumeration Documentation . . . 1424
15.275.2.1 Mode . . . 1424
15.275.3 Member Function Documentation . . . 1424
15.275.3.1 buffer() . . . 1425
15.275.3.2 init() . . . 1425
15.275.3.3 init_poll() . . . 1426
15.275.3.4 irq() . . . 1427
15.276 L4Re::Util::Item_alloc_base Class Reference . . . 1427
15.276.1 Detailed Description . . . 1428
15.277 L4Re::Util::Names::Name Class Reference . . . 1428
15.277.1 Detailed Description . . . 1429
15.278 L4Re::Util::Names::Name_space Class Reference . . . 1429
15.278.1 Detailed Description . . . 1430
15.278.2 Member Function Documentation . . . 1431
15.278.2.1 alloc_dynamic_entry() . . . 1431
15.278.2.2 copy_receive_cap() . . . 1431
15.278.2.3 free_capability() . . . 1431
15.278.2.4 free_dynamic_entry() . . . 1432
15.278.2.5 free_epiface() . . . 1432
15.278.2.6 get_epiface() . . . 1432
15.279 L4Re::Util::Object_registry Class Reference . . . 1433
15.279.1 Detailed Description . . . 1435
15.279.2 Constructor & Destructor Documentation . . . 1435
15.279.2.1 Object_registry() [1/2] . . . 1435
15.279.2.2 Object_registry() [2/2] . . . 1435
15.279.3 Member Function Documentation . . . 1436
15.279.3.1 register_irq_obj() . . . 1436
15.279.3.2 register_obj() [1/3] . . . 1436
15.279.3.3 register_obj() [2/3] . . . 1437
15.279.3.4 register_obj() [3/3] . . . 1437
15.279.3.5 unregister_obj() . . . 1438
15.280 L4Re::Util::Ref_cap< T > Struct Template Reference . . . 1439
15.280.1 Detailed Description . . . 1439
15.281 L4Re::Util::Ref_del_cap< T > Struct Template Reference . . . 1440
15.281.1 Detailed Description . . . 1440
15.282 L4Re::Util::Registry_server< LOOP_HOOKS > Class Template Reference . . . 1441
15.282.1 Detailed Description . . . 1444
15.282.2 Constructor & Destructor Documentation . . . 1444
15.282.2.1 Registry_server() [1/3] . . . 1444
15.282.2.2 Registry_server() [2/3] . . . 1444
15.282.2.3 Registry_server() [3/3] . . . 1445
15.282.3 Member Function Documentation . . . 1445

15.282.3.1 loop() . . . 1445
15.283 L4Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference . . . 1446
15.283.1 Detailed Description . . . 1446
15.284 L4Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference . . . 1447
15.284.1 Detailed Description . . . 1447
15.285 L4Re::Util::Vcon_svr< SVR > Class Template Reference . . . 1448
15.285.1 Detailed Description . . . 1448
15.286 L4Re::Util::Video::Goos_svr Class Reference . . . 1449
15.286.1 Detailed Description . . . 1450
15.286.2 Member Function Documentation . . . 1450
15.286.2.1 get_fb() . . . 1450
15.286.2.2 init_infos() . . . 1451
15.286.2.3 refresh() . . . 1451
15.286.2.4 screen_info() . . . 1451
15.286.2.5 view_info() . . . 1452
15.287 L4Re::Vfs::Be_file Class Reference . . . 1452
15.287.1 Detailed Description . . . 1455
15.287.2 Member Function Documentation . . . 1456
15.287.2.1 data_space() . . . 1456
15.287.2.2 fstat64() . . . 1456
15.287.2.3 unlock_all_locks() . . . 1457
15.288 L4Re::Vfs::Be_file_system Class Reference . . . 1457
15.288.1 Detailed Description . . . 1458
15.288.2 Constructor & Destructor Documentation . . . 1458
15.288.2.1 Be_file_system() . . . 1459
15.288.2.2 ~Be_file_system() . . . 1459
15.288.3 Member Function Documentation . . . 1459
15.288.3.1 type() . . . 1459
15.289 L4Re::Vfs::Directory Class Reference . . . 1460
15.289.1 Detailed Description . . . 1461
15.289.2 Member Function Documentation . . . 1462
15.289.2.1 faccessat() . . . 1462
15.289.2.2 link() . . . 1462
15.289.2.3 mkdir() . . . 1463
15.289.2.4 rename() . . . 1463
15.289.2.5 rmdir() . . . 1464
15.289.2.6 symlink() . . . 1464
15.289.2.7 unlink() . . . 1464
15.290 L4Re::Vfs::File Class Reference . . . 1465
15.290.1 Detailed Description . . . 1466
15.291 L4Re::Vfs::File_system Class Reference . . . 1467
15.291.1 Detailed Description . . . 1468

15.291.2 Member Function Documentation . . . 1468
15.291.2.1 mount() . . . 1468
15.291.2.2 type() . . . 1469
15.292 L4Re::Vfs::Fs Class Reference . . . 1469
15.292.1 Detailed Description . . . 1471
15.292.2 Member Function Documentation . . . 1472
15.292.2.1 alloc_fd() . . . 1472
15.292.2.2 free_fd() . . . 1472
15.292.2.3 get_file() . . . 1473
15.292.2.4 mount() . . . 1473
15.292.2.5 set_fd() . . . 1473
15.293 L4Re::Vfs::Generic_file Class Reference . . . 1474
15.293.1 Detailed Description . . . 1476
15.293.2 Member Function Documentation . . . 1476
15.293.2.1 fchmod() . . . 1477
15.293.2.2 fstat64() . . . 1477
15.293.2.3 get_status_flags() . . . 1477
15.293.2.4 set_status_flags() . . . 1477
15.293.2.5 unlock_all_locks() . . . 1478
15.294 L4Re::Vfs::Mman Class Reference . . . 1479
15.294.1 Detailed Description . . . 1480
15.295 L4Re::Vfs::Ops Class Reference . . . 1480
15.295.1 Detailed Description . . . 1482
15.296 L4Re::Vfs::Regular_file Class Reference . . . 1483
15.296.1 Detailed Description . . . 1484
15.296.2 Member Function Documentation . . . 1485
15.296.2.1 data_space() . . . 1485
15.296.2.2 fdatasync() . . . 1485
15.296.2.3 fsync() . . . 1485
15.296.2.4 ftruncate64() . . . 1485
15.296.2.5 get_lock() . . . 1486
15.296.2.6 lseek64() . . . 1486
15.296.2.7 readv() . . . 1487
15.296.2.8 set_lock() . . . 1487
15.296.2.9 writev() . . . 1487
15.297 L4Re::Vfs::Special_file Class Reference . . . 1488
15.297.1 Detailed Description . . . 1490
15.297.2 Member Function Documentation . . . 1490
15.297.2.1 ioctl() . . . 1490
15.298 L4Re::Video::Color_component Class Reference . . . 1491
15.298.1 Detailed Description . . . 1492
15.298.2 Constructor & Destructor Documentation . . . 1492

15.298.2.1 Color_component() . . . 1492
15.298.3 Member Function Documentation . . . 1492
15.298.3.1 dump() . . . 1492
15.298.3.2 get() . . . 1493
15.298.3.3 operator==() . . . 1493
15.298.3.4 set() . . . 1493
15.298.3.5 shift() . . . 1494
15.298.3.6 size() . . . 1494
15.299 L4Re::Video::Goos Class Reference . . . 1495
15.299.1 Detailed Description . . . 1498
15.299.2 Member Enumeration Documentation . . . 1498
15.299.2.1 Flags . . . 1498
15.299.3 Member Function Documentation . . . 1499
15.299.3.1 create_buffer() . . . 1499
15.299.3.2 create_view() . . . 1499
15.299.3.3 delete_buffer() . . . 1500
15.299.3.4 delete_view() . . . 1500
15.299.3.5 get_static_buffer() . . . 1501
15.299.3.6 info() . . . 1501
15.299.3.7 view() . . . 1501
15.300 L4Re::Video::Goos::Info Struct Reference . . . 1502
15.300.1 Detailed Description . . . 1503
15.301 L4Re::Video::Pixel_info Class Reference . . . 1503
15.301.1 Detailed Description . . . 1505
15.301.2 Constructor & Destructor Documentation . . . 1505
15.301.2.1 Pixel_info() [1/2] . . . 1505
15.301.2.2 Pixel_info() [2/2] . . . 1506
15.301.3 Member Function Documentation . . . 1506
15.301.3.1 a() [1/2] . . . 1506
15.301.3.2 a() [2/2] . . . 1506
15.301.3.3 b() [1/2] . . . 1507
15.301.3.4 b() [2/2] . . . 1507
15.301.3.5 bits_per_pixel() . . . 1507
15.301.3.6 bytes_per_pixel() [1/2] . . . 1508
15.301.3.7 bytes_per_pixel() [2/2] . . . 1508
15.301.3.8 dump() . . . 1508
15.301.3.9 g() [1/2] . . . 1509
15.301.3.10 g() [2/2] . . . 1509
15.301.3.11 has_alpha() . . . 1510
15.301.3.12 operator==() . . . 1510
15.301.3.13 padding() . . . 1511
15.301.3.14 r() [1/2] . . . 1511

15.301.3.15 r() [2/2] . . . 1511
15.302 L4Re::Video::View Class Reference . . . 1512
15.302.1 Detailed Description . . . 1513
15.302.2 Member Enumeration Documentation . . . 1513
15.302.2.1 Flags . . . 1513
15.302.2.2 V_flags . . . 1514
15.302.3 Member Function Documentation . . . 1514
15.302.3.1 info() . . . 1514
15.302.3.2 refresh() . . . 1515
15.302.3.3 set_info() . . . 1515
15.302.3.4 set_viewport() . . . 1516
15.302.3.5 stack() . . . 1516
15.303 L4Re::Video::View::Info Struct Reference . . . 1517
15.303.1 Detailed Description . . . 1519
15.304 l4re_aux_t Struct Reference . . . 1520
15.304.1 Detailed Description . . . 1520
15.305 l4re_ds_stats_t Struct Reference . . . 1521
15.305.1 Detailed Description . . . 1521
15.306 l4re_elf_aux_mword_t Struct Reference . . . 1521
15.306.1 Detailed Description . . . 1522
15.307 l4re_elf_aux_t Struct Reference . . . 1522
15.307.1 Detailed Description . . . 1523
15.308 l4re_elf_aux_vma_t Struct Reference . . . 1523
15.308.1 Detailed Description . . . 1523
15.309 l4re_env_cap_entry_t Struct Reference . . . 1524
15.309.1 Detailed Description . . . 1524
15.309.2 Constructor & Destructor Documentation . . . 1524
15.309.2.1 l4re_env_cap_entry_t() . . . 1525
15.309.3 Field Documentation . . . 1525
15.309.3.1 flags . . . 1525
15.310 l4re_env_t Struct Reference . . . 1526
15.310.1 Detailed Description . . . 1527
15.311 l4re_event_t Struct Reference . . . 1527
15.311.1 Detailed Description . . . 1528
15.312 l4re_video_color_component_t Struct Reference . . . 1528
15.312.1 Detailed Description . . . 1529
15.313 l4re_video_goos_info_t Struct Reference . . . 1529
15.313.1 Detailed Description . . . 1530
15.314 l4re_video_pixel_info_t Struct Reference . . . 1530
15.314.1 Detailed Description . . . 1531
15.315 l4re_video_view_info_t Struct Reference . . . 1531
15.315.1 Detailed Description . . . 1533

15.316 l4re_video_view_t Struct Reference . . . 1533
15.316.1 Detailed Description . . . 1533
15.317 l4util_idt_desc_t Struct Reference . . . 1534
15.317.1 Detailed Description . . . 1534
15.318 l4util_idt_header_t Struct Reference . . . 1535
15.318.1 Detailed Description . . . 1535
15.319 l4util_l4mod_info Struct Reference . . . 1536
15.319.1 Detailed Description . . . 1536
15.319.2 Field Documentation . . . 1536
15.319.2.1 vbe_ctrl_info . . . 1537
15.320 l4util_l4mod_mod Struct Reference . . . 1537
15.320.1 Detailed Description . . . 1538
15.321 l4util_mb_addr_range_t Struct Reference . . . 1538
15.321.1 Detailed Description . . . 1539
15.322 l4util_mb_apm_t Struct Reference . . . 1539
15.322.1 Detailed Description . . . 1539
15.323 l4util_mb_drive_t Struct Reference . . . 1540
15.323.1 Detailed Description . . . 1540
15.323.2 Field Documentation . . . 1541
15.323.2.1 drive_cylinders . . . 1541
15.323.2.2 drive_mode . . . 1541
15.323.2.3 drive_number . . . 1541
15.324 l4util_mb_info_t Struct Reference . . . 1542
15.324.1 Detailed Description . . . 1543
15.325 l4util_mb_mod_t Struct Reference . . . 1543
15.325.1 Detailed Description . . . 1544
15.326 l4util_mb_vbe_ctrl_t Struct Reference . . . 1544
15.326.1 Detailed Description . . . 1545
15.327 l4util_mb_vbe_mode_t Struct Reference . . . 1545
15.327.1 Detailed Description . . . 1548
15.328 L4vbus::Device Class Reference . . . 1549
15.328.1 Detailed Description . . . 1551
15.328.2 Member Function Documentation . . . 1551
15.328.2.1 bus_cap() . . . 1551
15.328.2.2 dev_handle() . . . 1552
15.328.2.3 device() . . . 1553
15.328.2.4 device_by_hid() . . . 1553
15.328.2.5 get_resource() . . . 1554
15.328.2.6 is_compatible() . . . 1555
15.328.2.7 next_device() . . . 1556
15.328.2.8 operator"!=() . . . 1556
15.328.2.9 operator==() . . . 1556

15.328.3 Field Documentation . . . 1557
15.328.3.1 _bus . . . 1557
15.329 L4vbus::Gpio_module Class Reference . . . 1557
15.329.1 Detailed Description . . . 1560
15.329.2 Member Function Documentation . . . 1560
15.329.2.1 config_pad() . . . 1560
15.329.2.2 get() . . . 1561
15.329.2.3 pin() . . . 1561
15.329.2.4 set() . . . 1562
15.329.2.5 setup() . . . 1563
15.330 L4vbus::Gpio_module::Pin_slice Struct Reference . . . 1563
15.330.1 Detailed Description . . . 1564
15.331 L4vbus::Gpio_pin Class Reference . . . 1564
15.331.1 Detailed Description . . . 1567
15.331.2 Member Function Documentation . . . 1567
15.331.2.1 config_get() . . . 1567
15.331.2.2 config_pad() . . . 1568
15.331.2.3 config_pull() . . . 1568
15.331.2.4 get() . . . 1569
15.331.2.5 pin() . . . 1570
15.331.2.6 set() . . . 1570
15.331.2.7 setup() . . . 1570
15.331.2.8 to_irq() . . . 1571
15.332 L4vbus::Icu Class Reference . . . 1572
15.332.1 Detailed Description . . . 1574
15.332.2 Member Enumeration Documentation . . . 1574
15.332.2.1 Src_types . . . 1574
15.332.3 Member Function Documentation . . . 1574
15.332.3.1 vicu() . . . 1574
15.333 L4vbus::Pci_dev Class Reference . . . 1575
15.333.1 Detailed Description . . . 1578
15.333.2 Member Function Documentation . . . 1578
15.333.2.1 cfg_read() . . . 1578
15.333.2.2 cfg_write() . . . 1579
15.333.2.3 irq_enable() . . . 1579
15.334 L4vbus::Pci_host_bridge Class Reference . . . 1580
15.334.1 Detailed Description . . . 1583
15.334.2 Member Function Documentation . . . 1583
15.334.2.1 cfg_read() . . . 1583
15.334.2.2 cfg_write() . . . 1584
15.334.2.3 irq_enable() . . . 1585
15.335 L4vbus::Pm< DEC > Class Template Reference . . . 1586

15.335.1 Detailed Description . . . 1587
15.336 L4vbus::Vbus Class Reference . . . 1587
15.336.1 Detailed Description . . . 1590
15.336.2 Member Function Documentation . . . 1590
15.336.2.1 assign_dma_domain() [1/2] . . . 1590
15.336.2.2 assign_dma_domain() [2/2] . . . 1591
15.336.2.3 release_ioport() . . . 1592
15.336.2.4 release_resource() . . . 1593
15.336.2.5 request_ioport() . . . 1594
15.336.2.6 request_resource() . . . 1595
15.336.2.7 root() . . . 1596
15.337 l4vbus_device_t Struct Reference . . . 1596
15.337.1 Detailed Description . . . 1597
15.338 l4vbus_resource_t Struct Reference . . . 1597
15.338.1 Detailed Description . . . 1598
15.339 L4vcpu::State Class Reference . . . 1598
15.339.1 Detailed Description . . . 1599
15.339.2 Constructor & Destructor Documentation . . . 1599
15.339.2.1 State() . . . 1599
15.339.3 Member Function Documentation . . . 1599
15.339.3.1 add() . . . 1599
15.339.3.2 clear() . . . 1600
15.339.3.3 set() . . . 1600
15.340 L4vcpu::Vcpu Class Reference . . . 1600
15.340.1 Detailed Description . . . 1604
15.340.2 Member Function Documentation . . . 1604
15.340.2.1 cast() [1/2] . . . 1604
15.340.2.2 cast() [2/2] . . . 1604
15.340.2.3 entry_ip() . . . 1605
15.340.2.4 entry_sp() . . . 1605
15.340.2.5 ext_alloc() . . . 1605
15.340.2.6 i() [1/2] . . . 1606
15.340.2.7 i() [2/2] . . . 1606
15.340.2.8 irq_disable_save() . . . 1607
15.340.2.9 irq_enable() . . . 1607
15.340.2.10 irq_restore() . . . 1608
15.340.2.11 is_irq_entry() . . . 1608
15.340.2.12 is_page_fault_entry() . . . 1609
15.340.2.13 r() [1/2] . . . 1609
15.340.2.14 r() [2/2] . . . 1610
15.340.2.15 saved_state() [1/2] . . . 1610
15.340.2.16 saved_state() [2/2] . . . 1610

15.340.2.17 state() [1/2] . . . 1611
15.340.2.18 state() [2/2] . . . 1611
15.340.2.19 task() . . . 1611
15.340.2.20 wait_for_event() . . . 1612
15.341 L4virtio::Device Class Reference . . . 1612
15.341.1 Detailed Description . . . 1615
15.341.2 Member Function Documentation . . . 1615
15.341.2.1 config_queue() . . . 1615
15.341.2.2 device_config() . . . 1616
15.341.2.3 device_notification_irq() . . . 1617
15.341.2.4 register_ds() . . . 1617
15.341.2.5 register_iface() . . . 1618
15.341.2.6 set_status() . . . 1619
15.342 L4virtio::Driver::Block_device Class Reference . . . 1619
15.342.1 Detailed Description . . . 1622
15.342.2 Member Function Documentation . . . 1622
15.342.2.1 add_block() . . . 1622
15.342.2.2 process_request() . . . 1623
15.342.2.3 process_used_queue() . . . 1624
15.342.2.4 send_request() . . . 1624
15.342.2.5 setup_device() . . . 1625
15.342.2.6 start_request() . . . 1627
15.343 L4virtio::Driver::Block_device::Handle Class Reference . . . 1627
15.343.1 Detailed Description . . . 1628
15.344 L4virtio::Driver::Device Class Reference . . . 1628
15.344.1 Detailed Description . . . 1631
15.344.2 Member Function Documentation . . . 1631
15.344.2.1 bind_notification_irq() . . . 1631
15.344.2.2 config_queue() . . . 1632
15.344.2.3 driver_acknowledge() . . . 1633
15.344.2.4 driver_connect() . . . 1634
15.344.2.5 max_queue_size() . . . 1635
15.344.2.6 register_ds() . . . 1636
15.344.2.7 send() . . . 1637
15.344.2.8 send_and_wait() . . . 1638
15.344.2.9 wait() . . . 1639
15.344.2.10 wait_for_next_used() . . . 1640
15.345 L4virtio::Driver::Virtqueue Class Reference . . . 1641
15.345.1 Detailed Description . . . 1644
15.345.2 Member Function Documentation . . . 1644
15.345.2.1 alloc_descriptor() . . . 1644
15.345.2.2 desc() . . . 1645

15.345.2.3 enqueue_descriptor() . . . 1646
15.345.2.4 find_next_used() . . . 1647
15.345.2.5 free_descriptor() . . . 1647
15.345.2.6 init_queue() [1/2] . . . 1648
15.345.2.7 init_queue() [2/2] . . . 1648
15.345.2.8 initialize_rings() . . . 1649
15.346 L4virtio::Ptr< T > Class Template Reference . . . 1650
15.346.1 Detailed Description . . . 1652
15.346.2 Member Enumeration Documentation . . . 1652
15.346.2.1 Invalid_type . . . 1652
15.346.3 Member Function Documentation . . . 1652
15.346.3.1 get() . . . 1652
15.346.3.2 is_valid() . . . 1653
15.347 L4virtio::Svr::Bad_descriptor Struct Reference . . . 1654
15.347.1 Detailed Description . . . 1655
15.347.2 Member Enumeration Documentation . . . 1655
15.347.2.1 Error . . . 1655
15.347.3 Constructor & Destructor Documentation . . . 1655
15.347.3.1 Bad_descriptor() . . . 1655
15.347.4 Member Function Documentation . . . 1656
15.347.4.1 message() . . . 1656
15.348 L4virtio::Svr::Block_dev_base< Ds_data > Class Template Reference . . . 1656
15.348.1 Detailed Description . . . 1659
15.348.2 Constructor & Destructor Documentation . . . 1660
15.348.2.1 Block_dev_base() . . . 1660
15.348.3 Member Function Documentation . . . 1660
15.348.3.1 finalize_request() . . . 1660
15.348.3.2 get_writeback() . . . 1661
15.348.3.3 process_request() . . . 1661
15.348.3.4 set_blk_size() . . . 1662
15.348.3.5 set_config_wce() . . . 1662
15.348.3.6 set_discard() . . . 1662
15.348.3.7 set_size_max() . . . 1664
15.348.3.8 set_topology() . . . 1664
15.348.3.9 set_write_zeroes() . . . 1664
15.349 L4virtio::Svr::Block_request< Ds_data > Class Template Reference . . . 1665
15.349.1 Detailed Description . . . 1666
15.349.2 Member Function Documentation . . . 1666
15.349.2.1 data_size() . . . 1666
15.349.2.2 next_block() . . . 1667
15.350 L4virtio::Svr::Data_buffer Struct Reference . . . 1668
15.350.1 Detailed Description . . . 1669

15.350.2 Constructor & Destructor Documentation . . . 1669
15.350.2.1 Data_buffer() . . . 1669
15.350.3 Member Function Documentation . . . 1669
15.350.3.1 copy_to() . . . 1669
15.350.3.2 done() . . . 1670
15.350.3.3 set() . . . 1670
15.350.3.4 skip() . . . 1671
15.351 L4virtio::Svr::Dev_config Class Reference . . . 1672
15.351.1 Detailed Description . . . 1673
15.351.2 Constructor & Destructor Documentation . . . 1673
15.351.2.1 Dev_config() [1/2] . . . 1673
15.351.2.2 Dev_config() [2/2] . . . 1674
15.351.3 Member Function Documentation . . . 1674
15.351.3.1 change_queue_config() . . . 1675
15.351.3.2 ds() . . . 1675
15.351.3.3 get_cmd() . . . 1676
15.351.3.4 guest_features() . . . 1676
15.351.3.5 hdr() . . . 1677
15.351.3.6 negotiated_features() . . . 1677
15.351.3.7 qconfig() . . . 1678
15.351.3.8 reset_cmd() . . . 1679
15.351.3.9 reset_queue() . . . 1679
15.351.3.10 set_device_needs_reset() . . . 1680
15.351.3.11 set_status() . . . 1681
15.351.3.12 status() . . . 1682
15.352 L4virtio::Svr::Dev_features Struct Reference . . . 1683
15.352.1 Detailed Description . . . 1684
15.353 L4virtio::Svr::Dev_status Struct Reference . . . 1684
15.353.1 Detailed Description . . . 1686
15.353.2 Member Function Documentation . . . 1686
15.353.2.1 running() . . . 1686
15.354 L4virtio::Svr::Device_t< DATA > Class Template Reference . . . 1687
15.354.1 Detailed Description . . . 1689
15.354.2 Member Function Documentation . . . 1689
15.354.2.1 device_error() . . . 1690
15.354.2.2 device_notify_irq() . . . 1690
15.354.2.3 handle_mem_cmd_write() . . . 1691
15.354.2.4 init_mem_info() . . . 1691
15.354.2.5 register_driver_irq() . . . 1692
15.354.2.6 reset_queue_config() . . . 1692
15.354.2.7 setup_queue() . . . 1693
15.355 L4virtio::Svr::Driver_mem_list_t< DATA > Class Template Reference . . . 1694

15.355.1 Detailed Description . . . 1697
15.355.2 Member Function Documentation . . . 1697
15.355.2.1 add() . . . 1697
15.355.2.2 find() . . . 1698
15.355.2.3 full() . . . 1699
15.355.2.4 init() . . . 1699
15.355.2.5 load_desc() [1/3] . . . 1700
15.355.2.6 load_desc() [2/3] . . . 1701
15.355.2.7 load_desc() [3/3] . . . 1702
15.355.2.8 remove() . . . 1703
15.356 L4virtio::Svr::Driver_mem_region_t< DATA > Class Template Reference . . . 1704
15.356.1 Detailed Description . . . 1705
15.356.2 Constructor & Destructor Documentation . . . 1706
15.356.2.1 Driver_mem_region_t() . . . 1706
15.356.3 Member Function Documentation . . . 1706
15.356.3.1 contains() . . . 1707
15.356.3.2 drv_base() . . . 1707
15.356.3.3 ds() . . . 1708
15.356.3.4 ds_offset() . . . 1708
15.356.3.5 empty() . . . 1708
15.356.3.6 flags() . . . 1708
15.356.3.7 is_writable() . . . 1709
15.356.3.8 local() . . . 1709
15.356.3.9 local_base() . . . 1710
15.356.3.10 size() . . . 1710
15.357 L4virtio::Svr::Request_processor Class Reference . . . 1711
15.357.1 Detailed Description . . . 1711
15.357.2 Member Function Documentation . . . 1712
15.357.2.1 current_flags() . . . 1712
15.357.2.2 has_more() . . . 1712
15.357.2.3 next() . . . 1713
15.357.2.4 start() [1/2] . . . 1715
15.357.2.5 start() [2/2] . . . 1717
15.358 L4virtio::Svr::Virtqueue Class Reference . . . 1718
15.358.1 Detailed Description . . . 1720
15.358.2 Member Function Documentation . . . 1720
15.358.2.1 consumed() . . . 1720
15.358.2.2 desc() . . . 1721
15.358.2.3 desc_avail() . . . 1721
15.358.2.4 disable_notify() . . . 1722
15.358.2.5 enable_notify() . . . 1723
15.358.2.6 next_avail() . . . 1723

15.359 L4virtio::Svr::Virtqueue::Head_desc Class Reference . . . 1724
15.359.1 Detailed Description . . . 1724
15.359.2 Member Function Documentation . . . 1724
15.359.2.1 desc() . . . 1725
15.359.2.2 operator Null_ptr_check const *() . . . 1725
15.359.2.3 valid() . . . 1725
15.360 L4virtio::Virtqueue Class Reference . . . 1726
15.360.1 Detailed Description . . . 1729
15.360.2 Member Function Documentation . . . 1729
15.360.2.1 avail_align() . . . 1729
15.360.2.2 avail_size() . . . 1729
15.360.2.3 desc_align() . . . 1730
15.360.2.4 desc_size() . . . 1731
15.360.2.5 disable() . . . 1732
15.360.2.6 dump() . . . 1732
15.360.2.7 get_avail_idx() . . . 1733
15.360.2.8 get_tail_avail_idx() . . . 1733
15.360.2.9 no_notify_guest() . . . 1733
15.360.2.10 no_notify_host() [1/2] . . . 1734
15.360.2.11 no_notify_host() [2/2] . . . 1735
15.360.2.12 num() . . . 1735
15.360.2.13 ready() . . . 1736
15.360.2.14 setup() . . . 1737
15.360.2.15 setup_simple() . . . 1738
15.360.2.16 total_size() [1/2] . . . 1739
15.360.2.17 total_size() [2/2] . . . 1740
15.360.2.18 used_align() . . . 1741
15.360.2.19 used_size() . . . 1741
15.361 L4virtio::Virtqueue::Avail Class Reference . . . 1742
15.361.1 Detailed Description . . . 1743
15.362 L4virtio::Virtqueue::Avail::Flags Struct Reference . . . 1744
15.362.1 Detailed Description . . . 1744
15.362.2 Member Typedef Documentation . . . 1744
15.362.2.1 no_irq_bfm_t . . . 1745
15.363 L4virtio::Virtqueue::Desc Class Reference . . . 1745
15.363.1 Detailed Description . . . 1747
15.364 L4virtio::Virtqueue::Desc::Flags Struct Reference . . . 1747
15.364.1 Detailed Description . . . 1748
15.364.2 Member Typedef Documentation . . . 1748
15.364.2.1 indirect_bfm_t . . . 1748
15.364.2.2 next_bfm_t . . . 1749
15.364.2.3 write_bfm_t . . . 1749

15.365 L4virtio::Virtqueue::Used Class Reference . . . 1749
15.365.1 Detailed Description . . . 1750
15.366 L4virtio::Virtqueue::Used::Flags Struct Reference . . . 1750
15.366.1 Detailed Description . . . 1751
15.366.2 Member Typedef Documentation . . . 1751
15.366.2.1 no_notify_bfm_t . . . 1751
15.367 L4virtio::Virtqueue::Used_elem Struct Reference . . . 1752
15.367.1 Detailed Description . . . 1752
15.367.2 Constructor & Destructor Documentation . . . 1752
15.367.2.1 Used_elem() . . . 1752
15.368 l4virtio_block_config_t Struct Reference . . . 1753
15.368.1 Detailed Description . . . 1754
15.369 l4virtio_block_discard_t Struct Reference . . . 1754
15.369.1 Detailed Description . . . 1754
15.370 l4virtio_block_header_t Struct Reference . . . 1755
15.370.1 Detailed Description . . . 1755
15.371 l4virtio_config_hdr_t Struct Reference . . . 1756
15.371.1 Detailed Description . . . 1757
15.371.2 Field Documentation . . . 1757
15.371.2.1 status . . . 1757
15.372 l4virtio_config_queue_t Struct Reference . . . 1757
15.372.1 Detailed Description . . . 1758
15.373 l4virtio_input_absinfo_t Struct Reference . . . 1759
15.373.1 Detailed Description . . . 1759
15.374 l4virtio_input_config_t Struct Reference . . . 1759
15.374.1 Detailed Description . . . 1760
15.375 l4virtio_input_devids_t Struct Reference . . . 1760
15.375.1 Detailed Description . . . 1761
15.376 l4virtio_input_event_t Struct Reference . . . 1761
15.376.1 Detailed Description . . . 1762

16 File Documentation **1763**
16.1 amd64/l4/util/apic.h File Reference . . . 1763
16.1.1 Detailed Description . . . 1763
16.2 apic.h . . . 1764
16.3 x86/l4/util/apic.h File Reference . . . 1768
16.3.1 Detailed Description . . . 1769
16.4 apic.h . . . 1769
16.5 amd64/l4/util/idt.h File Reference . . . 1774
16.5.1 Detailed Description . . . 1775
16.6 idt.h . . . 1775
16.7 x86/l4/util/idt.h File Reference . . . 1776

16.7.1 Detailed Description . . . 1776
16.8 idt.h . . . 1777
16.9 amd64/l4/util/perform.h File Reference . . . 1777
16.9.1 Detailed Description . . . 1778
16.10 perform.h . . . 1778
16.11 x86/l4/util/perform.h File Reference . . . 1783
16.11.1 Detailed Description . . . 1784
16.12 perform.h . . . 1784
16.13 amd64/l4/util/rdtsc.h File Reference . . . 1789
16.13.1 Detailed Description . . . 1791
16.14 rdtsc.h . . . 1791
16.15 x86/l4/util/rdtsc.h File Reference . . . 1793
16.15.1 Detailed Description . . . 1795
16.16 rdtsc.h . . . 1795
16.17 amd64/l4/util/spin.h File Reference . . . 1798
16.17.1 Detailed Description . . . 1798
16.18 spin.h . . . 1798
16.19 x86/l4/util/spin.h File Reference . . . 1799
16.19.1 Detailed Description . . . 1799
16.20 spin.h . . . 1800
16.21 amd64/l4/util/util.h File Reference . . . 1800
16.21.1 Detailed Description . . . 1801
16.21.2 Function Documentation . . . 1801
16.21.2.1 l4_sleep() . . . 1801
16.21.2.2 l4util_micros2l4to() . . . 1802
16.22 util.h . . . 1802
16.23 x86/l4/util/util.h File Reference . . . 1803
16.23.1 Detailed Description . . . 1804
16.23.2 Function Documentation . . . 1804
16.23.2.1 l4_sleep() . . . 1804
16.23.2.2 l4util_micros2l4to() . . . 1805
16.24 util.h . . . 1805
16.25 amd64/l4f/l4/sys/segment.h File Reference . . . 1806
16.25.1 Detailed Description . . . 1807
16.25.2 Function Documentation . . . 1807
16.25.2.1 fiasco_amd64_set_fs() . . . 1807
16.25.2.2 fiasco_amd64_set_segment_base() . . . 1808
16.26 segment.h . . . 1808
16.27 amd64/l4/sys/segment.h File Reference . . . 1809
16.27.1 Detailed Description . . . 1810
16.27.2 Enumeration Type Documentation . . . 1810
16.27.2.1 L4_sys_segment . . . 1810

16.27.2.2 L4_task_ldt_x86_consts . . . 1810
16.27.3 Function Documentation . . . 1811
16.27.3.1 fiasco_amd64_segment_info() . . . 1811
16.27.3.2 fiasco_amd64_set_fs() . . . 1811
16.27.3.3 fiasco_amd64_set_segment_base() . . . 1812
16.28 segment.h . . . 1812
16.29 x86/l4f/l4/sys/segment.h File Reference . . . 1814
16.29.1 Detailed Description . . . 1815
16.30 segment.h . . . 1815
16.31 x86/l4/sys/segment.h File Reference . . . 1815
16.31.1 Detailed Description . . . 1816
16.31.2 Enumeration Type Documentation . . . 1817
16.31.2.1 L4_task_ldt_x86_consts . . . 1817
16.32 segment.h . . . 1817
16.33 amd64/l4f/l4/util/port_io.h File Reference . . . 1818
16.33.1 Detailed Description . . . 1818
16.34 port_io.h . . . 1819
16.35 amd64/l4/util/port_io.h File Reference . . . 1819
16.35.1 Detailed Description . . . 1819
16.36 port_io.h . . . 1819
16.37 x86/l4f/l4/util/port_io.h File Reference . . . 1820
16.37.1 Detailed Description . . . 1821
16.37.2 Function Documentation . . . 1821
16.37.2.1 l4util_ioport_map() . . . 1821
16.38 port_io.h . . . 1822
16.39 x86/l4/util/port_io.h File Reference . . . 1823
16.39.1 Detailed Description . . . 1825
16.40 port_io.h . . . 1825
16.41 arm/l4/sys/linkage.h File Reference . . . 1827
16.41.1 Detailed Description . . . 1827
16.42 linkage.h . . . 1827
16.43 amd64/l4/sys/linkage.h File Reference . . . 1828
16.43.1 Detailed Description . . . 1828
16.44 linkage.h . . . 1828
16.45 x86/l4/sys/linkage.h File Reference . . . 1829
16.45.1 Detailed Description . . . 1829
16.46 linkage.h . . . 1829
16.47 arm/l4/sys/mem_op.h File Reference . . . 1830
16.47.1 Detailed Description . . . 1830
16.48 mem_op.h . . . 1831
16.49 arm/l4/sys/vm.h File Reference . . . 1832
16.49.1 Detailed Description . . . 1832

16.50 vm.h . . . 1832
16.51 arm/l4/util/bitops_arch.h File Reference . . . 1833
16.51.1 Detailed Description . . . 1833
16.52 bitops_arch.h . . . 1834
16.53 amd64/l4/util/bitops_arch.h File Reference . . . 1834
16.53.1 Detailed Description . . . 1834
16.54 bitops_arch.h . . . 1834
16.55 x86/l4/util/bitops_arch.h File Reference . . . 1837
16.55.1 Detailed Description . . . 1837
16.56 bitops_arch.h . . . 1838
16.57 arm/l4/util/cpu.h File Reference . . . 1841
16.57.1 Detailed Description . . . 1841
16.58 cpu.h . . . 1841
16.59 amd64/l4/util/cpu.h File Reference . . . 1842
16.59.1 Detailed Description . . . 1842
16.60 cpu.h . . . 1843
16.61 x86/l4/util/cpu.h File Reference . . . 1843
16.61.1 Detailed Description . . . 1844
16.62 cpu.h . . . 1845
16.63 arm/l4/util/l4_macros.h File Reference . . . 1846
16.63.1 Detailed Description . . . 1846
16.64 l4_macros.h . . . 1846
16.65 amd64/l4/util/l4_macros.h File Reference . . . 1846
16.65.1 Detailed Description . . . 1847
16.66 l4_macros.h . . . 1847
16.67 l4/util/l4_macros.h File Reference . . . 1847
16.67.1 Detailed Description . . . 1847
16.68 l4_macros.h . . . 1848
16.69 x86/l4/util/l4_macros.h File Reference . . . 1848
16.69.1 Detailed Description . . . 1848
16.70 l4_macros.h . . . 1848
16.71 arm/l4/util/mbi_argv.h File Reference . . . 1849
16.71.1 Detailed Description . . . 1849
16.72 mbi_argv.h . . . 1849
16.73 amd64/l4/util/mbi_argv.h File Reference . . . 1850
16.73.1 Detailed Description . . . 1850
16.74 mbi_argv.h . . . 1851
16.75 x86/l4/util/mbi_argv.h File Reference . . . 1851
16.75.1 Detailed Description . . . 1852
16.76 mbi_argv.h . . . 1852
16.77 arm/l4f/l4/sys/syscall_defs.h File Reference . . . 1853
16.77.1 Detailed Description . . . 1853

16.78 syscall_defs.h . . . 1853
16.79 contrib/libio-io/l4/io/io.h File Reference . . . 1853
16.79.1 Function Documentation . . . 1855
16.79.1.1 l4io_get_root_device() . . . 1855
16.79.1.2 l4io_iterate_devices() . . . 1855
16.79.1.3 l4io_request_all_ioports() . . . 1856
16.79.1.4 l4io_request_icu() . . . 1856
16.80 io.h . . . 1856
16.81 l4/cxx/alloc.h File Reference . . . 1857
16.81.1 Detailed Description . . . 1858
16.82 alloc.h . . . 1858
16.83 l4/cxx/avl_map File Reference . . . 1858
16.83.1 Detailed Description . . . 1860
16.84 avl_map . . . 1860
16.85 l4/cxx/avl_set File Reference . . . 1861
16.85.1 Detailed Description . . . 1862
16.86 avl_set . . . 1863
16.87 l4/cxx/avl_tree File Reference . . . 1866
16.87.1 Detailed Description . . . 1867
16.88 avl_tree . . . 1868
16.89 l4/cxx/basic_ostream File Reference . . . 1872
16.89.1 Detailed Description . . . 1872
16.90 basic_ostream . . . 1873
16.91 l4/cxx/basic_vector.h File Reference . . . 1876
16.91.1 Detailed Description . . . 1876
16.92 basic_vector.h . . . 1877
16.93 l4/cxx/bits/bst.h File Reference . . . 1877
16.93.1 Detailed Description . . . 1878
16.94 bst.h . . . 1879
16.95 l4/cxx/bits/bst_base.h File Reference . . . 1881
16.95.1 Detailed Description . . . 1883
16.96 bst_base.h . . . 1883
16.97 l4/cxx/bits/bst_iter.h File Reference . . . 1884
16.97.1 Detailed Description . . . 1885
16.98 bst_iter.h . . . 1886
16.99 l4/cxx/bits/smart_ptr_list.h File Reference . . . 1887
16.99.1 Detailed Description . . . 1889
16.100 smart_ptr_list.h . . . 1889
16.101 l4/cxx/exceptions File Reference . . . 1890
16.101.1 Detailed Description . . . 1892
16.102 exceptions . . . 1892
16.103 l4/cxx/iostream File Reference . . . 1894

16.103.1 Detailed Description . . . 1895
16.104 iostream . . . 1896
16.105 l4/cxx/ipc_helper File Reference . . . 1896
16.105.1 Detailed Description . . . 1897
16.106 ipc_helper . . . 1897
16.107 l4/cxx/ipc_stream File Reference . . . 1898
16.107.1 Detailed Description . . . 1900
16.107.2 Function Documentation . . . 1900
16.107.2.1 operator<<() [1/4] . . . 1900
16.107.2.2 operator<<() [2/4] . . . 1901
16.107.2.3 operator<<() [3/4] . . . 1902
16.107.2.4 operator<<() [4/4] . . . 1902
16.107.2.5 operator>>() [1/6] . . . 1903
16.107.2.6 operator>>() [2/6] . . . 1904
16.107.2.7 operator>>() [3/6] . . . 1905
16.107.2.8 operator>>() [4/6] . . . 1905
16.107.2.9 operator>>() [5/6] . . . 1906
16.107.2.10 operator>>() [6/6] . . . 1907
16.108 ipc_stream . . . 1908
16.109 l4/cxx/l4iostream File Reference . . . 1916
16.109.1 Detailed Description . . . 1916
16.110 l4iostream . . . 1917
16.111 l4/cxx/l4types.h File Reference . . . 1917
16.111.1 Detailed Description . . . 1918
16.112 l4types.h . . . 1918
16.113 l4/cxx/main_thread File Reference . . . 1919
16.113.1 Detailed Description . . . 1920
16.114 main_thread . . . 1920
16.115 l4/cxx/pair File Reference . . . 1920
16.115.1 Detailed Description . . . 1921
16.116 pair . . . 1921
16.117 l4/cxx/ref_ptr_list File Reference . . . 1922
16.117.1 Detailed Description . . . 1923
16.118 ref_ptr_list . . . 1923
16.119 l4/cxx/std_exc_io File Reference . . . 1923
16.119.1 Detailed Description . . . 1924
16.120 std_exc_io . . . 1924
16.121 l4/cxx/string.h File Reference . . . 1925
16.121.1 Detailed Description . . . 1926
16.122 string.h . . . 1927
16.123 l4/cxx/unique_ptr_list File Reference . . . 1927
16.123.1 Detailed Description . . . 1928

16.124 unique_ptr_list . . . 1929
16.125 l4/irq/irq.h File Reference . . . 1929
16.125.1 Detailed Description . . . 1930
16.126 irq.h . . . 1930
16.127 arm/l4/util/irq.h File Reference . . . 1931
16.127.1 Detailed Description . . . 1932
16.128 irq.h . . . 1932
16.129 amd64/l4/util/irq.h File Reference . . . 1933
16.129.1 Detailed Description . . . 1934
16.129.2 Function Documentation . . . 1934
16.129.2.1 l4util_irq_acknowledge() . . . 1934
16.130 irq.h . . . 1935
16.131 x86/l4/util/irq.h File Reference . . . 1935
16.131.1 Detailed Description . . . 1936
16.131.2 Function Documentation . . . 1936
16.131.2.1 l4util_irq_acknowledge() . . . 1936
16.132 irq.h . . . 1937
16.133 l4/sys/irq.h File Reference . . . 1938
16.133.1 Detailed Description . . . 1939
16.134 irq.h . . . 1940
16.135 l4/libedid/edid.h File Reference . . . 1942
16.136 edid.h . . . 1943
16.137 l4/libgfxbitmap/bitmap.h File Reference . . . 1943
16.137.1 Detailed Description . . . 1945
16.137.2 Macro Definition Documentation . . . 1945
16.137.2.1 pSLIM_BMAP_START_LSB . . . 1945
16.137.3 Typedef Documentation . . . 1945
16.137.3.1 gfxbitmap_color_pix_t . . . 1946
16.137.3.2 gfxbitmap_color_t . . . 1946
16.137.4 Function Documentation . . . 1946
16.137.4.1 gfxbitmap_bmap() . . . 1946
16.137.4.2 gfxbitmap_convert_color() . . . 1947
16.137.4.3 gfxbitmap_copy() . . . 1947
16.137.4.4 gfxbitmap_fill() . . . 1947
16.137.4.5 gfxbitmap_set() . . . 1948
16.138 bitmap.h . . . 1949
16.139 l4/libgfxbitmap/font.h File Reference . . . 1949
16.139.1 Detailed Description . . . 1951
16.139.2 Enumeration Type Documentation . . . 1951
16.139.2.1 anonymous enum . . . 1951
16.139.3 Function Documentation . . . 1951
16.139.3.1 gfxbitmap_font_data() . . . 1951

16.139.3.2 gfxbitmap_font_get() . . . 1952
16.139.3.3 gfxbitmap_font_height() . . . 1952
16.139.3.4 gfxbitmap_font_init() . . . 1952
16.139.3.5 gfxbitmap_font_text() . . . 1953
16.139.3.6 gfxbitmap_font_text_scale() . . . 1953
16.139.3.7 gfxbitmap_font_width() . . . 1954
16.140 font.h . . . 1954
16.141 l4/libgfxbitmap/support File Reference . . . 1955
16.141.1 Detailed Description . . . 1956
16.142 support . . . 1957
16.143 l4/re/c/dataspace.h File Reference . . . 1957
16.143.1 Detailed Description . . . 1958
16.144 dataspace.h . . . 1958
16.145 l4/re/c/debug.h File Reference . . . 1959
16.145.1 Detailed Description . . . 1960
16.146 debug.h . . . 1960
16.147 l4/re/c/dma_space.h File Reference . . . 1961
16.147.1 Detailed Description . . . 1962
16.147.2 Enumeration Type Documentation . . . 1962
16.147.2.1 l4re_dma_space_direction . . . 1962
16.147.2.2 l4re_dma_space_space_attribs . . . 1962
16.148 dma_space.h . . . 1963
16.149 l4/re/c/event.h File Reference . . . 1964
16.149.1 Detailed Description . . . 1965
16.150 event.h . . . 1965
16.151 l4/re/event.h File Reference . . . 1966
16.151.1 Detailed Description . . . 1967
16.152 event.h . . . 1967
16.153 l4/re/c/log.h File Reference . . . 1968
16.153.1 Detailed Description . . . 1968
16.154 log.h . . . 1969
16.155 l4/re/c/mem_alloc.h File Reference . . . 1969
16.155.1 Detailed Description . . . 1970
16.156 mem_alloc.h . . . 1971
16.157 l4/re/c/namespace.h File Reference . . . 1971
16.157.1 Detailed Description . . . 1972
16.158 namespace.h . . . 1973
16.159 l4/re/c/rm.h File Reference . . . 1973
16.159.1 Detailed Description . . . 1975
16.160 rm.h . . . 1975
16.161 l4/re/c/util/cap_alloc.h File Reference . . . 1977
16.161.1 Detailed Description . . . 1978

16.162 cap_alloc.h . . . 1978
16.163 l4/re/c/util/kumem_alloc.h File Reference . . . 1979
16.163.1 Detailed Description . . . 1980
16.163.2 Function Documentation . . . 1980
16.163.2.1 l4re_util_kumem_alloc() . . . 1980
16.164 kumem_alloc.h . . . 1980
16.165 l4/re/c/util/video/goos_fb.h File Reference . . . 1981
16.165.1 Detailed Description . . . 1982
16.166 goos_fb.h . . . 1982
16.167 l4/re/c/video/colors.h File Reference . . . 1982
16.167.1 Detailed Description . . . 1984
16.168 colors.h . . . 1984
16.169 l4/re/c/video/goos.h File Reference . . . 1985
16.169.1 Detailed Description . . . 1986
16.170 goos.h . . . 1987
16.171 l4/re/c/video/view.h File Reference . . . 1988
16.171.1 Detailed Description . . . 1989
16.172 view.h . . . 1990
16.173 l4/re/cap_alloc File Reference . . . 1991
16.173.1 Detailed Description . . . 1992
16.174 cap_alloc . . . 1992
16.175 l4/re/util/cap_alloc File Reference . . . 1994
16.175.1 Detailed Description . . . 1996
16.176 cap_alloc . . . 1996
16.177 l4/re/dataspace File Reference . . . 1997
16.177.1 Detailed Description . . . 1998
16.178 dataspace . . . 1999
16.179 l4/re/dataspace-sys.h File Reference . . . 2000
16.179.1 Detailed Description . . . 2001
16.180 dataspace-sys.h . . . 2001
16.181 l4/re/debug File Reference . . . 2001
16.181.1 Detailed Description . . . 2002
16.182 debug . . . 2003
16.183 l4/re/dma_space File Reference . . . 2003
16.184 dma_space . . . 2005
16.185 l4/re/elf_aux.h File Reference . . . 2006
16.185.1 Detailed Description . . . 2007
16.186 elf_aux.h . . . 2007
16.187 l4/re/env File Reference . . . 2008
16.187.1 Detailed Description . . . 2009
16.188 env . . . 2009
16.189 l4/re/env.h File Reference . . . 2010

16.189.1 Detailed Description . . . 2012
16.189.2 Typedef Documentation . . . 2012
16.189.2.1 l4re_env_t . . . 2012
16.190 env.h . . . 2012
16.191 l4/re/error_helper File Reference . . . 2014
16.191.1 Detailed Description . . . 2015
16.192 error_helper . . . 2016
16.193 l4/re/util/event File Reference . . . 2017
16.194 event . . . 2018
16.195 l4/re/impl/dataspace_impl.h File Reference . . . 2019
16.195.1 Detailed Description . . . 2020
16.196 dataspace_impl.h . . . 2020
16.197 l4/re/impl/mem_alloc_impl.h File Reference . . . 2022
16.197.1 Detailed Description . . . 2023
16.198 mem_alloc_impl.h . . . 2023
16.199 l4/re/impl/namespace_impl.h File Reference . . . 2023
16.199.1 Detailed Description . . . 2024
16.200 namespace_impl.h . . . 2024
16.201 l4/re/impl/rm_impl.h File Reference . . . 2026
16.201.1 Detailed Description . . . 2027
16.202 rm_impl.h . . . 2027
16.203 l4/re/l4aux.h File Reference . . . 2028
16.203.1 Detailed Description . . . 2029
16.204 l4aux.h . . . 2029
16.205 l4/re/log File Reference . . . 2030
16.205.1 Detailed Description . . . 2031
16.206 log . . . 2031
16.207 l4/re/log-sys.h File Reference . . . 2031
16.207.1 Detailed Description . . . 2031
16.208 log-sys.h . . . 2032
16.209 l4/re/mem_alloc File Reference . . . 2032
16.209.1 Detailed Description . . . 2033
16.210 mem_alloc . . . 2033
16.211 l4/re/mem_alloc-sys.h File Reference . . . 2034
16.211.1 Detailed Description . . . 2034
16.212 mem_alloc-sys.h . . . 2035
16.213 l4/re/mmio_space File Reference . . . 2035
16.213.1 Detailed Description . . . 2036
16.214 mmio_space . . . 2037
16.215 l4/re/namespace File Reference . . . 2037
16.215.1 Detailed Description . . . 2039
16.216 namespace . . . 2039

16.217 l4/re/namespace-sys.h File Reference . . . 2040
16.217.1 Detailed Description . . . 2040
16.218 namespace-sys.h . . . 2041
16.219 l4/re/parent File Reference . . . 2041
16.219.1 Detailed Description . . . 2043
16.220 parent . . . 2043
16.221 l4/re/parent-sys.h File Reference . . . 2043
16.221.1 Detailed Description . . . 2044
16.222 parent-sys.h . . . 2044
16.223 l4/re/protocols.h File Reference . . . 2044
16.223.1 Detailed Description . . . 2044
16.223.2 Enumeration Type Documentation . . . 2045
16.223.2.1 L4re_protocols . . . 2045
16.224 protocols.h . . . 2045
16.225 l4/re/random File Reference . . . 2046
16.225.1 Detailed Description . . . 2047
16.226 random . . . 2047
16.227 l4/re/rm File Reference . . . 2047
16.227.1 Detailed Description . . . 2048
16.228 rm . . . 2049
16.229 l4/re/rm-sys.h File Reference . . . 2053
16.229.1 Detailed Description . . . 2053
16.230 rm-sys.h . . . 2053
16.231 l4/re/shared_cap File Reference . . . 2054
16.231.1 Detailed Description . . . 2055
16.232 shared_cap . . . 2055
16.233 l4/re/util/shared_cap File Reference . . . 2056
16.233.1 Detailed Description . . . 2058
16.234 shared_cap . . . 2058
16.235 l4/re/unique_cap File Reference . . . 2059
16.235.1 Detailed Description . . . 2060
16.236 unique_cap . . . 2060
16.237 l4/re/util/unique_cap File Reference . . . 2061
16.237.1 Detailed Description . . . 2063
16.238 unique_cap . . . 2063
16.239 l4/re/util/bitmap_cap_alloc File Reference . . . 2063
16.239.1 Detailed Description . . . 2064
16.240 bitmap_cap_alloc . . . 2065
16.241 l4/re/util/cap File Reference . . . 2066
16.241.1 Detailed Description . . . 2067
16.242 cap . . . 2067
16.243 l4/re/util/cap_alloc_impl.h File Reference . . . 2067

16.243.1 Detailed Description . . . 2068
16.244 cap_alloc_impl.h . . . 2069
16.245 l4/re/util/counting_cap_alloc File Reference . . . 2069
16.245.1 Detailed Description . . . 2071
16.246 counting_cap_alloc . . . 2071
16.247 l4/re/util/item_alloc File Reference . . . 2073
16.247.1 Detailed Description . . . 2074
16.248 item_alloc . . . 2074
16.249 l4/re/util/kumem_alloc File Reference . . . 2075
16.249.1 Detailed Description . . . 2076
16.250 kumem_alloc . . . 2076
16.251 l4/re/util/region_mapping File Reference . . . 2076
16.251.1 Detailed Description . . . 2077
16.252 region_mapping . . . 2078
16.253 l4/re/video/goos-sys.h File Reference . . . 2083
16.253.1 Detailed Description . . . 2083
16.254 goos-sys.h . . . 2083
16.255 l4/shmc/shmc.h File Reference . . . 2084
16.255.1 Detailed Description . . . 2086
16.256 shmc.h . . . 2086
16.257 l4/sigma0/sigma0.h File Reference . . . 2088
16.257.1 Detailed Description . . . 2090
16.258 sigma0.h . . . 2091
16.259 l4/sys/__kernel_object_impl.h File Reference . . . 2092
16.259.1 Detailed Description . . . 2092
16.260 __kernel_object_impl.h . . . 2093
16.261 l4/sys/__ktrace-impl.h File Reference . . . 2093
16.261.1 Detailed Description . . . 2094
16.262 __ktrace-impl.h . . . 2094
16.263 l4/sys/__typeinfo.h File Reference . . . 2095
16.263.1 Detailed Description . . . 2098
16.264 __typeinfo.h . . . 2098
16.265 l4/sys/__vm-arm.h File Reference . . . 2108
16.265.1 Detailed Description . . . 2109
16.266 __vm-arm.h . . . 2109
16.267 l4/sys/arm_smccc File Reference . . . 2110
16.267.1 Detailed Description . . . 2111
16.268 arm_smccc . . . 2111
16.269 l4/sys/arm_smccc.h File Reference . . . 2111
16.269.1 Detailed Description . . . 2112
16.269.2 Function Documentation . . . 2112
16.269.2.1 l4_arm_smccc_call() . . . 2112

16.270 arm_smccc.h . . . 2113
16.271 l4/sys/cache.h File Reference . . . 2114
16.271.1 Detailed Description . . . 2115
16.272 cache.h . . . 2116
16.273 arm/l4/sys/cache.h File Reference . . . 2116
16.273.1 Detailed Description . . . 2117
16.274 cache.h . . . 2118
16.275 amd64/l4/sys/cache.h File Reference . . . 2119
16.275.1 Detailed Description . . . 2119
16.276 cache.h . . . 2120
16.277 x86/l4/sys/cache.h File Reference . . . 2120
16.277.1 Detailed Description . . . 2121
16.278 cache.h . . . 2121
16.279 l4/sys/capability File Reference . . . 2122
16.279.1 Detailed Description . . . 2123
16.279.2 Macro Definition Documentation . . . 2123
16.279.2.1 L4_DISABLE_COPY . . . 2123
16.280 capability . . . 2124
16.281 l4/sys/compiler.h File Reference . . . 2125
16.281.1 Detailed Description . . . 2126
16.282 compiler.h . . . 2127
16.283 l4/sys/consts.h File Reference . . . 2129
16.283.1 Detailed Description . . . 2131
16.284 consts.h . . . 2131
16.285 arm/l4/sys/consts.h File Reference . . . 2133
16.285.1 Detailed Description . . . 2133
16.286 consts.h . . . 2133
16.287 amd64/l4/sys/consts.h File Reference . . . 2134
16.287.1 Detailed Description . . . 2134
16.288 consts.h . . . 2134
16.289 x86/l4/sys/consts.h File Reference . . . 2135
16.289.1 Detailed Description . . . 2135
16.290 consts.h . . . 2135
16.291 l4/re/consts.h File Reference . . . 2135
16.291.1 Detailed Description . . . 2136
16.291.2 Enumeration Type Documentation . . . 2136
16.291.2.1 anonymous enum . . . 2137
16.292 consts.h . . . 2137
16.293 l4/re/consts File Reference . . . 2137
16.293.1 Detailed Description . . . 2138
16.294 consts . . . 2139
16.295 l4/sys/cxx/ipc_client File Reference . . . 2139

16.295.1 Macro Definition Documentation . . . 2141
16.295.1.1 L4_RPC_DEF . . . 2141
16.296 ipc_client . . . 2141
16.297 l4/sys/cxx/ipc_iface File Reference . . . 2142
16.297.1 Detailed Description . . . 2144
16.297.2 Macro Definition Documentation . . . 2144
16.297.2.1 L4_INLINE_RPC . . . 2144
16.297.2.2 L4_INLINE_RPC_NF . . . 2145
16.297.2.3 L4_INLINE_RPC_NF_OP . . . 2145
16.297.2.4 L4_INLINE_RPC_OP . . . 2146
16.297.2.5 L4_RPC . . . 2146
16.297.2.6 L4_RPC_NF . . . 2147
16.297.2.7 L4_RPC_NF_OP . . . 2147
16.297.2.8 L4_RPC_OP . . . 2148
16.298 ipc_iface . . . 2148
16.299 l4/cxx/ipc_server File Reference . . . 2153
16.299.1 Detailed Description . . . 2154
16.300 ipc_server . . . 2155
16.301 l4/sys/cxx/ipc_types File Reference . . . 2156
16.302 ipc_types . . . 2158
16.303 l4/sys/cxx/smart_capability_1x File Reference . . . 2164
16.304 smart_capability_1x . . . 2165
16.305 l4/sys/cxx/types File Reference . . . 2168
16.305.1 Macro Definition Documentation . . . 2169
16.305.1.1 L4_TYPES_FLAGS_OPS_DEF . . . 2169
16.306 types . . . 2169
16.307 l4/sys/debugger File Reference . . . 2172
16.307.1 Detailed Description . . . 2173
16.308 debugger . . . 2173
16.309 l4/sys/debugger.h File Reference . . . 2174
16.309.1 Detailed Description . . . 2175
16.309.2 Function Documentation . . . 2175
16.309.2.1 __strcpy_maxlen() . . . 2175
16.309.2.2 l4_debugger_get_object_name() . . . 2176
16.309.2.3 l4_debugger_query_log_name() . . . 2176
16.309.2.4 l4_debugger_query_log_typeid() . . . 2177
16.309.2.5 l4_debugger_switch_log() . . . 2177
16.310 debugger.h . . . 2178
16.311 l4/sys/err.h File Reference . . . 2181
16.311.1 Detailed Description . . . 2182
16.312 err.h . . . 2182
16.313 l4/sys/exception File Reference . . . 2182

16.313.1 Detailed Description . . . 2183
16.314 exception . . . 2184
16.315 l4/sys/factory File Reference . . . 2184
16.315.1 Detailed Description . . . 2186
16.316 factory . . . 2186
16.317 l4/sys/factory.h File Reference . . . 2188
16.317.1 Detailed Description . . . 2189
16.318 factory.h . . . 2189
16.319 l4/sys/icu File Reference . . . 2193
16.319.1 Detailed Description . . . 2194
16.320 icu . . . 2194
16.321 l4/sys/icu.h File Reference . . . 2195
16.321.1 Detailed Description . . . 2197
16.322 icu.h . . . 2198
16.323 l4/sys/ipc.h File Reference . . . 2201
16.323.1 Detailed Description . . . 2203
16.323.2 Function Documentation . . . 2203
16.323.2.1 l4_ipc_to_errno() . . . 2203
16.324 ipc.h . . . 2203
16.325 arm/l4f/l4/sys/ipc.h File Reference . . . 2206
16.325.1 Detailed Description . . . 2207
16.326 ipc.h . . . 2207
16.327 x86/l4f/l4/sys/ipc.h File Reference . . . 2208
16.327.1 Detailed Description . . . 2209
16.328 ipc.h . . . 2209
16.329 l4/sys/ipc_gate File Reference . . . 2209
16.329.1 Detailed Description . . . 2211
16.330 ipc_gate . . . 2211
16.331 l4/sys/ipc_gate.h File Reference . . . 2211
16.331.1 Detailed Description . . . 2213
16.332 ipc_gate.h . . . 2214
16.333 l4/sys/irq File Reference . . . 2215
16.333.1 Detailed Description . . . 2216
16.334 irq . . . 2216
16.335 l4/sys/kernel_object.h File Reference . . . 2218
16.335.1 Detailed Description . . . 2219
16.336 kernel_object.h . . . 2219
16.337 l4/sys/kip File Reference . . . 2220
16.337.1 Detailed Description . . . 2221
16.338 kip . . . 2222
16.339 l4/sys/ktrace.h File Reference . . . 2223
16.339.1 Detailed Description . . . 2224

16.340 ktrace.h . . . 2225
16.341 l4/sys/l4int.h File Reference . . . 2225
16.341.1 Detailed Description . . . 2226
16.342 l4int.h . . . 2226
16.343 arm/l4/sys/l4int.h File Reference . . . 2227
16.343.1 Detailed Description . . . 2227
16.344 l4int.h . . . 2227
16.345 amd64/l4/sys/l4int.h File Reference . . . 2227
16.345.1 Detailed Description . . . 2228
16.346 l4int.h . . . 2228
16.347 x86/l4/sys/l4int.h File Reference . . . 2228
16.347.1 Detailed Description . . . 2229
16.348 l4int.h . . . 2229
16.349 l4/sys/memdesc.h File Reference . . . 2229
16.349.1 Detailed Description . . . 2231
16.350 memdesc.h . . . 2231
16.351 l4/sys/meta File Reference . . . 2233
16.351.1 Detailed Description . . . 2235
16.352 meta . . . 2235
16.353 l4/sys/pager File Reference . . . 2235
16.353.1 Detailed Description . . . 2237
16.354 pager . . . 2237
16.355 l4/sys/platform_control File Reference . . . 2237
16.355.1 Detailed Description . . . 2238
16.356 platform_control . . . 2239
16.357 l4/sys/platform_control.h File Reference . . . 2239
16.357.1 Detailed Description . . . 2241
16.358 platform_control.h . . . 2241
16.359 l4/sys/rcv_endpoint File Reference . . . 2243
16.359.1 Detailed Description . . . 2245
16.360 rcv_endpoint . . . 2245
16.361 l4/sys/rcv_endpoint.h File Reference . . . 2245
16.361.1 Detailed Description . . . 2246
16.361.2 Enumeration Type Documentation . . . 2247
16.361.2.1 L4_rcv_ep_ops . . . 2247
16.362 rcv_endpoint.h . . . 2247
16.363 l4/sys/scheduler File Reference . . . 2248
16.363.1 Detailed Description . . . 2249
16.364 scheduler . . . 2249
16.365 l4/sys/scheduler.h File Reference . . . 2249
16.365.1 Detailed Description . . . 2251
16.366 scheduler.h . . . 2252

16.367 l4/sys/semaphore File Reference . . . 2254
16.367.1 Detailed Description . . . 2256
16.368 semaphore . . . 2256
16.369 l4/sys/semaphore.h File Reference . . . 2256
16.369.1 Detailed Description . . . 2258
16.370 semaphore.h . . . 2258
16.371 l4/sys/smart_capability File Reference . . . 2259
16.371.1 Detailed Description . . . 2260
16.372 smart_capability . . . 2260
16.373 l4/sys/task File Reference . . . 2262
16.373.1 Detailed Description . . . 2264
16.374 task . . . 2264
16.375 l4/sys/task.h File Reference . . . 2265
16.375.1 Detailed Description . . . 2266
16.376 task.h . . . 2266
16.377 l4/sys/thread File Reference . . . 2269
16.377.1 Detailed Description . . . 2270
16.378 thread . . . 2271
16.379 l4/cxx/thread File Reference . . . 2272
16.379.1 Detailed Description . . . 2273
16.380 thread . . . 2274
16.381 l4/sys/typeinfo_svr File Reference . . . 2274
16.381.1 Detailed Description . . . 2276
16.382 typeinfo_svr . . . 2276
16.383 l4/sys/types.h File Reference . . . 2277
16.383.1 Detailed Description . . . 2279
16.383.2 Function Documentation . . . 2279
16.383.2.1 l4_capability_next() . . . 2279
16.384 types.h . . . 2279
16.385 l4/sys/utcb.h File Reference . . . 2282
16.385.1 Detailed Description . . . 2284
16.386 utcb.h . . . 2284
16.387 arm/l4/sys/utcb.h File Reference . . . 2286
16.387.1 Detailed Description . . . 2287
16.388 utcb.h . . . 2288
16.389 amd64/l4/sys/utcb.h File Reference . . . 2289
16.389.1 Detailed Description . . . 2290
16.390 utcb.h . . . 2290
16.391 x86/l4/sys/utcb.h File Reference . . . 2292
16.391.1 Detailed Description . . . 2293
16.392 utcb.h . . . 2293
16.393 l4/sys/vcon File Reference . . . 2294

16.393.1 Detailed Description . . . 2296
16.394 vcon . . . 2296
16.395 l4/sys/vcon.h File Reference . . . 2296
16.395.1 Detailed Description . . . 2299
16.395.2 Enumeration Type Documentation . . . 2299
16.395.2.1 L4_vcon_read_flags . . . 2299
16.396 vcon.h . . . 2299
16.397 l4/sys/vhw.h File Reference . . . 2302
16.397.1 Detailed Description . . . 2303
16.398 vhw.h . . . 2304
16.399 l4/sys/vm File Reference . . . 2305
16.399.1 Detailed Description . . . 2306
16.400 vm . . . 2306
16.401 l4/util/assert.h File Reference . . . 2306
16.401.1 Detailed Description . . . 2307
16.402 assert.h . . . 2307
16.403 l4/sys/assert.h File Reference . . . 2308
16.403.1 Detailed Description . . . 2310
16.403.2 Macro Definition Documentation . . . 2310
16.403.2.1 l4_assert . . . 2310
16.404 assert.h . . . 2310
16.405 l4/util/atomic.h File Reference . . . 2311
16.405.1 Detailed Description . . . 2313
16.406 atomic.h . . . 2314
16.407 arm/l4/sys/atomic.h File Reference . . . 2318
16.407.1 Detailed Description . . . 2318
16.408 atomic.h . . . 2319
16.409 l4/cxx/atomic.h File Reference . . . 2319
16.409.1 Detailed Description . . . 2320
16.410 atomic.h . . . 2320
16.411 l4/util/backtrace.h File Reference . . . 2320
16.411.1 Detailed Description . . . 2321
16.411.2 Function Documentation . . . 2321
16.411.2.1 l4util_backtrace() . . . 2322
16.412 backtrace.h . . . 2322
16.413 l4/util/base64.h File Reference . . . 2322
16.413.1 Detailed Description . . . 2323
16.414 base64.h . . . 2324
16.415 l4/util/bitops.h File Reference . . . 2324
16.415.1 Detailed Description . . . 2326
16.416 bitops.h . . . 2326
16.417 l4/util/elf.h File Reference . . . 2329

16.417.1 Detailed Description . . . 2335
16.418 elf.h . . . 2335
16.419 l4/util/getopt.h File Reference . . . 2345
16.419.1 Detailed Description . . . 2346
16.420 getopt.h . . . 2346
16.421 l4/util/keymap.h File Reference . . . 2347
16.421.1 Detailed Description . . . 2348
16.422 keymap.h . . . 2348
16.423 l4/util/kip.h File Reference . . . 2348
16.424 kip.h . . . 2349
16.425 l4/sys/kip.h File Reference . . . 2350
16.425.1 Detailed Description . . . 2351
16.426 kip.h . . . 2351
16.427 l4/util/kprintf.h File Reference . . . 2353
16.427.1 Detailed Description . . . 2353
16.428 kprintf.h . . . 2353
16.429 l4/util/list_alloc.h File Reference . . . 2354
16.429.1 Detailed Description . . . 2354
16.429.2 Function Documentation . . . 2355
16.429.2.1 l4la_alloc() . . . 2355
16.429.2.2 l4la_avail() . . . 2355
16.429.2.3 l4la_dump() . . . 2355
16.429.2.4 l4la_free() . . . 2356
16.429.2.5 l4la_init() . . . 2356
16.430 list_alloc.h . . . 2356
16.431 l4/util/lock.h File Reference . . . 2357
16.431.1 Detailed Description . . . 2357
16.432 lock.h . . . 2358
16.433 l4/util/mb_info.h File Reference . . . 2358
16.433.1 Detailed Description . . . 2361
16.433.2 Macro Definition Documentation . . . 2361
16.433.2.1 l4util_mb_for_each_mmap_entry . . . 2361
16.433.2.2 L4UTIL_MB_MEMORY . . . 2361
16.433.2.3 MB_ARD_MEMORY . . . 2361
16.433.2.4 MB_ART_MEMORY . . . 2362
16.434 mb_info.h . . . 2362
16.435 l4/util/parse_cmd.h File Reference . . . 2366
16.435.1 Detailed Description . . . 2367
16.436 parse_cmd.h . . . 2367
16.437 l4/util/rand.h File Reference . . . 2367
16.437.1 Detailed Description . . . 2368
16.438 rand.h . . . 2369

16.439 l4/util/splitlog2.h File Reference . . . 2369
16.439.1 Detailed Description . . . 2370
16.440 splitlog2.h . . . 2370
16.441 l4/util/thread.h File Reference . . . 2371
16.441.1 Detailed Description . . . 2371
16.441.2 Macro Definition Documentation . . . 2372
16.441.2.1 __L4UTIL_THREAD_FUNC . . . 2372
16.442 thread.h . . . 2372
16.443 l4/sys/thread.h File Reference . . . 2373
16.443.1 Detailed Description . . . 2375
16.444 thread.h . . . 2375
16.445 arm/l4/sys/thread.h File Reference . . . 2382
16.445.1 Detailed Description . . . 2382
16.446 thread.h . . . 2382
16.447 l4/vbus/vbus.h File Reference . . . 2383
16.447.1 Detailed Description . . . 2385
16.447.2 Enumeration Type Documentation . . . 2385
16.447.2.1 anonymous enum . . . 2385
16.447.2.2 l4vbus_icu_src_types . . . 2385
16.448 vbus.h . . . 2386
16.449 l4/vbus/vbus_interfaces.h File Reference . . . 2387
16.449.1 Detailed Description . . . 2388
16.449.2 Typedef Documentation . . . 2388
16.449.2.1 l4vbus_iface_type_t . . . 2388
16.449.3 Enumeration Type Documentation . . . 2388
16.449.3.1 anonymous enum . . . 2388
16.449.3.2 l4vbus_iface_type_t . . . 2389
16.449.4 Function Documentation . . . 2389
16.449.4.1 l4vbus_subinterface_supported() . . . 2389
16.450 vbus_interfaces.h . . . 2389
16.451 l4/vbus/vbus_types.h File Reference . . . 2390
16.451.1 Detailed Description . . . 2391
16.451.2 Enumeration Type Documentation . . . 2391
16.451.2.1 l4vbus_device_flags_t . . . 2391
16.451.2.2 l4vbus_resource_flags_t . . . 2392
16.451.2.3 l4vbus_resource_type_t . . . 2392
16.452 vbus_types.h . . . 2392

17 Example Documentation **2395**
17.1 hello/server/src/main.c . . . 2395
17.2 examples/sys/ipc/ipc_example.c . . . 2395
17.3 examples/sys/ipc/ipc.cfg . . . 2396

17.4 examples/sys/start-with-exc/main.c . . . 2397
17.5 examples/sys/singlestep/main.c . . . 2398
17.6 examples/sys/aliens/main.c . . . 2400
17.7 examples/sys/utcb-ipc/main.c . . . 2403
17.8 examples/sys/ux-vhw/main.c . . . 2405
17.9 examples/sys/isr/main.c . . . 2405
17.10 examples/clntsrv/server.cc . . . 2406
17.11 examples/clntsrv/client.cc . . . 2407
17.12 examples/clntsrv/clntsrv.cfg . . . 2407
17.13 examples/libs/l4re/c/ma+rm.c . . . 2408
17.14 examples/libs/l4re/c++/mem_alloc/ma+rm.cc . . . 2409
17.15 examples/libs/l4re/c++/shared_ds/ds_clnt.cc . . . 2410
17.16 examples/libs/l4re/c++/shared_ds/ds_srv.cc . . . 2411
17.17 examples/libs/l4re/c++/shared_ds/shared_ds.cfg . . . 2412
17.18 examples/libs/l4re/streammap/server.cc . . . 2413
17.19 examples/libs/l4re/streammap/client.cc . . . 2414
17.20 examples/libs/l4re/streammap/streammap.cfg . . . 2414
17.21 examples/libs/libirq/loop.c . . . 2415
17.22 examples/libs/libirq/async_isr.c . . . 2415
17.23 examples/sys/migrate/thread_migrate.cc . . . 2416
17.24 examples/sys/migrate/thread_migrate.cfg . . . 2417
17.25 tmpfs/lib/src/fs.cc . . . 2417
17.26 examples/libs/shmc/prodcons.c . . . 2423

Index **2425**

Chapter 1

Overview

Welcome to the documentation of the L4 Runtime Environment, or L4Re for short. There are two parts to this documentation: a user manual, which provides a birds eye view of L4Re and its environment, and a reference section which documents the complete programming API.

User Manual

1. Introduction shortly explains the concept of microkernels and introduces the basic terminology.
2. Tutorial helps you getting started with setting up the development environment and writing your own first L4Re application.
3. Programming for L4Re explains in detail the most important programming concepts.
4. L4Re Servers provides a quick overview over standard services running on the L4Re operating system.

Reference

The second part provides the complete reference of all classes and functions of the L4Re environment as well as a list of example code.

Chapter 2

Introduction

The intention of this section is to provide a short overview about the L4Re environment.

The general structure of a microkernel-based system will be introduced and the principal functionality of the servers in the basic environment outlined.

2.1 Fiasco.OC

The Fiasco.OC microkernel is the lowest-level piece of software running in an L4-based system. The microkernel is the only program that runs in privileged processor mode. It does not include complex services such as program loading, device drivers, or file systems; those are implemented in user-level programs on top of it (a basic set of these services and abstractions is provided by the L4 Runtime Environment).

Fiasco.OC kernel services are implemented in kernel objects. Tasks hold references to kernel objects in their respective *"object space"*, which is a kernel-protected table. These references are called *capabilities*. Fiasco system calls are function invocations on kernel objects through the corresponding capabilities. These can be thought of as function invocations on object references in an object-oriented programming environment. Furthermore, if a task owns a capability, it may grant other tasks the same (or fewer) rights on this object by passing the capability from its own to the other task's object space.

From a design perspective, capabilities are a concept that enables flexibility in the system structure. A thread that invokes an object through a capability does not need to care about where this object is implemented. In fact, it is possible to implement all objects either in the kernel or in a user-level server and replace one implementation with the other transparently for clients.

2.1.1 Communication

The basic communication mechanism in L4-based systems is called *"Inter Process Communication (IPC)"*. It is always synchronous, i.e. both communication partners need to actively rendezvous for IPC. In addition to transmitting arbitrary data between threads, IPC is also used to resolve hardware exceptions, faults and for virtual memory management.

2.1.2 Kernel Objects

The following list gives a short overview of the kernel objects provided by the Fiasco.OC microkernel:

- **Task** A task comprises a memory address space (represented by the task's page table), an object space (holding the kernel protected capabilities), and on x86 an IO-port address space.
- **Thread** A thread is bound to a task and executes code. Multiple threads can coexist in one task and are scheduled by the Fiasco scheduler.
- **Factory** A factory is used by applications to create new kernel objects. Access to a factory is required to create any new kernel object. Factories can control and restrict object creation.
- **IPC Gate** An IPC gate is used to create a secure communication channel between different tasks. It embeds a label (kernel protected payload) that securely identifies the gate through which a message is received. The gate label is not visible to and cannot be altered by the sender.
- **IRQ** IRQ objects provide access to hardware interrupts. Additionally, programs can create new virtual interrupt objects and trigger them. This allows to implement a signaling mechanism. The receiver cannot decide whether the interrupt is a physical or virtual one.
- **Vcon** Provides access to the in-kernel debugging console (input and output). There is only one such object in the kernel and it is only available, if the kernel is built with debugging enabled. This object is typically interposed through a user-level service or without debugging in the kernel can be completely based on user-level services.
- **Scheduler** Implements scheduling policy and assignment of threads to CPUs, including CPU statistics.

2.2 L4Re System Structure

The system has a multi-tier architecture consisting of the following layers depicted in the figure below:

- **Microkernel** The microkernel is the component at the lowest level of the software stack. It is the only piece of software that is running in the privileged mode of the processor.

- **Tasks** Tasks are the basic containers (address spaces) in which system services and applications are executed. They run in the processor's deprivileged user mode.

Figure 2.1 Basic Structure of an L4Re based system

In terms of functionality, the system is structured as follows:

- **Microkernel** The kernel provides primitives to execute programs in tasks, to enforce isolation among them, and to provide means of secure communication in order to let them cooperate. As the kernel is the most privileged, security-critical software component in the system, it is a general design goal to make it as small as possible in order to reduce its attack surface. It provides only a minimal set of mechanisms that are necessary to support applications.

- **Runtime Environment** The small kernel offers a concise set of interfaces, but these are not necessarily suited for building applications directly on top of it. The L4 Runtime Environment aims at providing more convenient abstractions for application development. It comprises low-level software components that interface directly with the microkernel. The root pager *sigma0* and the root task *Moe* are the most basic components of the runtime environment. Other services (e.g., for device enumeration) use interfaces provided by them.

- **Applications** Applications run on top of the system and use services provided by the Runtime Environment – or by other applications. There may be several types of applications in the system and even virtual machine monitors and device drivers are considered applications in the terminology used in this document. They are running alongside other applications on the system.

Lending terminology from the distributed systems area, applications offering services to other applications are usually called *servers*, whereas applications using those services are named *clients*. Being in both roles is also common, for instance, a file system server may be viewed as a server with respect to clients using the file system, while the server itself may also act as a client of a hard disk driver.

2.3 L4 Runtime Environment (L4Re)

The L4 Runtime Environment (L4Re) provides a basic set of services and abstractions, which are useful to implement and run user-level applications on top of the Fiasco.OC microkernel.

L4Re consists of a set of libraries and servers. Libraries as well as server interfaces are completely object oriented. They implement prototype implementations for the classes defined by the L4Re specification.

A minimal L4Re-based application needs 3 components to be booted beforehand: the Fiasco microkernel, the root pager (Sigma0), and the root task (Moe). The Sigma0 root pager initially owns all system resources, but is usually used only to resolve page faults for the Moe root task. Moe provides the essential services to normal user applications such as an initial program loader, a region-map service for virtual memory management, and a memory (data space) allocator.

Chapter 3

Tutorial

This tutorial assumes that the reader is familiar with the basic L4 concepts that were discussed in the Introduction section and has a working knowledge of C++.

Here you can find the first steps to boot a very simple setup. The setup consists of the following components:

- Fiasco.OC — Microkernel
- Sigma0 — Root Pager
- Moe — Root Task
- Ned — Init Process
- hello — Hello World Application

The guide assumes that you already compiled the base components and describes how to generate an ISO image, with GRUB 1 or GRUB 2 as a boot loader, that can for example be booted within QEMU.

First you need a `modules.list` file that contains an entry for the scenario.

```
modaddr  0x002000000
entry hello
  kernel   fiasco -serial_esc
  roottask moe rom/hello.cfg
  module   l4re
  module   ned
  module   hello.cfg
  module   hello
```

This file describes all the binaries and scripts to put into the ISO image, and also describes the GRUB `menu.lst` contents. What you need to do is to set the `make` variable `MODULE_SEARCH_PATH` to contain the path to your Fiasco.OC build directory and the directory containing your `hello.cfg` script.

The `hello.cfg` script should look like the following. A ready to use version can be found in l4/conf/examples.

```
local L4 = require("L4");
L4.default_loader:start({}, "rom/hello");
```

The first line of this script ensures that the `L4` package is available for the script. The second line uses the default loader object defined in that package and starts the binary `rom/hello`.

Note

All modules defined in `modules.list` are available as data spaces (L4Re::Dataspace) and registered in a name space (L4Re::Namespace). This name space is in turn available as 'rom' to the init process (Ned).

Now you can go to your L4Re build directory and run the following command.

Note

The example assumes that you have created the `modules.list` and `hello.cfg` files in the /tmp directory. Adapt if you created them somewhere else.

```
make grub1iso E=hello MODULES_LIST=/tmp/modules.list MODULE_SEARCH_PATH=/tmp:<path_to_fiasco_builddir>
```

Or as an alternative use GRUB 2:

```
make grub2iso E=hello MODULES_LIST=/tmp/modules.list MODULE_SEARCH_PATH=/tmp:<path_to_fiasco_builddir>
```

Now you should be able to boot the image in QEMU by running:

```
qemu-system-i386 -cdrom images/hello.iso -serial stdio
```

If you press <ESC> in the terminal that shows you the serial output you enter the Fiasco.OC kernel debugger... Have fun.

3.1 Customizations

A basic set of bootable entries can be found in `l4/conf/modules.list`. This file is the default for any image creation as shown above. It is recommended that local modification regarding image creation are done in `conf/Makeconf.boot`. Initially you may copy `Makeconf.boot.example` to `Makeconf.boot`. You can overwrite `MODULES_LIST` to set your own modules-list file. Set `MODULE_SEARCH_PATH` to your setup according to the examples given in the file. When configured a `make` call is reduced to:

```
make grub2iso E=hello
```

All other local configuration can be done in a `Makeconf.local` file located in the `l4` directory.

Chapter 4

Programming for L4Re

This part of the documentation discusses the concept of microkernel-based programming in more detail.

You should already have a basic understanding of the L4Re programming environment from the tutorial.

- Capabilities and Naming
- Spaces and Mappings
- Initial Environment and Application Bootstrapping
- Memory management - Data Spaces and the Region Map
- Program Input and Output
- Initial Memory Allocator and Factory
- Application and Server Building Blocks
- Pthread Support
- tasks and threads
- communication channels
- server loops
- Interface Definition Language
- hardware access
- L4Re Build System
- Kernel Factory

4.1 Capabilities and Naming

The L4Re system is a capability based system which uses and offers capabilities to implement fine-grained access control.

Generally, owning a capability means to be allowed to communicate with the object the capability points to. All user-visible kernel objects, such as tasks, threads, and IRQs, can only be accessed through a capability. Please refer to the Kernel Objects documentation for details. Capabilities are stored in per-task capability tables (the object space) and are referenced by capability selectors or object flex pages. In a simplified view, a capability selector is a natural number indexing into the capability table of the current task.

As a matter of fact, a system designed solely based on capabilities uses so-called 'local names' because each task can only access those objects made available to this task. Other objects are not visible to and accessible by the task.

Figure 4.1 Capabilities and Local Naming in L4

So how does an application get access to a service? In general all applications are started with an initial set of available objects. This set of objects is predetermined by the creator of a new application process and granted directly to the new task before starting the first application thread. The application can then use these initial objects to request access to further objects or to transfer capabilities to its own objects to other applications. A central L4Re object for exchanging capabilities at runtime is the name-space object, implementing a store of named capabilities.

From a security perspective, the set of initial capabilities (access rights to objects) completely define the execution environment of an application. Mandatory security policies can be defined by well known properties of the initial objects and carefully handled access rights to them.

4.2 Spaces and Mappings

Each task in the L4Re system has access to two resource spaces (three on IA32) which are maintained by the kernel.

These are the

1. object space,
2. memory space, and
3. IO-port space (only on IA32).

The entities addressed in each space are capabilities to objects, virtual memory pages, and IO ports. The addresses are unsigned integers and the largest valid address depends on which space is referenced, the hardware, and the configuration of the kernel. Although a program can access memory at byte granularity, from the kernel's point of view the address granularity in the memory space is not bytes but pages, as determined by the hardware. The address of a capability is also called its "capability slot".

Flexpages describe a range in any of the spaces that has a power-of-two length and is also aligned to this length. They additionally hold access rights information and further space specific information.

When a resource is present at some address in a task's corresponding resource space, then we say that resource is mapped to that task. For example, a capability to the task's main thread may be mapped to capability slot 5, or the first page of the code segment a thread executes is mapped to virtual memory page 12345. However, there need not be any resource mapped to an address.

Tasks can exchange resources through a process called "mapping" during IPC and using the L4::Task::map() method. The sending task specifies a send flexpage and the receiving task a receive flexpage. The resources mapped to the send flexpage will then be mapped to the receive flexpage by the kernel.

4.3 Initial Environment and Application Bootstrapping

New applications that are started by a loader conforming to L4Re get provided an Initial Environment.

This environment comprises a set of capabilities to initial L4Re objects that are required to bootstrap and run this application. These capabilities include:

- A capability to an initial memory allocator for obtaining memory in the form of data spaces
- A capability to a factory which can be used to create additional kernel objects
- A capability to a Vcon object for debugging output and maybe input
- A set of named capabilities to application specific objects

During the bootstrapping of the application, the loader establishes data spaces for each individual region in the ELF binary. These include data spaces for the code and data sections, and a data space backed with RAM for the stack of the program's first thread.

One loader implementation is the `moe` root task. Moe usually starts an *init* process that is responsible for coordinating the further boot process. The default *init* process is `ned`, which implements a script-based configuration and startup of other processes. Ned uses Lua (`http://www.lua.org`) as its scripting language, see Ned Script example for more details.

4.3.1 Configuring an application before startup

The default L4Re init process (Ned) provides a Lua script based configuration of initial capabilities and application startup. Ned itself also has a set of initial objects available that can be used to create the environment for an application. The most important object is a kernel object factory that allows creation of kernel objects such as IPC gates (communication channels), tasks, threads, etc. Ned uses Lua tables (associative arrays) to represent sets of capabilities that shall be granted to application processes.

```
local caps = {
    name = some_capability
}
```

The L4 Lua package in Ned also has support functions to create application tasks, region-map objects, etc. to start an ELF binary in a new task. The package also contains Lua bindings for basic L4Re objects, for example, to generic factory objects, which are used to create kernel objects and also user-level objects provided by user-level servers.

```
L4.default_loader:start({ caps = { some_service = service } }, "rom/program --arg");
```

4.3.2 Connecting clients and servers

In general, a connection between a client and a server is represented by a communication channel (IPC gate) that is available to both of them. You can see the simplest connection between a client and a server in the following example.

```
local loader = L4.default_loader; -- which is Moe
local svc = loader:new_channel();  -- create an IPC gate
loader:start({ caps = { service = svc:svr() }}, "rom/my_server");
loader:start({ caps = { service = svc:m("rw") }}, "rom/my_client");
```

As you can see in the snippet, the first action is to create a new channel (IPC gate) using `loader:new_channel()`. The capability to the gate is stored in the variable `svc`. Then the binary `my_server` is started in a new task, and full (`:svr()`) access to the IPC gate is granted to the server as initial object. The gate is accessible to the server application as "service" in the set of its initial capabilities. Virtually in parallel a second task, running the client application, is started and also given access to the IPC gate with less rights (`:m("rw")`, note, this is essential). The server can now receive messages via the IPC gate and provide some service and the client can call operations on the IPC gate to communicate with the server.

Services that keep client specific state need to implement per-client server objects. Usually it is the responsibility of some authority (e.g., Ned) to request such an object from the service via a generic factory object that the service provides initially.

```
local loader = L4.default_loader; -- which is Moe
local svc = loader:new_channel():m("rws");  -- create an IPC gate with rws rights
loader:start({ caps = { service = svc:svr() } }, "rom/my-service");
loader:start({ caps = { foo_service = svc:create(object_to_create, "param") }}, "rom/client");
```

This example is quite similar to the first one, however, the difference is that Ned itself calls the create method on the factory object provided by the server and passes the returned capability of that request as "foo_service" to the client process.

Note

The `svc:create(..)` call blocks on the server. This means the script execution blocks until the my-service application handles the create request.

4.4 Memory management - Data Spaces and the Region Map

4.4.1 User-level paging

Memory management in L4-based systems is done by user-level applications, the role is usually called *pager*. Tasks can give other tasks full or restricted access rights to parts of their own memory. The kernel offers means to give access to memory in a secure way, often referred to as *memory* mapping.

The mapping mechanism allows one task to resolve page faults of another: A thread usually has a pager assigned to it. When the thread causes a page fault, the kernel sends an IPC message to the pager with information about the page fault. The pager answers this IPC by either providing a backing page, or with an error. The kernel will map the backing page into the address space of the faulting thread's task.

These mechanisms can be used to construct a memory and paging hierarchy among tasks. The root of the hierarchy is `sigma0`, which initially gets all system resources and hands them out once on a first-come-first-served basis. Memory resources can be mapped between tasks at a page-size granularity. This size is predetermined by the CPU's memory management unit and is commonly set to 4 kB.

4.4.1.1 Data spaces

A data space is the L4Re abstraction for objects which may be accessed in a memory mapped fashion (i.e., using normal memory read and write instructions). Examples include the sections of a binary which the loader attaches to the application's address space, files in the ROM or on disk provided by a file server, the registers of memory-mapped devices and anonymous memory such as the heap or the stack.

Anonymous memory data spaces in particular (but in general all data spaces except memory mapped IO) can either be constructed entirely from a portion of the RAM or the current working set may be multiplexed on some portion of the RAM. In the first case it is possible to eagerly insert all pages (more precisely page-frame capabilities) into the application's address space such that no further page faults occur when this data space is accessed. In general, however, only the pages for some portion are provided and further pages are inserted by the pager as a result of page faults.

4.4.1.2 Virtual Memory Handling

The virtual memory of each task is constructed from data spaces backing virtual memory regions (VMRs). The management of the VMRs is provided by an object called *region map*. A dedicated region-map object is associated with each task; it allows attaching and detaching data spaces to an address space as well as reserving areas of virtual memory. Since the region-map object possesses all knowledge about the virtual memory layout of a task, it also serves as an application's default pager.

4.4.1.3 Memory Allocation

Operating systems commonly use anonymous memory for implementing dynamic memory allocation (e.g., using `malloc` or `new`). In an L4Re-based system, each task gets assigned a memory allocator providing anonymous memory using data spaces.

See also

L4Re::Dataspace and L4Re::Rm.

4.5 Program Input and Output

The initial environment provides a Vcon capability used as the standard input/output stream.

Output is usually connected to the parent of the program and displayed as debugging output. The standard output is also used as a back end to the C-style printf functions and the C++ streams.

Vcon services are implemented in Moe and the loader as well as by the Fiasco kernel and connected either to the serial line or to the screen if available.

See also

Virtual Console

4.6 Initial Memory Allocator and Factory

The purpose of the memory allocator and of the factory is to provide the application with the means to allocate memory (in the form of data spaces) and kernel objects respectively.

An initial memory allocator and an initial factory are accessible via the initial L4Re environment.

See also

L4Re::Mem_alloc

The factory is a kernel object that provides the ability to create new kernel objects dynamically. A factory imposes a resource limit for kernel memory, and is thus a means to prevent denial of service attacks on kernel resources. A factory can also be used to create new factory objects.

See also

Factory

4.7 Application and Server Building Blocks

So far we have discussed the environment of applications in which a single thread runs and which may invoke services provided through their initial objects.

In the following we describe some building blocks to extend the application in various dimensions and to eventually implement a server which implements user-level objects that may in turn be accessed by other applications and servers.

4.7.1 Creating Additional Application Threads

To create application threads, one must allocate a stack on which this thread may execute, create a thread kernel object and setup the information required at startup time (instruction pointer, stack pointer, etc.). In L4Re this functionality is encapsulated in the pthread library.

4.7.2 Providing a Service

In capability systems, services are typically provided by transferring a capability to those applications that are authorised to access the object to which the capability refers to.

Let us discuss an example to illustrate how two parties can communicate with each other: Assume a simple file server, which implements an interface for accessing individual files: read(pos, buf, length) and write(pos, data, length).

L4Re provides support for building servers based on the class L4::Server_object. L4::Server_object provides an abstract interface to be used with the L4::Server class. Specific server objects such as, in our case, files inherit from L4::Server_object. Let us call this class File_object. When invoked upon receiving a message, the L4::Server will automatically identify the corresponding server object based on the capability that has been provided to its clients and invoke this object's *dispatch* function with the incoming message as a parameter. Based on this message, the server must then decide which of the protocols it implements was invoked (if any). Usually, it will evaluate a protocol specific opcode that clients are required to transmit as one of the first words in the message. For example, assume our server assigns the following opcodes: Read = 0 and Write = 1. The `dispatch` function calls the corresponding server function (i.e., `File_object::read()` or `File_object::write()`), which will in turn parse additional parameters given to the function. In our case, this would be the position and the amount of data to be read or written. In case the write function was called the server will now update the contents of the file with the data supplied. In case of a read it will store the requested part of the file in the message buffer. A reply to the client finishes the client request.

4.8 Pthread Support

L4Re supports the standard pthread library functionality.

Therefore L4Re itself does not contain any documentation for pthreads itself. Please refer to the standard pthread documentation instead.

The L4Re specific parts will be described herein.

- Include pthread-l4.h header file:
  ```
  #include <pthread-l4.h>
  ```
- Return the local thread capability of a pthread thread:

  Use `pthread_l4_cap(pthread_t t)` to get the capability index of the pthread t.

  For example:
  ```
  pthread_l4_cap(pthread_self());
  ```
- Setting the L4 priority of an L4 thread works with a special scheduling policy (other policies do not affect the L4 thread priority):
  ```
  pthread_t t;
  pthread_attr_t a;
  struct sched_param sp;
  pthread_attr_init(&a);
  sp.sched_priority = l4_priority;
  pthread_attr_setschedpolicy(&a, SCHED_L4);
  pthread_attr_setschedparam(&a, &sp);
  pthread_attr_setinheritsched(&a, PTHREAD_EXPLICIT_SCHED);
  if (pthread_create(&t, &a, pthread_func, NULL))
    // failure...
  pthread_attr_destroy(&a);
  ```

- You can prevent your pthread from running immediately after the call to `pthread_create(..)` by adding `PTHREAD_L4_ATTR_NO_START` to the `create_flags` of the pthread attributes. To finally start the thread you need to call `scheduler()->run_thread()` passing the capability of the pthread and scheduling parameters.

```
pthread_t t;
pthread_attr_t attr;
pthread_attr_init(&attr);
attr.create_flags |= PTHREAD_L4_ATTR_NO_START;
if (pthread_create(t, &attr, pthread_func, nullptr))
  // failure...
pthread_attr_destroy(&attr);
// do stuff
auto ret = L4Re::Env::env()->scheduler()->run_thread(pthread_l4_cap(t),
                                                     l4_sched_param(2));
if (l4_error(ret))
  // failure...
```

Constraints on pthread_t, user-land capability slot, and kernel thread-object

- `pthread_l4_cap()` is guaranteed to return the valid capability slot of the pthread (A) until `pthread_join()` or `pthread_detach()` is invoked on (A)'s `pthread_t`.
- `pthread_l4_cap()` exposes internal state of the pthread management, take the necessary precautions as you would for any shared data in concurrent environments. If you use `pthread_l4_cap()` guarding against concurrency issues is your duty.
- There is no guarantee that a valid capability slot points to a present capability.
- **Example**

  It is possible to obtain a valid thread capability slot and for `l4_task_cap_valid()` to return the capability as not present. The following example showcases a possible sequence of events.

```
// Assume: void some_func(void *)
pthread_t pthread = nullptr;
pthread_create(pthread, nullptr, some_func, nullptr);
// pthread running some_func()
l4_cap_idx_t cap_idx = pthread_l4_cap(pthread);
l4_is_valid_cap(cap_idx); // ---> true
long valid = l4_task_cap_valid(L4RE_THIS_TASK_CAP, cap_idx).label());
// valid == 1 --> capability object is present (refers to a kernel object).
// some_func() exits
cap_idx = pthread_l4_cap(pthread);
l4_is_valid_cap(cap_idx); // ---> true
valid = l4_task_cap_valid(L4RE_THIS_TASK_CAP, cap_idx).label());
// valid == 0 --> capability object is not present (refers to no kernel object).
pthread_join(pthread, nullptr); // invalidates the cap slot and frees
                                // the pthread's data structures
// using cap_idx here is undefined behavior.
```

4.9 Interface Definition Language

An interface definition in L4Re is normally declared as a class derived from L4::Kobject_t.

For example, the simple calculator example declares its interface like that:

```
struct Calc : L4::Kobject_t<Calc, L4::Kobject>
{
  L4_INLINE_RPC(int, sub, (l4_uint32_t a, l4_uint32_t b, l4_uint32_t *res));
  L4_INLINE_RPC(int, neg, (l4_uint32_t a, l4_uint32_t *res));
  typedef L4::Typeid::Rpcs<sub_t, neg_t> Rpcs;
};
```

The signature of each function is first declared using one of the RPC macros (see below) and then all the functions need to be listed in the Rpcs type.

Clients invoke these functions with the name given to the RPC macros, `sub` and `neg` above. Servers implement them by defining functions with an `op_` prepended, `op_sub` and `op_neg`. The types of the parameters in the macro definition, on the server side, and on the client side are not the same. The following section describes how they are related to each other.

4.9.1 Parameter types for RPC

Generally all value parameters, const reference parameters, and const pointer parameters to an RPC interface are considered as input parameters for the RPC and are transmitted from the client to the server.

Note

This means that `char const *` is treated as an input `char` and not as a zero terminated string value, for strings see L4::Ipc::String<>.

Parameters that are non-const references or non-const pointers are treated as output parameters going from the server to the client.

There are special data types that appear on only one side (client or server) when used, see the following table for details.

```
L4_RPC(long, test, (int arg1, char const *arg2, unsigned *ret1));
```

The example shows the declaration of a method called `test` with `long` as return type, `arg1` is an `int` input, `arg2` a `char` input, and `ret1` an `unsigned` output parameter.

Type	Direction	Client-Type	Server-Type
`T`	Input	`T`	`T`
`T const &`	Input	`T const &`	`T const &`
`T const *`	Input	`T const *`	`T const &`
`T &`	Output	`T &`	`T &`
`T *`	Output	`T *`	`T &`
`L4::Ipc::In_out<T &>`	In/Out	`T &`	`T &`
`L4::Ipc::In_out<T *>`	In/Out	`T *`	`T &`
`L4::Ipc::Cap<T>`	Input	`L4::Ipc::Cap<T>`	`L4::Ipc::Snd_fpage`
`L4::Ipc::Out<L4::Cap<T> >`	Output	`L4::Cap<T>`	`L4::Ipc::Cap<T> &`
`L4::Ipc::Rcv_fpage`	Input	`L4::Ipc::Rcv_fpage`	`void`
`L4::Ipc::Small_buf`	Input	`L4::Ipc::Small_buf`	`void`

Array types can be used to transmit arrays of variable length. They can either be stored in a client-provided buffer (L4::Ipc::Array), copied into a server-provided buffer (L4::Ipc::Array_in_buf) or directly read and written into the UTCB (L4::Ipc::Array_ref). For latter type, the start position of an array type needs to be known in advance. That implies that only one such array can be transmitted per direction and it must be the last part in the message.

Type	Direction	Client-Type	Server-Type
`L4::Ipc::Array<const T>`	Input	`L4::Ipc::Array<const T>`	`L4::Ipc::Array_ref<const T>`
`L4::Ipc::Array<const T>`	Input	`L4::Ipc::Array<const T>`	`L4::Ipc::Array_in_buf<T> const &`
`L4::Ipc::Array<T> &`	Output	`L4::Ipc::Array<T> &`	`L4::Ipc::Array_ref<T> &`
`L4::Ipc::Array_ref<T> &`	Output	`L4::Ipc::Array_ref<T> &`	`L4::Ipc::Array_ref<T> &`

Finally, there are some optional types where the sender can choose if the parameter should be included in the message. These types are for the implementation of some legacy message formats and should generally not be needed for the definition of ordinary interfaces.

Type	Direction	Client-Type	Server-Type
L4::Ipc::Opt<T>	Input	L4::Ipc::Opt<T>	T
L4::Ipc::Opt<const T*>	Input	L4::Ipc::Opt<const T*>	T
L4::Ipc::Opt<T &>	Output	T &	L4::Ipc::Opt<T> &
L4::Ipc::Opt<T *>	Output	T *	L4::Ipc::Opt<T> &
L4::Ipc::Opt<Array↩ _ref<T> &>	Output	Array_ref<T> &	L4::Ipc::Opt<Array↩ _ref<T>> &

4.9.2 RPC Return Types

On the server side, the return type of an RPC handling function is always `long`. The return value is transmitted via the label field in l4_msgtag_t and is therefore restricted to its length. Per convention, a negative return value is interpreted as an error condition. If the return value is negative, output parameters are not transmitted back to the client.

Attention

The client must never rely on the content of output parameters when the return value is negative.

On the client-side, the return value of the RPC is set as defined in the RPC macro. If `l4_msgtag_t` is given, then the client has access to the full message tag, otherwise the return type should be `long`. Note that the client might not only receive the server return value in response but also an IPC error code.

4.9.3 RPC Method Declaration

RPC member functions can be declared using one of the following C++ macros.

For inline RPC stubs, where the RPC stub code itself is `inline`:

- `L4_INLINE_RPC(res, name, (args...), flags)`

  Define an inline RPC call (type and callable).
- `L4_INLINE_RPC_OP(op, res, name, (args...), flags)`

  Define an inline RPC call with specific opcode (type and callable).
- `L4_INLINE_RPC_NF(res, name, (args...), flags)`

  Define an inline RPC call type (the type only, no callable).
- `L4_INLINE_RPC_NF_OP(opcode, Ret_type, func_name, (args...), flags)`

  Define an inline RPC call type with specific opcode (the type only, no callable).

For external RPC stubs, where the RPC stub code must be defined in a separate compile unit (usually a `.cc` file):

- `L4_RPC(Ret_type, func_name, (args...), flags)`

  Define an RPC call (type and callable).
  - `L4_RPC_OP(opcode, Ret_type, func_name, (args...), flags)`

    Define an RPC call with specific opcode (type and callable).

  `L4_RPC_NF(Ret_type, func_name, (args...), flags)`
- Define an RPC call type (the type only, no callable).
- `L4_RPC_NF_OP(opcode, Ret_type, func_name, (args...), flags)`

  Define an RPC call type with specific opcode (the type only, no callable).

To generate the implementation of an external RPC stub:

- `L4_RPC_DEF(class_name::func_name)`

  Generate the definition of an RPC stub.

The `NF` versions of the macro generally do not generate a callable member function named `<name>` but do only generate the type `<name>_t`. This data type can be used to call the RPC stub explicitly using `<name>_t↩ ::call(L4::Cap<Iface_class> cap, args...)`.

4.10 L4Re Build System

L4Re uses a custom make-based build system, often simply referred to as *BID*.

This section explains how to use BID when writing applications and libraries for L4Re.

4.10.1 Building L4Re

Setting up the Build Directory

L4Re must be built out-of-source. Therefore the first mandatory step is creating and populating a build directory. From the root of the L4Re source tree run

```
make B=<builddir>
```

Other targets that can be executed in the source directory are

update

Update the source directory from svn. Only makes sense when you have downloaded L4Re from the official subversion repository.

help

Show a short help with the most important targets.

Invoking Make

Once the build directory is set up, BID make can be invoked in one of two ways:

1. Go to the build directory and invoke make without special options.
2. Go to a source directory with a BID make file and invoke `make O=<builddir> ...`.

The default target builds the source (as you would expect), other targets that are available in build mode are

cleanfast

Quickly cleans the build directory by removing all subdirectories that contain generated files. The configuration will remain untouched.

clean

Remove generated files. Slower than `make cleanfast` but can be used on selected packages. Use `S=...` to select the target package.

In addition to these targets, there are a number of targets to generate images which are explained elsewhere.

4.10.2 Writing BID Make Files

The BID build system exports different roles that define what should be done in the subdirectory. So a BID make file essentially consists of defining the role and a number of role-dependent make variables. The basic layout should look like this:

```
PKGDIR  ?= <path to package's root directory>  # e.g., '.' or '..'
L4DIR   ?= <path to L4Re source directory>     # e.g. '$(PKGDIR)/../..'

<various definitions>

include $(L4DIR)/mk/<role>.mk
```

`PKGDIR` in the first line defines the root directory of the current package. `L4DIR` in the next line must be pointed to the root of the L4Re source tree the package should be built against. After this custom variable definitions for the role follow. In the final line of the file, the make file with the role-specific rules must be sourced.

The following roles are currently defined:

- project.mk - Sub-project Role
- subdir.mk - Directory Role
- prog.mk - Application Role
- lib.mk - Library Role
- include.mk - Header File Role
- doc.mk - Documentation Role
- test.mk - Test Application Role
- idl.mk - IDL File Role (currently unused)
- runux.mk - Tests in FiascoUX Role

BID-global Variables

This section lists variables that configure how the BID build system behaves. They are applicable for all roles.

Variable	Description
CC	C compiler for target
CXX	C++ compiler for target
HOST_CC	C compiler for host
HOST_CXX	C++ compiler for host

4.10.3 prog.mk - Application Role

The prog role is used to build executable programs.

General Configuration Variables

The following variables can only be set globally for the Makefile:

MODE

Kind of target to build for. The following values are possible:

- `static` - build a statically linked binary (default)
- `shared` - build a dynamically linked binary
- `l4linux` - build a binary for running on L4Linux on the target platform
- `host` - build for host system
- `targetsys` - build a binary for the target platform with the compiler's default settings

SYSTEMS

List of architectures the target can be built for. The entries must be space-separated entries either naming an architecture (e.g. amd64) or an architecture and ABI (e.g, arm-l4f). When not defined, the target will be built for all possible platforms.

TARGET

Name or names of the binaries to compile. This variable may also be postfixed with a specific architecture.

SRC_CC_IS_CXX11

C++ standard to use. Default is `c++0x`.

Target-specific Configuration Variables

The following variables may either be used with or without a description suffix. Without suffix they will be used for all operations. With a specific description their use is restricted to a subset. These specifications include a target file and the architecture, both optional but in this order, separated by underscores. The specific variables will be used in addition to the more general ones.

SRC_C / SRC_CC / SRC_F / SRC_S

.c, .cc, .f90, .S source files.

REQUIRES_LIBS

List of libraries the binary depends on. This works only with libraries that export a pkg_config configuration file. Automatically adds any required include and link options.

DEPENDS_PKGS

List of packages this binary depends on. If one these packages is missing then building of the binary will be skipped.

CPPFLAGS / CFLAGS / CXXFLAGS / FFLAGS / ASFLAGS

Options for the C preprocessor, C compiler, C++ compiler, Fortran compiler and assembler. When used with suffix, the referred element is the source file, not the target file.

LDFLAGS

Options for the linker ld.

LIBS

Additional libraries to link against (with -l).

PRIVATE_LIBDIR

Additional directories to search for libraries.

CRT0 / CRTN

(expert use only) Files containing custom startup and finish code.

LDSCRIPT

(expert use only) Custom link script to use.

4.10.4 include.mk - Header File Role

The header file role is responsible for installing header file at the appropriate location.

The following variables can be used for customizing the process:

INCSRC_DIR

Source directory where the headers can be found. Default is the directory where the Makefile resides.

TARGET

List of header files to install. If left undefined, then `INCSRC_DIR` will be scanned for files with suffix `.h` or `.i`.

EXTRA_TARGET

When TARGET is undefined, then add these files to the headers found by scanning the source directory. Has no effect if `TARGET` has been defined.

CONTRIB_HEADERS

When set, the headers will be installed in `${BUILDDIR}/include/contrib/${PKGNAME}` rather than `${BUILDDIR}/include/l4/${PKGNAME}`.

INSTALL_INC_PREFIX

Base directory where to install the headers. Overwrites `CONTRIB_HEADERS`. The headers will then be found under `${BUILDDIR}/include/${INSTALL_INC_PREFIX}`.

PC_FILENAME

When set, a pkg_config configuration file is created with the given name.

4.10.5 test.mk - Test Application Role

The test role is very similar to the application role, it also builds an executable binary.

The difference is that is also builds for each target a test script that executes the test target either on the host (MODE=host) or a target platform (currently only qemu).

The role accepts all make variables that are accepted by the application role. The only difference is that the `TARGET` variable is not required. If it is missing, the source directory will be scanned for source files that fit the pattern `test_*.c[c]` and create one target for each of them.

Note

It is possible to still use SRC_C[C] when targets are determined automatically. In that case the specified sources will be used in addition* to the main `test_*.c[c]` source.

In addition to the variables above, there are a number of variables that control how the test is executed. All these variables may be used as a global variable that applies to all test or, if the target name is added as a suffix, set for a specific target only.

TEST_TARGET

Name of binary containing the test (default: same as `TARGET`).

TARGET_$(ARCH)

When TARGET is undefined, these targets are added to the list of targets for the specified architecture. For all targets `SRC_C[C]` files must be defined separately.

TEST_KERNEL_ARGS

Arguments to append to the kernel command line. These are also appended when specifying custom ones via a .t-file's -f parameter or when using -d.

TEST_EXPECTED

File containing expected output. By default the variable is empty, which means the test binary is expected to produce TAP test output, that can be directly processed.

TEST_EXPECTED_REPEAT

Number of times the expected output should be repeated, by default 1. When set to 0 then output is expected to repeat forever. This is particularly useful to make sure that stress tests that are meant to run in an endless loop are still alive. Note that such endless tests can only be run by directly executing the test script. They will be skipped when run in a test harness like `prove`.

TEST_TAP_PLUGINS

Specify the plugins that are used to process the output of the test run. The syntax is of the values is:

plugin1:arg1=a,arg2=b plugin2:arg=foo

Multiple plugins separated by a space are loaded in order. Spaces are not allowed inside a plugin specification. One or more arguments are optionally passed to the plugin separated by commas and delimited by a colon.

If the variable is not specified the plugins for TAPOutput and OutputMatching (depending on the TEST_EXPECTED variable) are automatically loaded.

For the supported plugins and their options please refer to their in-line documentation in tool/lib/L4/TapWrapper/↩ Plugin/. The plugin name corresponds to the file stem name in that directory.

TEST_TIMEOUT

Non-standard timeout after which the test run is aborted (useful for tests involving sleep).

NED_CFG

LUA configuration file for startup to give to Ned

REQUIRED_MODULES

Additional modules needed to run the test.

QEMU_ARGS

Additional parameters to supply to QEMU.

MOE_ARGS

Additional parameters to supply to moe.

TEST_ARGS

Additional arguments for the `TEST_STARTER` (tapper-wrapper per default).

TEST_ROOT_TASK

Alternative root task to be used during a test instead of moe.

TEST_ROOT_TASK_ARGS

Arguments passed to `TEST_ROOT_TASK` if `TEST_ROOT_TASK` is different from moe.

KERNEL_CONF

Features the Fiasco kernel must have been compiled with. A space-separated list of config options as used by Kconfig. `run_test` looks for a `globalconfig.out` file in the same directory as the kernel and checks that all options are enabled. If not, the test is skipped. Has only an effect if the `globalconfig.out` file is present.

L4LINUX_CONF

Features the L4Linux kernel must have been compiled with. Similar to `KERNEL_CONF` but checks for a `.config` file in the directory of the L4Linux kernel.

TEST_SETUP

Command to execute before the test is run. The test will only be executed if the command returns 0. If the exit code is 69, the test is marked as skipped with the reason provided in the final line of stdout.

TEST_LOGFILE

Append output of test execution to the given file unless TEST_WORKDIR is given.

TEST_WORKDIR

Create logs, temp and other files below the given directory. That directory is taken as base dir for more automatically created subdir levels using the current test path, in order to guarantee conflict-free usage when running many different tests with a common workdir. When TEST_WORKDIR is provided then TEST_↩ LOGFILE is ignored as it is organized below workdir.

TEST_TAGS

List of conditions for tags provided during execution of a test. A tag can be set to 1, set to 0 or be unspecified via TEST_RUN_TAGS during execution. Therefore there are 4 possible conditions for a tag that can be specified in TEST_TAGS: tag, !tag, +tag and -tag. The following table shows the conditions they represent.

TEST_RUN_TAGS \ TEST_TAGS	tag	!tag	+tag	-tag
tag or tag=1	y		y	
unspecified		y	y	
tag=0		y		y

Example usage:

The tag `long-running` is used by tests which take a long time and should be skipped by default. These tests are marked with the tag long-running unprefixed.

The tag `hardware` is set to 1 at runtime when the tests will run on real hardware. Tests that must not run on real hardware are marked with `!hardware`.

The tag `impl-def` is used by tests that test implementation details. They are run by default but can be excluded from execution by settings TEST_RUN_TAGS to impl-def=0. These tests are marked using `+impl-def`.

If you want to specify multiple tag conditions they need to be separated with a comma.

TEST_PLATFORM_ALLOW and TEST_PLATFORM_DENY

Deny and allow lists of platforms a test is banned from or limited to. If you list platforms in the TEST_↩ PLATFORM_ALLOW variable the test will only be run on these listed platforms and will be skipped on any other platform. If you list platforms in the TEST_PLATFORM_DENY variable the test will be skipped on the listed platforms and will be run on any other platform. You can only use one of these variables per test, not both. See mk/platforms/ for the various identifiers.

Example usage:

Do not run this test on the Raspberry Pi platform

TEST_PLATFORM_DENY_test_xyz := rpi

Only run this test on this test on the RCar3 platform.

TEST_PLATFORM_ALLOW_test_abc := rcar3

TAPARCHIVE

Filename for an archive file to store the resulting TAP output.

In addition to compiled tests, it is also possible to create tests where the test binary or script comes from a different source. These tests must be listed in `EXTRA_TARGET` and for each target a custom `TEST_TARGET` must be provided.

Running Tests

The make role creates a test script which can be found in `<builddir>/test/t/<arch>/<api>`. It is possible to organise the tests further in subdirectories below by specifying a TEST_GROUP.

To be able to execute the test, a minimal test environment needs to be set up by exporting the following environment variables:

KERNEL_<arch>, KERNEL

Fiasco kernel binary to use. The test runner is able to check if the kernel has all features necessary for the test and skip tests accordingly. In order for this to work, the `globalconfig.out` config file from the build directory needs to be available in the same directory as the kernel.

L4LX_KERNEL_<arch>, L4LX_KERNEL

L4Linux binary to use. This is only required to run tests in `mode=l4linux`. If no L4Linux kernel is set then these tests will simply be skipped. The test runner is also able to check if the kernel has all features compiled in that are required to run the test successfully (see make variable `L4LINUX_CONF` above). For this to work, the `.config` configuration file from the build directory needs to be available in the same directory as the kernel.

LINUX_RAMDISK_<arch>, LINUX_RAMDISK

Ramdisk to mount as root in L4Linux. This is only required to run tests in `mode=l4linux`. If not supplied, L4Linux tests will be skipped. The ramdisk must be set up to start the test directly after the initial startup is finished. The name of the test binary is supplied via the kernel command line option `l4re_testprog`. The `tool/test` directory contains an example script `launch-l4linux-test`. which can be copied onto the ramdisk and started by the init script.

In addition to these variables, the following BID variables can be overwritten at runtime: `PT` (for the plaform type) and `TEST_TIMEOUT`. You may also supply `QEMU_ARGS` and `MOE_ARGS` which will be appended to the parameters specified in the BID test make file.

Once the environment is set up, the tests can be run either by simply executing all of them from the build directory with

```
make test
```

or executing them directly, like

```
test/t/amd64_amdfam10/l4f/l4re-core/moe/test_namespace.t
```

or running one or more tests through the test harness `prove`, like

```
prove test/t/amd64_amdfam10/l4f/l4re-core/moe/test_namespace.t
prove -r test/t/amd64_amdfam10/l4f/l4re-core/
prove -rv test/t/amd64_amdfam10/l4f/l4re-core/
```

TEST_TAGS allow for a way to include or exclude whole groups of tests during execution, primarily with prove. You can specify which tests to run at runtime using one of the following ways:

```
$ test/t/amd64_amdfam10/l4f/l4re-core/test_one.t --run-tags slow,gtest-shuffle=0
$ test/t/amd64_amdfam10/l4f/l4re-core/test_one.t -T slow,gtest-shuffle=0
$ prove -r test/t/amd64_amdfam10/l4f/l4re-core/ :: -T slow,gtest-shuffle=0
$ TEST_RUN_TAGS=slow,gtest-shuffle=0 prove -r test/t/amd64_amdfam10/l4f/l4re-core/
```

For each test tag requirements defined in the corresponding TEST_TAGS Makefile variable are tested. If the requirements for tags do not match the test is skipped. The SKIP message will provide insight why the test was skipped:

```
$ make test
...
test/t/amd64_amdfam10/test_one.t .... ok
test/t/amd64_amdfam10/test_two.t .... skipped: Running this test requires tag slow to be set to 1.
test/t/amd64_amdfam10/test_three.t .. ok
```

When tags are provided, the tests requiring those tags are now also executed while the tests that forbid them are skipped:

```
$ TEST_RUN_TAGS=slow,gtest-shuffle
$ make test
...
test/t/amd64_amdfam10/test_one.t .... ok
test/t/amd64_amdfam10/test_two.t .... ok
test/t/amd64_amdfam10/test_three.t .. skipped: Running this test requires tag gtest-shuffle to be set to 0 or
```

For further details on how values in TEST_TAGS and TEST_RUN_TAGS interact, see the help text for TEST_TAGS.

Running Tests in External Programs

You can hand-over test execution to an external program by setting the environment variable `EXTERNAL_TEST↩_STARTER` to the full path of that program:

```
export EXTERNAL_TEST_STARTER=/path/to/external/test-starter
make test
```

EXTERNAL_TEST_STARTER

This variable is evaluated by `tool/bin/run_test` (the backend behind `make test`) and contains the full path to the tool which actually starts the test instead of the test itself.

The `EXTERNAL_TEST_STARTER` can be any program instead of the default execution via `make qemu E=maketest`. Its output is taken by `run_test` as the test output.

Usually it is just a bridge to prepare the test execution, e.g., it could create the test as image and start that image via a simulator.

Running Tests in a Simulator

Based on above mechanism there is a dedicated external test starter `tool/bin/teststarter-image-telnet.↩pl` shipped in BID which assumes an image to be started with another program which provides test execution output on a network port.

This can be used to execute tests in a simulator, like this:

```
export EXTERNAL_TEST_STARTER=$L4RE_SRC/tool/bin/teststarter-image-telnet.pl
export SIMULATOR_START=/path/to/configured/simulator-exe
make test
```

After building the image and starting the simulator it contacts the simulator via a network port (sometimes called "telnet" port) to pass-through its execution output as its own output so it gets captured by `run_test` as usual.

The following variables control `teststarter-image-telnet.pl`:

SIMULATOR_START

This points to the full path of the program that actually starts the prepared test image. Most often this is the frontend script of your simulator environment which is pre-configured so that it actually works in the way that `teststarter-image-telnet.pl` expects from the following settings.

SIMULATOR_IMAGETYPE

The image type to be generated via `make $SIMULATOR_IMAGETYPE E=maketest`. Default is `elfimage`.

SIMULATOR_HOST

The simulator will be contacted via socket on that host to read its output. Default is `localhost`.

SIMULATOR_PORT

The simulator will be contacted via socket on that port to read its output. Default is `11111`.

SIMULATOR_START_SLEEPTIME

After starting the simulator it waits that many seconds before reading from the port. Default is `1` (second).

Running tests without tapper-wrapper

In case you want to replace the tapper-wrapper test starter, you can replace the default one by setting the environment variable `TEST_STARTER` to the path of your test starter. Then your test starter can use the same environment which is normally set up for the default starter, which includes environment variables provided by the build system as well as the test itself. Among these are `SEARCHPATH`, `MODE`, `ARCH`, `MOE_CFG`, `MOE_ARGS`, `TEST_TIMEOUT`, `TEST_TARGET`, `TEST_EXPECTED`, `QEMU_ARGS` and many more.

Debugging Tests

The test script is only a thin wrapper that sets up the test environment as it was defined in the make file and then executes two scripts: `tapper-wrapper` and `run_test`.

The main work horse of the two is `tool/bin/run_test`. It collects the necessary files and starts qemu to execute the test. This script is always required.

There is then a second script wrapped around the test runner: `tool/bin/tapper-wrapper`. This tool inspects the output of the test runner and reformats it, so that it can be read by tools like `prove`. If the test produces tap output, then the script scans for this output and filters away all the debug output. If `TEST_EXPECTED` was defined, then the script scans the output for the expected lines and prints a suitable TAP message with success or failure. It also makes sure that qemu is killed as soon as the test is finished.

There are a number of command-line parameters that allow to quickly change test parameters for debugging purposes. Run the test with '–help' for more information about available parameters.

4.11 Kernel Factory

The kernel factory is a kernel object that provides the ability to create new kernel objects dynamically.

The kernel factory enforces a memory quota. This quota defines the maximum amount of kernel memory the factory service can use to construct the requested objects. When the quota is depleted, the factory refuses the creation of new objects.

The quota may be higher than the amount of available kernel memory; ultimately, the amount of available kernel memory is the strict limit for the factory to remain operational.

The kernel factory creates the following kinds of objects:

- DMA space
- L4::Factory
- L4::Ipc_gate
- L4::Irq
- L4::Semaphore
- L4::Task
- L4::Thread
- L4::Vm

The protocol IDs for objects in this list are given in l4_msgtag_protocol. The protocol ID shall be used as the second argument for L4::Factory.create(Cap<void>, long, l4_utcb_t *).

For the C++ interface see L4::Factory, for the C interface see Factory.

4.11.1 Passing parameters for the create stream

L4::Factory.create() returns a create stream that allows arguments to be forwarded to the constructor of the object to be created.

Objects that support additional parameters on their creation are presented with a non-empty list of parameters. The parameters are listed in the order they should be provided to a create stream returned by L4::Factory.create().

- Dmar_space()
- L4::Factory(l4_umword_t)
  - Argument: factory quota (in bytes).
  - See L4::Factory.create_factory() for details.
- L4::Ipc_gate()
  - Creates an unbound IPC gate.
  - Alternatively, an IPC gate can be immediately bound to a thread upon creation using L4::Factory.create_gate().
- L4::Irq()
- L4::Semaphore()
- L4::Task(l4_fpage_t)
  - Argument: utcb_area
  - See L4::Factory.create_task() for details.
- L4::Thread()
- L4::Vm()

Chapter 5

L4Re Servers

Here you shall find a quick overview over the standard services running on Fiasco.OC and L4Re.

Sigma0, the Root Pager

Sigma0 is a special server running on L4 because it is responsible of resolving page faults for the root task, the first useful task on L4Re. Sigma0 can be seen as part of the kernel, however it runs in unprivileged mode. To run something useful on Fiasco.OC you usually need to run Sigma0, nevertheless it is possible to replace Sigma0 by a different implementation.

For more details see Sigma0, the Root-Pager

Moe, the Root Task

Moe is our implementation of the L4 root task that is responsible for bootstrapping the system, and to provide basic resource management services to the applications on top. Therefore Moe provides L4Re resource management and multiplexing services:

- **Memory** in the form of memory allocators (L4Re::Mem_alloc, L4::Factory) and data spaces (L4Re::Dataspace)
- **Cpu** in the form of basic scheduler objects (L4::Scheduler)
- **Vcon** multiplexing for debug output (output only)
- **Virtual memory management** for applications, L4Re::Rm

Moe further provides an implementation of L4Re name spaces (L4Re::Namespace), which are for example used to provide a read only directory of all multi-boot modules. In the case of a boot loader, like grub that enables a VESA frame buffer, there is also a single instance of an L4Re graphics session (L4Re::Goos).

To start the system Moe starts a single ELF program, the init process. The init process (usually Ned, see the next section) gets access to all resources managed by Moe and to the Sigma0 root pager interface.

For more details see Moe, the Root-Task.

Ned, the Default Init Process

To keep the root task free from complicated scripting engines and to avoid circular dependencies in application startup (that could lead to dead locks) the configuration and startup of the real system is managed by an extra task, the init process.

Ned is such an init process that allows system configuration via Lua scripts.

For more information see Ned.

Io, the Platform and Device Resource Manager

Because all peripheral management of Fiasco.OC is done in user-level applications, there is the need to have a centralized management of the resources belonging to the platform and to peripheral devices.

This is the job of Io. Io provides portable abstractions for iterating and accessing devices and their resources (IRQ's, IO Memory...), as well as delegating access to those resources to other applications (e.g., device drivers).

For more details see Io, the Io Server.

Other Servers

The following additional server package are available on top of the core L4Re environment.

- Rtc, the Real-Time Clock Server

  is a simple multiplexer for real-time clock hardware on your platform.

- fb-drv, the Low-Level Graphics Driver

  provides low-level access and initialization of various graphics hardware. It has support for running VESA BIOS calls on Intel x86 platforms, as well as support for various ARM display controllers. `fb-drv` provides a single instance of the L4Re::Goos interface and can serve as a back end for the Mag server, in particular, if there is no graphics support in the boot loader.

- l4vio_net_p2p, a virtual network point-to-point link

- Uvmm, the virtual machine monitor

- Mag, the GUI Multiplexer

  Our default multiplexer for the graphics hardware is Mag. Mag is a Nitpicker (TODO: ref) derivate that allows secure multiplexing of the graphics and input hardware among multiple applications and multiple complete windowing environments.

- Sigma0, the Root-Pager

- Cons, the Console Multiplexer

  An interactive multiplexer for console in- and output. It buffers the output from different L4 clients and allows to switch between them to redirect input.

5.1 Sigma0, the Root-Pager

Sigma0 is a special L4 server that serves as the origin for mapping memory.

It is started by Fiasco.OC on the system boot and gets full access to all userland RAM and device memory. It functions as the pager (main memory provider) for Moe and as the provider for device memory for Io. Moe and Io are trusted and usually the only applications besides Ned that get a capability for Sigma0. Memory can be requested from Sigma0 directly via an IPC, or indirectly by causing page faults and having them resolved by Sigma0.

5.1.1 Factory

There is only one instance of Sigma0 in an L4Re system, which is made accessible to Moe via an IPC gate capability. Using this capability, Moe can request Sigma0 to create new communication channels to itself by creating additional IPC gate capabilities. This request is done using the L4::Factory interface. This is the only kind of object that can be created by the factory in Sigma0.

List of objects that the Sigma0 Factory can create:

- Sigma0 ()
  - Use protocol id L4_PROTO_SIGMA0 for creation
  - No arguments supported
  - identical to l4sigma0_new_client()

See also

Sigma0 API

5.2 Moe, the Root-Task

Moe is the default root-task implementation for L4Re-based systems.

Moe is the first task which is usually started in L4Re-based systems. The micro kernel starts *Moe* as the Root-Task.

5.2.1 Moe objects

Moe provides a default implementation for the basic L4Re abstractions, such as data spaces (L4Re::Dataspace), region maps (L4Re::Rm), memory allocators (L4::Factory, L4Re::Mem_alloc), name spaces (L4Re::Namespace) and so on (see L4Re Interface). These are described in the following subsections.

5.2.1.1 Factory

The factory in Moe is responsible for all kinds of dynamic object allocation.

Moe's factory allows allocation of the following objects:

- L4Re::Namespace
- L4Re::Dataspace, RAM allocation
- L4Re::Dma_space, memory management for DMA-capable devices
- L4Re::Rm, virtual memory management for application tasks
- L4::Vcon (output only)
- L4::Scheduler, to provide a restricted priority / CPU range for clients
- L4::Factory, to provide a quota limited allocation for clients

Note

L4::Scheduler objects can be only created through the user factory provided by Moe to the initial application. Other factory instances cannot create this object.

5.2.1.1.1 Passing parameters to the create stream L4::Factory.create() returns a create stream that allows arguments to be forwarded to the the object creation in Moe.

Objects that support additional parameters on their creation are presented next with a non-empty list of parameters. The parameters are listed in the order they should be provided to a create stream. Optional parameters are identified by their default values. Multiple entries in the next list denote different ways of initializing an object.

- L4Re::Namespace `()`
  - For more details see Namespace
- L4Re::Dataspace `(l4_mword_t size, l4_umword_t flags = 0, l4_umword_t align = 0)`
  - Argument `size`: size in bytes (mandatory)
  - Argument `flags`: special dataspace properties, see L4Re::Mem_alloc::Mem_alloc_flags
  - Argument `align`: Log2 alignment of dataspace if supported by allocator
  - See detailed description of the parameters in L4Re::Mem_alloc::alloc()
  - For details on the types of dataspaces provided by Moe, see Dataspace
- L4Re::Dma_space `()`
  - For more details see DMA Space
- L4Re::Rm `()`
  - For more details see Region Map
- L4::Vcon `(char const *label, l4_mword_t color = 7)`
  - Argument `label`: label used as prefix for the console output
  - Argument `color`: color code `0..15`
  - For more details see Log Subsystem

- L4::Vcon `(char const *label, char const *color = "w")`
  - Argument `label`: label used as prefix for the console output
  - Argument `color`: color code
    - The color is identified by a single character
    - Supported colors: `N`, `n`, `R`, `r`, `G`, `g`, `Y`, `y`, `B`, `b`, `M`, `m`, `C`, `c`, `W`, `w`
  - For more details see Log Subsystem
- L4::Scheduler `(l4_mword_t limit, l4_mword_t offset, l4_umword_t bitmap = ~0UL)`
  - Argument `limit`: maximum priority
  - Argument `offset`: priority offset
  - Argument `bitmap`: bitmap of CPUs
  - Argument `limit` must be greater than `offset`
  - For more details see Scheduler subsystem
- L4::Factory `(l4_mword_t quota)`
  - Argument `quota`: limit in bytes (not zero)
  - The limit is deducted from the limit of the factory that creates the new factory

5.2.1.2 Namespace

Moe provides a name space conforming to the L4Re::Namespace interface (see Name-space API). Per default Moe creates a single name space for the Boot FS. That is available as `rom` in the initial objects of the init process.

5.2.1.2.1 Boot FS The Boot FS subsystem provides access to the files loaded during the platform boot (or available in ROM). These files are either linked into the boot image or loaded via a flexible boot loader, such as GRUB.

The subsystem provides an L4Re::Namespace object as directory and an L4Re::Dataspace object for each file.

By default all files are read only and visible in the namespace *rom*. As an option, files can be supplied with the argument `:rw` to mark them as writable modules. Moe will allow read and write access to these dataspaces and make them visible in a different namespace called *rwfs*.

An example entry in 'modules.list' would look like this:

```
module somemodule :rw
```

Note

In order for a client to receive write permissions to the dataspace, the corresponding cap also needs write permissions.

5.2.1.3 Dataspace

Dataspaces can be allocated with an arbitrary size. The granularity for memory allocation however is the machine page size (L4_PAGESIZE). A dataspace user must be aware that, as a result of this page-size granularity, there may be padding memory at the end of a dataspace which is accessible to each client. Moe currently allows most dataspace operations on this padding area. Nonetheless, the client must not make any assumptions about the size or content of the padding area, as this behaviour might change in the future.

The provided data spaces can have different characteristics:

- Physically contiguous and pre-allocated
- Non contiguous and on-demand allocated with possible copy on write (COW)

5.2.1.4 Log Subsystem

The logging facility of Moe provides per application tagged and synchronized log output.

5.2.1.5 DMA Space

5.2.1.6 Scheduler subsystem

The scheduler subsystem provides a simple scheduler proxy for scheduling policy enforcement.

The priority offset provided on the creation of a scheduler proxy defines the minimum priority assigned to threads which are scheduled by that instance of the scheduler proxy. The offset is implicitly added to priorities provided to L4::Scheduler.run_thread().

5.2.1.7 Region Map

5.2.2 Command Line Options

Moe's command-line syntax is:

```
moe [--debug=<flags>] [--init=<binary>] [--l4re-dbg=<flags>] [--ldr-flags=<flags>] [-- <init options>]
```

--debug=<debug flags>

This option enables debug messages from Moe itself, the `<debug flags>` values are a combination of `info`, `warn`, `boot`, `server`, `loader`, `exceptions`, and `ns` (or `all` for full verbosity).

--init=<init process>

This options allows to override the default init process binary, which is 'rom/ned'.

--l4re-dbg=<debug flags>

This option allows to set the debug options for the L4Re runtime environment of the init process. The flags are the same as for `--debug=`.

--ldr-flags=<loader flags>

This option allows setting some loader options for the L4Re runtime environment. The flags are `pre_alloc`, `all_segs_cow`,and `pinned_segs`.

-- <init options>

All command-line parameters after the special `--` option are passed directly to the init process.

5.3 Ned, the Init Process

Ned's job is to bootstrap the system running on L4Re.

The main thing to do here is to coordinate the startup of services and applications as well as to provide the communication channels for them. The central facility in Ned is the Lua (http://www.lua.org) script interpreter with the L4Re and ELF-loader bindings.

The boot process is based on the execution of one or more Lua scripts that create communication channels (IPC gates), instantiate other L4Re objects, organize capabilities to these objects in sets, and start application processes with access to those objects (or based on those objects).

For starting applications, Ned depends on the services of Moe, the Root-Task or another *loader*, which must provide data spaces and region maps. Ned also uses the 'rom' capability as source for Lua scripts and at least the 'l4re' binary (the runtime environment core) running in each application.

Each application Ned starts is equipped with an L4Re::Env environment that provides information about all the initial objects made accessible to this application.

5.3.1 Lua Bindings for L4Re

Ned provides various bindings for L4Re abstractions. These bindings are located in the 'L4' package (`require "L4"`).

5.3.1.1 Capabilities in Lua

Capabilities are handled as normal values in Lua. They can be stored in normal variables or Lua compound structures (tables). A capability in Lua possesses additional information about the access rights that shall be transfered to other tasks when the capability is transfered. To support implementation of the Principle of Least Privilege, minimal rights are assigned by default. Extended rights can be added using the method `mode`("...") (short `m`("...")) that returns a new reference to the capability with the given rights.

Note

It is generally impossible to elevate the real access rights to an object. This means that if Ned has only restricted rights to an object it is not possible to upgrade the access rights with the `mode` method.

The capabilities in Lua also carry dynamic type information about the referenced objects. They thereby provide type-specific operations on the objects, such as the `create` operation on a generic factory or the `query` and `register` operations on a name space.

5.3.1.2 Access to L4Re::Env Capabilities

The initial objects provided to Ned itself are accessible via the table `L4.Env`. The default (usually unnamed) capabilities are accessible as `factory`, `log`, `mem_alloc`, `parent`, `rm`, and `scheduler` in the `L4.Env` table.

5.3.1.3 Constants

Protocols

The protocol constants are defined by default in the L4 package's table `L4.Proto`. The definition is not complete and only covers what is usually needed to configure and start applications. The protocols are for example used as first argument to the `Factory:create` method.

```
Proto = {
  Dataspace = 0x4000,
  Namespace = 0x4001,
  Goos      = 0x4003,
  Mem_alloc = 0x4004,
  Rm        = 0x4005,
  Event     = 0x4006,
  Inhibitor = 0x4007,
  Sigma0    = -6,
  Log       = -13,
  Scheduler = -14,
  Factory   = -15,
  Vm         = -16,
  Dma_space  = -17,
  Irq_sender = -18,
  Semaphore  = -20,
  Iommu      = -22,
  Ipc_gate  = 0,
}
```

Debugging Flags

Debugging flags used for the applications L4Re core:

```
Dbg = {
  Info       = 1,
  Warn       = 2,
  Boot       = 4,
  Server     = 0x10,
  Exceptions = 0x20,
  Cmd_line   = 0x40,
  Loader     = 0x80,
  Name_space = 0x400,
  All        = 0xffffffff,
}
```

Loader Flags

Flags for configuring the loading process of an application.

```
Ldr_flags = {
  eager_map    = 0x1, -- L4RE_AUX_LDR_FLAG_EAGER_MAP
  all_segs_cow = 0x2, -- L4RE_AUX_LDR_FLAG_ALL_SEGS_COW
  pinned_segs  = 0x4, -- L4RE_AUX_LDR_FLAG_PINNED_SEGS
}
```

5.3.1.4 Application Startup Details

The central facility for starting a new task with Ned is the class `L4.Loader`. This class provides interfaces for conveniently configuring and starting programs. It provides three operations:

- `new_channel()` Returns a new IPC gate that can be used to connect two applications
- `start()` and `startv()` Start a new application process and return a process object

The `new_channel()` call is used to provide a service application with a communication channel to bind its initial service to. The concrete behavior of the object and the number of IPC gates required by a server depends on the server implementation. The channel can the be passed to client applications as well or can be used for operations within the script itself.

`start()` and `startv()` always require at least two arguments. The first one is a table that contains information about the initial objects an application shall get. The second argument is a string, which for `start()` is the program

name plus a white-space-separated list of program arguments (argv). For `startv()` the second argument is just the program binary name – which may contain spaces –, and the program arguments are provided as separate string arguments following the binary name (allowing spaces in arguments, too). The last optional argument is a table containing the POSIX environment variables for the program.

The Loader class uses reasonable defaults for most of the initial objects. However, you can override any initial object with some user-defined values. The main elements of the initial object table are:

- `factory` The factory used by the new process to create new kernel objects, such as threads etc. This must be a capability to an object implementing the L4::Factory protocol and defaults to the factory object provided to Ned.
- `mem` The memory allocator provided to the application and used by Ned allocates data spaces for the process. This defaults to Ned's memory allocator object (see L4Re::Mem_alloc).
- `rm_fab` The generic factory object used to allocate the region-map object for the process. (defaults to Ned's memory allocator).
- `log_fab` The generic factory to create the L4Re::Log object for the application's output (defaults to Ned's memory allocator). The `create` method of the `log_fab` object is called with `log_tag` and `log_color`, from this table, as arguments.
- `log_tag` The string used for tagging log output of this process (defaults to the program name) (see `log_fab`).
- `log_color` The color used for the log tag (defaults to "white").
- `scheduler` The scheduler object used for the process' threads (defaults to Ned's own scheduler).
- `caps` The table with application-specific named capabilities (default is an empty table). If the table does not contain a capability with the name 'rom', the 'rom' capability from Ned's initial caps is inserted into the table.

5.3.1.5 Access to the kernel debugger

Applications can enrich the kernel debugger with information using the API defined in l4/sys/debugger. In order to do so, the developer has to assign access to the kernel debugger kernel object to the application. This can be done like this:

```
L4.default_loader:start({ caps = { jdb = L4.Env.jdb; }}, "rom/example")
```

5.3.1.6 Using the interactive ned prompt

Ned can be used in interactive mode by connecting the small ned-prompt helper tool to the command capability. Add the following code snippet at the end of your ned script:

```
local L4 = require("L4");
local l = L4.default_loader;
cmd = l:new_channel()
l:start({caps = {log = L4.Env.log, svr = cmd}}, "rom/ned-prompt")
L4.server_loop(cmd)
```

The script hands in ned's own log capability to `ned-prompt`. This ensures that input and output of ned and the prompt appear on the same console.

`ned-prompt` needs to be added to your modules list.

5.3.2 Command Line Options

Ned's command line syntax is:

```
[--execute|-e STATEMENT] <lua script> [options passed to lua script]
```

Ned interprets the first non-option argument `<lua script>` as the Lua script which it should load and run. All arguments following the first non-option argument are passed as arguments to the Lua script via Lua's global `arg` table.

- Execute Statement Option: **execute**, **e**

  Execute the Lua statement `STATEMENT`.

5.4 Io, the Io Server

The Io server handles all platform devices and resources such as I/O memory, ports (on x86) and interrupts, and grants access to those to clients.

Upon startup Io discovers all platform devices using available means on the system, e.g. on x86 the PCI bus is scanned and the ACPI subsystem initialised. Available I/O resource can also be configured via configuration scripts.

Io's configuration consists of two parts:

- the description of the real hardware
- the description of virtual buses

Both descriptions represent a hierarchical (tree) structure of device nodes. Where each device has a set of resources attached to it. And a device that has child devices can be considered a bus.

Hardware Description

The hardware description represents the devices that are available on the particular platform including their resource descriptions, such as MMIO regions, IO-Port regions, IRQs, bus numbers etc.

The root of the hardware devices is formed by a system bus device (accessible in the configuration via Io.system↩_bus()). As mentioned before, platforms that support methods for device discovery may populate the hardware description automatically, for example from ACPI. On platforms that do not have support for such methods you have to specify the hardware description by hand. A simple example for this is `x86-legacy.devs`.

Virtual Bus Description

Each Io server client is provided with its own virtual bus which it can iterate to find devices. A virtual PCI bus may be a part of this virtual bus.

Figure 5.1 IO Service Architecture Overview

The Io server must be configured to create virtual buses for its clients.

This is done with at least one configuration file specifying static resources as well as virtual buses for clients. The configuration may be split across several configuration files passed to Io through the command line.

To allow clients access to available devices, a virtual system bus needs to be created that lists the devices and their resources that should be available to that client. The names of the buses correspond to the capabilities given to Io in its launch configuration.

A very simple configuration for Io could look like this:

```
-- vim:ft=lua
-- Example configuration for io
-- Configure two platform devices to be known to io
Io.Dt.add_children(Io.system_bus(), function()
  -- create a new hardware device called "FOODEVICE"
  FOODEVICE = Io.Hw.Device(function()
    -- set the compatibility IDs for this device
    -- a client tries to match against these IDs and configures
    -- itself accordingly
    -- the list should be sorted from specific to less specific IDs
    compatible = {"dev-foo,mmio", "dev-foo"};
    -- set the 'hid' property of the device, the hid can also be used
    -- as a compatible ID when matching clients
    Property.hid  = "dev-foo,Example";
    -- note: names for resources are truncated to 4 letters and a client
```

```
    -- can determine the name from the ID field of a l4vbus_resource_t
    -- add two resources 'irq0' and 'reg0' to the device
    Resource.irq0 = Io.Res.irq(17);
    Resource.reg0 = Io.Res.mmio(0x6f000000, 0x6f007fff);
  end);
  -- create a new hardware device called "BARDEVICE"
  BARDEVICE = Io.Hw.Device(function()
    -- set the compatibility IDs for this device
    -- a client tries to match against these IDs and configures
    -- itself accordingly
    -- the list should be sorted from specific to less specific IDs
    compatible = {"dev-bar,mmio", "dev-bar"};
    -- set the 'hid' property of the device, the hid can also be used
    -- as a compatible ID when matching clients
    Property.hid  = "dev-bar,Example";
    -- Specify that this device is able to use direct memory access (DMA).
    -- This is needed to allow clients to gain access to DMA addresses
    -- used by this device to directly access memory.
    Property.flags = Io.Hw_device_DF_dma_supported;
    -- note: names for resources are truncated to 4 letters and a client
    -- can determine the name from the ID field of a l4vbus_resource_t
    -- add three resources 'irq0', 'irq1', and 'reg0' to the device
    Resource.irq0 = Io.Res.irq(19);
    Resource.irq1 = Io.Res.irq(20);
    Resource.reg0 = Io.Res.mmio(0x6f100000, 0x6f100fff);
  end);
end);
Io.add_vbusses
{
-- Create a virtual bus for a client and give access to FOODEVICE
  client1 = Io.Vi.System_bus(function ()
    dev = wrap(Io.system_bus():match("FOODEVICE"));
  end);
-- Create a virtual bus for another client and give it access to BARDEVICE
  client2 = Io.Vi.System_bus(function ()
    dev = wrap(Io.system_bus():match("BARDEVICE"));
  end);
}
```

Each device supports a 'compatible' property. It is a list of compatibility strings. A client matches itself against one (or multiple) compatibility IDs and configures itself accordingly. All other device members are handled according to their type. If the type is a resource (Io.Res) it is added as a named resource. Note that resource names are truncated to 4 letters and are stored in the ID field of a l4vbus_resource_t. If the type is a device it is added as a child device to the current one. All other types are treated as a device property which can be used to configure a device driver. Right now, device properties are internal to Io only.

Matching and Assigning PCI Devices

Assigning clients PCI devices could look like this:

```
-- This is a configuration snippet for PCI device selection
local hw = Io.system_bus();
Io.add_vbusses
{
  pciclient = Io.Vi.System_bus(function ()
    PCI = Io.Vi.PCI_bus(function ()
      pci_mm      = wrap(hw:match("PCI/CC_04"));
      pci_net     = wrap(hw:match("PCI/CC_02"));
      pci_storage = wrap(hw:match("PCI/CC_01"));
    end)
  end)
}
```

The "PCI/" is followed by a bus-specific ID string. The format of the PCI ID string may be one of the following:

- PCI/CC_cc
- PCI/CC_ccss
- PCI/CC_ccsspp
- PCI/VEN_vvvv
- PCI/DEV_dddd

- PCI/SUBSYS_ssssssss
- PCI/REV_rr
- PCI/ADR_xxxx:xx:xx.x

Where:

- `cc` is the hexadecimal representation of the class code byte
- `ss` is the hexadecimal representation of the subclass code byte
- `pp` is the hexadecimal representation of the programming interface byte
- `vvvv` is the hexadecimal representation of the vendor ID
- `dddd` is the hexadecimal representation of the device ID
- `ssssssss` is the hexadecimal representation of the subsystem ID
- `rr` is the hexadecimal representation of the revision byte
- `xxxx:xx:xx.x` is the bus address in PCI nomenclature

As a special extension Io supports replacing the ID string with a human-readable common PCI class name. The following table gives an overview of the names known to Io and their respective PCI class and subclass.

Common Name	Description	PCI ID string
storage	Mass storage controller	CC_01
scsi	SCSI storage controller	CC_0100
ide	IDE interface	CC_0101
floppy	Floppy disk controller	CC_0102
raid	RAID bus controller	CC_0104
ata	ATA controller	CC_0105
sata	SATA controller	CC_0106
sas	Serial attached SCSI controller	CC_0107
nvm	Non-volatile memory controller	CC_0108
-	-	-
network	Network controller	CC_02
ethernet	Ethernet controller	CC_0200
token_ring	Token ring network controller	CC_0201
fddi	FDDI network controller	CC_0202
atm	ATM network controller	CC_0203
isdn	ISDN controller	CC_0204
picmg	PICMG controller	CC_0206
net_infiniband	Infiniband controller	CC_0207
fabric	Fabric controller	CC_0208
network_nw	Network controller e.g. Wifi	CC_0280
-	-	-
display	Display controller	CC_03
vga	VGA compatible controller	CC_0300
xga	XGA compatible controller	CC_0301
-	-	-
media	Multimedia controller	CC_04
mm_video	Multimedia video controller	CC_0400
mm_audio	Multimedia audio controller	CC_0401

Common Name	Description	PCI ID string
telephony	Computer telephony device	CC_0402
audio	Audio device	CC_0403
-	-	-
bridge	Bridge	CC_06
br_host	Host bridge	CC_0600
br_isa	ISA bridge	CC_0601
br_eisa	EISA bridge	CC_0602
br_microchannel	MicroChannel bridge	CC_0603
br_pci	PCI bridge	CC_0604
br_pcmcia	PCMCIA bridge	CC_0605
br_nubus	NuBus bridge	CC_0606
br_cardbus	CardBus bridge	CC_0607
br_raceway	RACEway bridge	CC_0608
br_semi_pci	Semi-transparent PCI-to-PCI bridge	CC_0609
br_infiniband_to_pci	InfiniBand to PCI host bridge	CC_060a
-	-	-
com	Communication controller	CC_07
com_serial	Serial controller	CC_0700
com_parallel	Parallel controller	CC_0701
com_multiport_ser	Multiport serial controller	CC_0702
com_modem	Modem	CC_0703
com_gpib	GPIB controller	CC_0704
com_smart_card	Smart card controller	CC_0705
-	-	-
serial_bus	Serial bus controller	CC_0c
firewire	FireWire (IEEE 1394)	CC_0c00
access_bus	ACCESS bus	CC_0c01
ssa	SSA	CC_0c02
usb	USB controller	CC_0c03
fibre_channel	Fibre channel	CC_0c04
smbus	SMBus	CC_0c05
bus_infiniband	InfiniBand	CC_0c06
ipmi_smic	IPMI SMIC interface	CC_0c07
sercos	SERCOS interface	CC_0c08
canbus	CAN bus	CC_0c09
-	-	-
wireless	Wireless controller	CC_0d
bluetooth	Bluetooth	CC_0d11
w_8021a	802.1a controller	CC_0d20
w_8021b	802.1b controller	CC_0d21

Strong Matching of PCI Devices

If more specific matching of PCI devices is required it is possible to concatenate multiple ID strings using `&`. An example where a specific device from a specific vendor at a fixed bus address is matched would use the string `PCI/VEN_vvvv&DEV_dddd&ADR_xxxx:xx:xx.x`.

Isolation of PCIe devices

PCIe encodes device communication with a network-like protocol with destination headers and packet fragmentation allowing a devices to talk directly to other devices. This potentially works against security boundaries for a system. E.g. two network cards could exchange packets and thereby leak information from one security domain to the other without involvement of the OS.

PCIe introduced an optional capability named PCI Access Control Services (PCI/ACS) to control communication between PCIe devices.

With PCI/ACS it is possible to restrict inter-device communication between PCIe devices.

PCI/ACS is optional and for Intel chipsets, it is usually only implemented on high-end PCI platform controller hubs (PCHs), and is missing on low-end and mobile PCHs. On some Intel-PCHs there exist facilities that allow for similar isolation.

If IO encounters a supported PCH, it will enable those facilities in order to enforce device isolation.

Command Line Options

The Io Server supports the following optional parameters:

```
[--verbose|v] [--transparent-msi] [--trace <trace_mask>] [--acpi-debug-level <debug_level>] [config_files]
```

- **verbose|v**

  By default, error debug messages are enabled. This option increments the verboseness level, and can be applied multiple times to reach the desired debug level. The available debug levels are ordered as: `DBG_ERR` (default, level 1), `DBG_WARN`, `DBG_INFO`, `DBG_DEBUG`, `DBG_DEBUG2` and `DBG_ALL` (level 6).

- **transparent-msi**

  Enable MSI on PCI devices which support this feature. This is transparent to clients, as there are no changes in the API used to interact with PCI device via interrupts.

- **acpi-debug-level <level_mask>**

  Set the ACPI debug level. The `<level_mask>` is a mask that selects components of interest for debugging. It can be constructed from the ACPI debug constants defined in the linux kernel, see `ACPI Debug Output` for details. By default, the ACPI debug level is set to `ACPI_LV_INIT | ACPI_LV_TABLES | ACPI_LV_VERBOSE_INFO`.

- **trace <trace_mask>**

  Enable tracing of events matching `trace_mask`. The only supported trace mask is `1` and this matches ACPI events.

- **config_files**

  Space separated list of Lua configuration files specifying real hardware and virtual buses. See example on Virtual Bus Description.

5.5 l4vio_net_p2p, a virtual network point-to-point link

The virtual network point-to-point server (p2p) connects two clients with a virtual network connection.

It uses virtio as the transport mechanism. Each virtual network p2p endpoint implements the device-side of a virtio network device. Each client can access its endpoint using the driver-side semantics of a virtio network device.

Building and Configuration

The virtual network p2p server can be built using the L4Re build system by placing this project into the `pkg` directory.

Starting the service

The virtual network p2p server can be started with Lua like this:

```
local p2p = L4.default_loader:new_channel();
L4.default_loader:start(
{
  caps = {
    svr = p2p:svr(),
  },
},
"rom/l4vio_net_p2p [<options>]");
```

First an IPC gate (`p2p`) is created which is used between the virtual network p2p server and a client to create new virtual ports. The server-side is assigned to the mandatory `svr` capability of the virtual network p2p server. See the section below on how to create a new virtual port and connect a client to it.

Options

The following command line options are supported:

- `-p <num_usec>`, `--poll <num_usec>`

  Enable polling mode and set the poll interval. IRQ notification is disabled for queues while in polling mode. Must be a positive integer specified in microseconds.

- `-s <num>`, `--size <num>`

  Set the maximum queue size for the device-side virtio queues. Must be a power of 2 in the range of 1 to 32768 inclusive.

Connecting a client

Prior to connecting a client to a virtual network p2p server port it has to be created using the following Lua function. It has to be called on the client side of the IPC gate capability whose server side is bound to the virtual network p2p server.

The "key=value" pairs passed to create() can be omitted and their order is not important.

```
create(obj_type, ["ds-max=<max>" , "mac-addr=<mac_address>"])
```

- `obj_type`

  The type of object that should be created by the server. The type must be a positive integer. Currently the following objects are supported:

  - `0`: Virtual p2p port

- "ds-max=<max>"

  Specifies the upper limit of the number of dataspaces the client is allowed to register with the server for virtio DMA. Must be in the range of 1 to 80 inclusive. The default value is 2.

- "mac-addr=<mac_address>"

  Specify the MAC address of the endpoint where <mac_address> is of the form X:XX:Xx:x:xx:xX.

If the `create()` call is successful a new capability which references a virtual network p2p server port is returned. A client uses this capability to talk to the virtual network p2p server using the virtio network protocol.

A couple of examples on how to create ports with different properties are listed below.

```
-- two normal ports with at most 4 data spaces
net0 = p2p:create(0, "ds-max=4")
net1 = p2p:create(0, "ds-max=4")
-- normal port with 4 data spaces and MAC address
net0 = p2p:create(0, "ds-max=4", "mac-addr=11:22:33:44:55:66")
```

5.6 Uvmm, the virtual machine monitor

Command Line Options

uvmm provides the following command line options:

- -c, --cmdline=<guest command line>

  Command line that is passed to the guest on boot.

- -k, --kernel=<kernel image name>

  The name of the guest-kernel image file present in the ROM namespace.

- -d, --dtb=<DTB overlay>

  The name of the device tree file present in the ROM namespace.

- -r, --ramdisk=<RAM disk name>

  The name of the RAM disk file present in the ROM namespace

- -b, --rambase=<Base address of the guest RAM>

  Physical start address for the guest RAM. This value is platform specific.

- -D, --debug=[<component>=][level]

  Control the verbosity level of the uvmm. Possible `level` values are: quiet, warn, info, trace

  Using the `component` prefix, the verbosity level of each uvmm component is configurable. The component names are: core, cpu, mmio, irq, dev, pm, vbus_event

  For example, the following command line sets the verbosity of all uvmm components to `info` except for IRQ handling, which is set to `trace`.

  ```
  uvmm -D info -D irq=trace
  ```

- -f, --fault-mode

  Control the handling of guest reads/writes to non-existing memory. Possible values are:

  - `ignore` - Invalid writes are ignored. Invalid reads either return 0 or are skipped. The guest may experience undefined behaviour.
  - `halt` - Halt the VM on the first invalid memory access.

  - `inject` - Try to forward the invalid access to the guest. This is not supported on all architectures. Falls back to `halt` if the error could not be forwarded to the guest.

  Defaults to `ignore`.

- `-q, --quiet`

  Silence all uvmm output.

- `-v, --verbose`

  Increase the verbosity of the uvmm. Repeating the option increases the verbosity by another level.

- `-W, --wakeup-on-system-resume`

  When set, the uvmm resumes when the host system resumes after a suspend call.

- `-i`

  When set, the option forces the guest RAM to be mapped to its corresponding host-physical addresses.

Setting up guest memory

In the most simple setup, memory for the guest can be provided via a simple dataspace. In your ned script, create a new dataspace of the required size and hand it into uvmm as the `ram` capability:

```
local ramds = L4.Env.user_factory:create(L4.Proto.Dataspace, 60 * 1024 * 1024)

L4.default_loader::startv({caps = {ram = ramds:m("rw")}}, "rom/uvmm")
```

The memory will be mapped to the most appropriate place and a memory node added to the device tree, so that the guest can find the memory.

For a more complex setup, the memory can be configured via the device tree. uvmm scans for memory nodes and tries to set up the memory from them. A memory device node should look like this:

```
memory@0 {
  device_type = "memory";
  reg = <0x00000000 0x00100000
         0x00200000 0xffffffff>;
  l4vmm,dscap = "memcap";
  dma-ranges = <>;
};
```

The `device_type` property is mandatory and needs to be set to `memory`.

`l4vmm,dscap` contains the name of the capability containing the dataspace to be used for the RAM. `reg` describe the memory regions to use for the memory. The regions will be filled up to the size of the supplied dataspace. If they are larger, then the remaining area will be cut.

If the optional `dma-ranges` property is given, the host-physical address ranges for the memory regions will be added here. Note that the property is not cleared first, so it should be left empty.

Memory layout

uvmm populates the RAM with the following data:

- kernel binary
- (optional) ramdisk
- (optional) device tree

The kernel binary is put at the predefined address. For ELF binaries, this is an absolute physical address. If the binary supports relative addressing, the binary is put to the requested offset relative to beginning of the first 'memory' region defined in the device tree.

The ramdisk and device tree are placed as far as possible to the end of the regions defined in the first 'memory' node.

If there is a part of RAM that must remain empty, then define an extra memory node for it in the device tree. uvmm only writes to memory in the first memory node it finds.

Warning: uvmm does not touch any unpopulated memory. In particular, it does not ensure that the memory is cleared. It is the responsibility of the provider of the RAM dataspace to make sure that no data leakage can happen. Normally this is not an issue because dataspaces are guaranteed to be cleaned when they are newly created but users should be careful when reusing memory or dataspaces, for example, when restarting the uvmm.

Forwarding hardware resources to the guest

Hardware resources must be specified in two places: the device tree contains the description of all hardware devices the guest could see and the Vbus describes which resources are actually available to the uvmm.

The vbus allows the uvmm access to hardware resources in the same way as any other L4 application. uvmm expects a capability named 'vbus' where it can access its hardware resources. It is possible to leave out the capability for purely virtual guests (Note that this is not actually practical on some architectures. On ARM, for example, the guest needs hardware access to the interrupt controller. Without a 'vbus' capability, interrupts will not work.) For information on how to configure a vbus, see the IO documentation.

The device tree needs to contain the hardware description the guest should see. For hardware devices this usually means to use a device tree that would also be used when running the guest directly on hardware.

On startup, uvmm scans the device tree for any devices that require memory or interrupt resources and compares the required resources with the ones available from its vbus. When all resources are available, it sets up the appropriate forwarding, so that the guest now has direct access to the hardware. If the resources are not available, the device will be marked as 'disabled'. This mechanism allows to work with a standard device tree for all guests in the system while handling the actual resource allocation in a flexible manner via the vbus configuration.

The default mechanism assigns all resources 1:1, i.e. with the same memory address and interrupt number as on hardware. It is also possible to map a hardware device to a different location. In this case, the assignment between vbus device and device tree device must be known in advance and marked in the device tree using the `l4vmm,vbus-dev` property.

The following device will for example be bound with the vbus device with the HID 'l4-test,dev':

```
test@e0000000 {
    compatible = "memdev,bar";
    reg = <0 0xe0000000 0 0x50000>,
          <0 0xe1000000 0 0x50000>;
    l4vmm,vbus-dev = "l4-test,dev";
    interrupts-extended = <&gic 0 139 4>;
};
```

Resources are then matched by name. Memory resources in the vbus must be named `reg0` to `reg9` to match against the address ranges in the device tree `reg` property. Interrupts must be called `irq0` to `irq9` and will be matched against `interrupts` or `interrupts-extended` entries in the device tree. The vbus must expose resources for all resources defined in the device tree entry or the initialisation will fail.

An appropriate IO entry for the above device would thus be:

```
MEM = Io.Hw.Device(function()
    Property.hid = "l4-test,dev"
    Resource.reg0 = Io.Res.mmio(0x41000000, 0x4104ffff)
    Resource.reg1 = Io.Res.mmio(0x42000000, 0x4204ffff)
    Resource.irq0 = Io.Res.irq(134);
end)
```

Please note that HIDs on the vbus are not necessarily unique. If multiple devices with the HID given in `l4vmm,vbus-dev` are available on the vbus, then one device is chosen at random.

If no vbus device with the given HID is available, the device is disabled.

How to enable guest suspend/resume

Note

Currently only supported on ARM. It should work fine with Linux version 4.4 or newer.

Uvmm (partially) implements the power state coordination interface (PSCI), which is the standard ARM power management interface. To make use of this interface, you have to announce its availability to the guest operating system via the device tree like so:

```
psci {
      compatible = "arm,psci-0.2";
      method = "hvc";
};
```

The Linux guest must be configured with at least these options:

```
CONFIG_SUSPEND=y
CONFIG_ARM_PSCI=y
```

How to communicate power management (PM) events

Uvmm can be instructed to inform a PM manager of PM events through the L4::Platform_control interface. To that end, uvmm may be equipped with a `pfc` cap. On suspend, uvmm will call l4_platform_ctl_system_suspend().

The `pfc` cap can also be implemented by IO. In that case the guest can start a machine suspend/shutdown/reboot.

Ram block device support

The example ramdisk works by loading a file system into RAM, which needs RAM block device support to work. In the Linux kernel configuration add: CONFIG_BLK_DEV_RAM=y

Framebuffer support for uvmm/amd64 guests

Uvmm can be instructed to pass along a framebuffer to the Linux guest. To enable this three things need to be done:

1. Configure Linux to support a simple framebuffer by enabling CONFIG_FB_SIMPLE=y CONFIG_X86_↩ SYSFB=y
2. Configure a simple framebuffer device in the device tree (currently only read by uvmm, linearer framebuffer at [0xf0000000 - 0xf1000000])

   simplefb { compatible = "simple-framebuffer"; reg = <0x0 0xf0000000 0x0 0x1000000>; l4vmm,cap = "fb"; };
3. Start a framebuffer instance and connect it to uvmm e.g. – Start fb-drv (but only if we need to) local fbdrv_fb = L4.Env.vesa; if (not fbdrv_fb) then fbdrv_fb = l:new_channel(); l:start({ caps = { vbus = io_busses.fbdrv, fb = fbdrv_fb:svr(), }, log = { "fbdrv", "r" }, }, "rom/fb-drv"); end vmm.start_vm{ ext_caps = { fb = fbdrv_fb }, – ...

Recommended Linux configuration options for uvmm/amd64 guests

The following options are recommended in additon to the amd64 defaults provided by a `make defconfig`:

Virtio support is required to access virtual devices provided by uvmm:

```
CONFIG_VIRTIO=y
CONFIG_VIRTIO_PCI=y
CONFIG_VIRTIO_BLK=y
CONFIG_BLK_MQ_VIRTIO=y
CONFIG_VIRTIO_CONSOLE=y
CONFIG_VIRTIO_INPUT=y
CONFIG_VIRTIO_NET=y
```

It is highly recommended to use the X2APIC, which needs virtualization awareness to work under uvmm:

```
CONFIG_X86_X2APIC=y
CONFIG_PARAVIRT=y
CONFIG_PARAVIRT_SPINLOCKS=y
```

KVM clock for uvmm/amd64 guests

When executing L4Re + uvmm on QEMU, the PIT as clock source is not reliable. The paravirtualized KVM clock provides the guest with a stable clock source.

A KVM clock device is available to the guest, if the device tree contains the corresponding entry:

```
kvm_clock {
    compatible = "kvm-clock";
    reg = <0x0 0x0 0x0 0x0>;
};
```

To make use of this clock, the Linux guest must be built with the following configuration options:

```
CONFIG_HYPERVISOR_GUEST=y
CONFIG_KVM_GUEST=y
CONFIG_PTP_1588_CLOCK_KVM is not set
```

Note: KVM calls besides the KVM clock are unhandled and lead to failure in the uvmm, e.g. vmcall 0x9 for the PTP_1588_CLOCK_KVM.

This is considered a development feature. The KVM clock is not required when running on physical hardware as TSC calibration via the PIT works as expected.

Development notes for amd64

When you are developing on Linux using QEMU please note that nested virtualization support is necessary on your host system to run uvmm guests. Your host Linux version should be 4.12 or greater, **excluding 4.20**.

To have KVM support nested virtualization you need to enable it via:

```
modprobe kvm_intel nested=1
```

uvmm does currently not support AMD-V.

QEMU network setup for a uvmm guest on amd64

qemu-system-x86_64 -M q35 -cpu host -enable-kvm -device intel-iommu -device e1000e,netdev=net0 -netdev bridge,id=net0,br=virbr0

where 'virbr0' is the name of the host's bridge device. The line 'allow virbr0' needs to be present in /etc/qemu/bridge.conf. The bridge can either be created via the network manager or via the command line↩ :

```
brctl addbr virbr0
ip addr add 192.168.124.1/24 dev virbr0
ip link set up dev virbr0
```

In the guest linux with eth0 as network device:

```
ip a a 192.168.124.5/24 dev eth0
ip li se up dev eth0
```

Now the host and guest can ping each other using their respective IPs.

Of course, uvmm needs to be connected to Io and Io needs a vbus configuration for the uvmm client like this:

```
Io.add_vbusses
{
  vm_pci = Io.Vi.System_bus(function ()
    Property.num_msis = 6
    PCI = Io.Vi.PCI_bus(function ()
      pci_net = wrap(Io.system_bus():match("PCI/CC_0200"))
    end)
  end)
}
```

QEMU emulated VirtIO devices and IO-MMU on amd64

QEMU does not route VirtIO devices through the IO-MMU per default. To use QEMU emulated VirtIO devices add the `disable-legacy=on,disable-modern=off,iommu_platform=on` flags to the option list of the device. The e1000e card in the network example above can be replaced with an virtio-net-pci card like this:

```
-device virtio-net-pci,disable-legacy=on,disable-modern=off,
        iommu_platform=on,netdev=net0
```

For more information on VirtIO devices and their options see https://wiki.qemu.org/Features/↩ VT-d.

Using the uvmm monitor interface

Uvmm implements an interface with which parts of the guest's state can be queried and manipulated at runtime. This monitor interface needs to be enabled during compilation as well as during startup of uvmm. This is described in detail below.

Compiling uvmm with monitor interface support

To compile uvmm with monitor interface support pass the `CONFIG_MONITOR=y`, option during the `make` step (or set in in the Makefile.config). This option is available on all architectures but note that the set of available monitor interface features may vary significantly between them. Also note that the monitor interface will always be disabled in release mode, i.e. if `CONFIG_RELEASE_MODE=y`.

Enabling the monitor interface at runtime

When starting a uvmm instance from inside a `ned` script using the `vmm.start_vm` function, the `mon` argument controls whether the monitor interface is enabled at runtime. There are three cases to distinguish:

- `mon=true` (default): The monitor interface is enabled but no server implementing the client side of the monitor interface is started. The monitor interface can still be utilized via `cons` but no readline functionality will be available.
- ‘mon='some_binary’: `If a string is passed as the value of`mon`, the monitor interface is enabled and the string is interpreted as the name of a server binary which implements the client side of the monitor interface. This server is automatically started and has access to a vcon capability named `mon` at startup through which it can make use of the monitor interface. Unless you have written your own server you should specify ‘'uvmm_cli’` which is a server implementing a simple readline interface.
- `mon=false`: The monitor interface is disabled at runtime.

Using the monitor interface

If the monitor interface was enabled you can connect to it via `cons` under the name `mon<n>` where `<n>` is a unique integer for every uvmm instance that is started with the monitor interface enabled (numbered starting from one in order of corresponding `vmm.start_vm` calls). If ‘mon='uvmm_cli’` was specified, readline functionality such as command completion and history will be available. Enter a command followed by enter to run that command. To obtain a list of all available commands issue the `help` command, to obtain usage information for a specific command `foo` issue `help foo`.

Note

Some commands will modify the guests state. Since it should be obvious to which ones this applies this is usually not specifically highlighted. Exercise reasonable caution.

Using the guest debugger

The guest debugger provides monitoring functionality akin to a very bare-bone GDB interface, e.g. guest RAM and page table dumping, breakpointing and single stepping. Additional functionality might be added in the future.

Note

> The guest debugger is currently still under development. The guest debugger may also not be available on all architectures. To check whether the guest debugger is available check if `help dbg` returns usage information.

If the guest debugger is available, you have to manually load it at runtime using the monitor interface. This saves resources if the guest debugger is not used. To enable the guest debugger, issue the `dbg on` monitor command. Once enabled, the guest debugger can not be disabled again.

To list available guest debugger subcommands, issue `dbg help` after `dbg on`.

Note

> When using SMP, most guest debugger subcommands require you to explicitly specify a guest vcpu using an index starting from zero.

5.7 Mag, the GUI Multiplexer

Mag is the default multiplexer for graphics hardware.

It is not, and does not attempt to be, a fully-fledged window manager.

Command Line Options

As Mag's only command line option it supports loading additional plugins via the application's command line. Plugins must be either a Lua file or a shared library. Shared libraries must be named `libmag-$LIBNAME.so`.

Mag Sessions

Mag provides two types of sessions which a client can create via the Factory interface.

1. Mag client session

   A client with a mag client session gets access to the whole screen. The client has to allocate and manage it's own buffers and has to position them on the screen on its own. Mag provides the factory to create client sessions via the capability named `mag`.

2. Client framebuffer session

   For a client framebuffer session mag allocates a view of the requested size and displays it at the requested coordinates on the screen. Mag provides the factory to create framebuffer sessions via the capability named `svc`.

The options described below are options the client provides to the L4::Factory::create() call. These options influence the appearance and behaviour of the newly created session.

Session Factory Options

As a simple nitpicker clone Mag supports the so-called Xray mode. This mode displays all session labels and draws a colored frame around them. The session that currently has the input focus is highlighted. The Xray mode is actived via the special keys *Scroll* or *NEXTSONG*.

Mag allows to define a text label and a color for all client session types. The label and the color are displayed when Mag enters the Xray mode.

- Label Option: **label**

  `l=LABEL`, `label=LABEL` Set the session's text label to LABEL. The label is restricted to a length of 256 characters.

- Color Option: **col**

  `col=COLOR` Set the session's color which is used in Xray mode to tint the session's screen area and the border drawn around it. The argument can be either one of the following letters or a hexdecimal representation of the RGB values.

  - `r`, `R` Red color
  - `g`, `G` Green color
  - `b`, `B` Blue color
  - `w`, `W` White color
  - `y`, `Y` Yellow color
  - `v`, `V` Magenta color

Example

```
-- set label to "Linux" and use a light blue color
fb = mag_client:create(L4.Proto.Goos, "l=Linux", "col=98d9ff");
```

Mag Client Session Options

These options only apply to Mag client sessions.

- Default Background Option: **default-background**

  `dfl-bg`, `default-background` Marks this session as the default background.

Mag Client Framebuffer Session Options

These options only apply to Mag client framebuffer sessions.

- Geometry Option: **geometry**

  `g=GEOMETRY`, `geometry=GEOMETRY` Set the session's geometry and position on the screen. GEOMETRY is provided in an X11-style format, e.g. `g=WIDTHxHEIGHT+X_OFFSET+Y_OFFSET`.

- Focus Option: **focus**

  `focus` Set the focus to this session.

- Collapsed Option: **shaded**

  `shaded` The window is collapsed and only the title bar is visible. The window can be expanded by clicking into the title bar with the middle mouse button. Collapsing and expanding works also independently of this option.

- Fixed Option: **fixed** `fixed` The window cannot be moved on the screen.

- Barheight Option: **barheight**

  `barheight=X` Set the height of the title bar in pixels.

Example

```
-- create a window of 640x480 pixels at position (100,100) on the screen.
fb = mag_fb:create(L4.Proto.Goos, "g=640x480+100+100");
```

5.8 Cons, the Console Multiplexer

cons allows to multiplex console output from different L4 clients to different L4::Vcon-capable in/output servers.

Multiplexers and Frontends

cons is able to connect multiple clients with multiple in/output servers.

Clients are handled by a *multiplexer*. Each multiplexer publishes a server capability that allows to create new client connections. The default multiplexer is normally known under the `cons` capability.

Actual in/output is handled by separate frontends. From the point-of-view of cons, a frontend consists of an IPC channel to a server that speaks an appropriate server protocol. By default the L4.Env.log capability is used.

For clients cons implements the L4::Vcon and the Virtio console interface. The supported frontends is limited to L4::Vcon only.

Command Line Options

cons accepts the following command line switches:

- `-a`, `--show-all`

  Initially show output from all clients.

- `-c <client>`, `--autoconnect <client>`

  Automatically connect to the client with the given name. That means that output of this client will be visible and input will be routed to it.

- `-k`, `--keep`

  Keep the console buffer when a client disconnects.

- `-n`, `--defaultname`

  Default multiplexer capability to use. Default: `cons`.

- `-B <size>`, `--defaultbufsize <size>`

  Default buffer size per client in bytes. Default: 40960

- `-m <prompt name>`, `--mux <prompt name>`

  Add a new multiplexer named <prompt name>. This is necessary if output should be sent to different frontends.

- `-f <cap>`, `--frontend <cap>`

  Set the frontend for the current multiplexer. Output for the multiplexer is then sent to the capability with the given name. The server connected to the capability needs to understand the L4::Vcon protocol.

Connecting a client

```
create(backend_type, ["client_name"], ["color"], ["options"])
```

- `backend_type`

  The type of backend that should be created for the client. The type is a positive integer and currently the following types are supported:

  - `L4.Proto.Log`: L4::Vcon client
  - `1`: Virtio console client

Chapter 6

Bootstrap, the L4 kernel bootstrapper

@subpage Bootstrap Command Line Options

`bootstrap` and the kernel can be configured through command line switches. `bootstrap` is responsible for parsing both command lines: `bootstrap` options are the ones directly given to bootstrap, whereas kernel options are those directly given to the kernel, respectively.

When using Multiboot boot, the first module directly after bootstrap is considered the kernel: An example using GRUB2 might look like this:

```
multiboot /path/to/bootstrap bootstrap -bs-boolean-opt -bs-opt-with-argument=foo
module /path/to/fiasco fiasco -kernel-opt-with-argument=bar -kernel-boolean-opt
```

Note

The exact way to provide the command line to `bootstrap` is platform-dependent. On platforms supporting booting via Multiboot Specification the command line can be specified in the boot configuration (currently x86/amd64 only, e.g. using GRUB) whereas other platforms need the command line to be compiled into the `bootstrap` binary.

Platforms utilising flattened device trees usually also allow specifying a command line through the DT. This mechanism is **not** currently supported in `bootstrap`!

Note

`bootstrap` will ignore options it does not understand. For most cases, this also holds true if an option's arguments cannot be understood.

`bootstrap` options

Command line options directly understood by `bootstrap` itself are as follows (passed via `bootstrap` command line):

- `-comirq=<irqno>` (x86/amd64 only)

  If serial logging is enabled (default on), `<irqno>` defines which IRQ to use for serial port communication. This option is ignored if `-noserial` is also specified (see below).

- -comport=<portspec> (x86/amd64 only)

  If serial logging is enabled (default on), <portspec> defines which serial port to use, being one of:

  - <number>
    Use legacy port <number>, e.g. use -comport=1 for the port commonly known as *COM1*, or
  - pci:<card>:<dev>
    Use serial port number <dev> at PCI card number <card>, e.g. use -comport=pci:0:0 for the first port on the first PCI card.
  - pci:probe
    Use this to have bootstrap autodiscover all PCI serial lines. On each discovered line a message will be displayed, telling the correct <portspec> to be given to use the respective line.

  Note

  bootstrap does not support specifying the serial port's baudrate and always uses 115200 bps.

- -noserial

  Disable serial logging.

- -wait

  Wait for key press at early startup.

  Note

  This is not to be confused with the kernel's -wait option, see below.

- -maxmem=<mbytes>

  Limit the available memory to at most <mbytes> MiB.

- -mem=<size>@<offset>

  Add a region of memory of <size> at <offset> to the system's memory map. Both <size> and <offset> may be suffixed by either G, M, or K/k to denote GiB, MiB, or KiB, respectively.

  This option may be specified multiple times. If the option is not given, a platform-specific method for determining the memory layout will be used, if available.

- -presetmem=<intval>

  Initialise memory regions with <intval> before starting the kernel.

- -modaddr=<paddr>

  Relocate modules to the physical address <paddr>. Use this when utilising a version of GRUB that lacks support for the modaddr command.

Kernel Options

Command line options for the kernel (passed on kernel command line, i.e. to first module) bootstrap understands are as follows.

Note

Availability of individual options might depend on used platform and/or actual kernel configuration.

- -wait

  Enter debugger directly after startup, prior to executing any task.

- -serial_esc

  Enable entering the debugger over serial line by pressing Esc.

- -noserial

  Disable serial logging.

Note

If this is given as kernel command line argument, it does not affect `bootstrap` but only the kernel.

- `-noscreen`

  Disable VGA console.

- `-esc`

  Enable entering the debugger by pressing `Esc` on attached keyboard.

- `-nojdb`

  Disable the kernel debugger.

- `-nohlt`

  Enable quirk for broken HLT instruction.

- `-apic`

  Use Advanced Programmable Interrupt Controller (APIC) if available and known to be well-behaving.

- `-loadcnt`

  Use load counter for performance counting.

- `-watchdog`

  Enable watchdog timer.

- `-irq0`

  Allow IRQ 0 to be used by userland. This enables some sanity checks to ensure IRQ 0 is not used by the kernel, e.g. for profiling or scheduling purposes. This only has an effect on x86.

- `-nosfn`

  Disable SFN (special fully nested) mode of interrupt controller. This only has an effect on x86 with PIC8259 interrupt controller.

- `-jdb_never_stop`

  Prevent system from stopping to enter JDB. This only has an effect on x86.

- `-kmemsize=<kbytes>`

  Reserve `<kbytes>` KiB of memory for the kernel.

- `-tbuf_entries=<number>`

  Specify the `<number>` of trace buffer entries.

- `-out_buf=<length>`

  Specify length of console buffer to be `<length>` bytes.

- `-jdb_cmd=<ctrlseq>`

  Execute JDB command sequence `<ctrlseq>` right after start-up. If `-wait` is also given, `<ctrlseq>` is executed right before entering JDB.

Chapter 7

Deprecated List

Global L4::Irq::unmask (l4_utcb_t *utcb=l4_utcb()) noexcept

Use L4::Irq_eoi::unmask()

Class L4::Irq_mux

IRQ muxer objects are no longer supported by the kernel.

Global l4_irq_mux_chain (l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW

IRQ muxer objects are no longer supported by the kernel.

Global L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (l4_utcb_t *, L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory)

{Note that this variant of the constructor is deprecated, please do not supply the UTCB pointer, it's not used.}

Global L4vbus::Vbus::release_resource (l4vbus_resource_t *res) const

This function is deprecated since Q3 2019. Use release_ioport() instead.

Global L4vbus::Vbus::request_resource (l4vbus_resource_t *res, int=0) const

This function is deprecated since Q3 2019. Use request_ioport() instead.

Global l4vbus_release_resource (l4_cap_idx_t vbus, l4vbus_resource_t const *res)

This function is deprecated since Q3 2019. Use l4vbus_release_ioport() instead.

Global l4vbus_request_resource (l4_cap_idx_t vbus, l4vbus_resource_t const *res, int flags)

This function is deprecated since Q3 2019. Use l4vbus_request_ioport() instead.

Global L4virtio::Device::register_iface (L4::Ipc::Cap< L4::Triggerable > guest_irq, L4::Ipc::Out< L4::Cap< L4::Triggerable > > host_irq, L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > config_ds)

Use device_config(), device_notification_irq() and Icu::bind() instead.

Chapter 8

Module Index

8.1 Modules

Here is a list of all modules:

- Base API . . . 130
  - Basic Macros . . . 132
  - C++ IPC Interface Definition. . . . 146
    - Internal Helpers . . . 301
  - Cache Consistency . . . 149
  - Capabilities . . . 152
  - Error codes . . . 216
  - Fiasco extensions . . . 231
    - Fiasco real time scheduling extensions . . . 236
    - Kernel Debugger . . . 317
  - Flex pages . . . 237
  - Integer Types . . . 289
  - Kernel Interface Page . . . 319
    - Fiasco-UX Virtual devices . . . 236
    - Memory descriptors (C version) . . . 398
  - Kernel Objects . . . 328
    - DMA space . . . 176
    - Factory . . . 225
    - IPC-Gate API . . . 268
    - IRQs . . . 271
    - Interrupt controller . . . 303
    - Kernel-provided semaphore . . . 329
    - L4 kernel object type information . . . 360
    - Platform Control C API . . . 446
    - Scheduler . . . 469
    - Task . . . 494
    - Thread . . . 502
      - Thread control . . . 520
      - vCPU API . . . 592
    - Virtual Console . . . 574
    - Virtual Machines . . . 587
      - VM API for SVM . . . 548
      - VM API for TZ . . . 549
      - VM API for VMX . . . 549

Memory operations. 403
Memory related 405
Object Invocation 432
Error Handling 210
Message Items 411
Message Tag 415
Realtime API 453
Timeouts 525
Virtual Registers (UTCBs) 588
ARM Virtual Registers (UTCB) 109
Buffer Registers (BRs) 143
Message Registers (MRs) 415
Exception registers 221
Thread Control Registers (TCRs) 520
amd64 Virtual Registers (UTCB) 592
x86 Virtual Registers (UTCB) 604
EDID parsing functionality 181
IO interface 261
IRQ handling library 271
Interface for asynchronous ISR handlers. 292
Interface for asynchronous ISR handlers with a given IRQ capability. 291
Interface using direct functionality. 296
Interface using direct functionality. 294
L4 IPC Opcodes 333
L4 VIRTIO Interface 339
L4 VIRTIO Block Device 337
L4 VIRTIO Input Device 338
L4 VIRTIO Transport Layer 340
L4 Vbus functions 348
L4vbus GPIO functions 373
L4vbus PCI functions 384
L4Re C Interface 361
Capability allocator 156
DMA Space Interface 172
Dataspace interface 176
Debug interface 179
Event interface 218
Initial Environment 284
Kumem allocator utility 331
L4Re Util C Interface 372
Log interface 389
Memory allocator 394
Namespace interface 428
Region map interface 454
Video API 565
L4Re C++ Interface 364
Auxiliary data 129
C++ Exceptions 145
Console API 164
Debugging API 180
Event API 217
L4Re ELF Auxiliary Information 369
L4Re Protocol identifiers 371
L4Re Util C++ Interface 372
Kumem utiltities 332
L4Re Capability API 366

Logging interface . . . 393
Name-space API . . . 428
Parent API . . . 445
Region map API . . . 453
Vbus API . . . 563
Server-Side IPC framework . . . 474
Shared Memory Library . . . 476
Chunks . . . 157
Consumer . . . 165
Producer . . . 449
Signals . . . 487
Consumer . . . 168
Producer . . . 451
Sigma0 API . . . 481
Internal constants . . . 301
Small C++ Template Library . . . 491
Utility Functions . . . 541
Atomic Instructions . . . 110
Bit Manipulation . . . 135
Bitmap graphics and fonts . . . 143
Functions for rendering bitmap data in frame buffers . . . 254
Functions for rendering bitmap fonts to frame buffers . . . 254
CPU related functions . . . 146
Comfortable Command Line Parsing . . . 162
ELF binary format . . . 185
Functions to manipulate the local IDT . . . 255
IA32 Port I/O API . . . 255
Internal functions . . . 303
Kernel Interface Page API . . . 326
Low-Level Thread Functions . . . 393
Random number support . . . 452
Timestamp Counter . . . 533
vCPU Support Library . . . 596
Extended vCPU support . . . 224

Chapter 9

Namespace Index

9.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

cxx
Our C++ library 607
cxx::Bits
Internal helpers for the cxx package 609
L4
L4 low-level kernel interface 610
L4::Ipc
IPC related functionality 620
L4::Ipc::Msg
IPC Message related functionality 628
L4::Ipc_svr
Helper classes for L4::Server instantiation 633
L4::Typeid
Definition of interface data-type helpers 634
L4::Types
L4 basic type helpers for C++ 635
L4Re
L4Re C++ Interfaces 636
L4Re::Util
Documentation of the L4 Runtime Environment utility functionality in C++ 651
L4Re::Vfs
Virtual file system for interfaces POSIX libc 662
L4vbus
C++ interface of the Vbus API. 663
L4virtio
L4-VIRTIO Transport C++ API 664

Chapter 10

Hierarchical Index

10.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

L4::Ipc::Array_ref< char const, unsigned long > . . . 947
  L4::Ipc::Array< char const, unsigned long > . . . 943
L4::Ipc::Array_ref< ELEM_TYPE, Array_len_default > . . . 947
  L4::Ipc::Array< ELEM_TYPE, LEN_TYPE > . . . 943
cxx::Bits::Base_avl_set< ITEM_TYPE, Lt_functor< ITEM_TYPE >, New_allocator, Bits::Avl_set_get_key< ITEM_TYPE > > . . . 729
  cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > . . . 674
cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, Lt_functor< KEY_TYPE >, New_allocator, Bits::Avl_map_get_key< KEY_TYPE > > . . . 729
  cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > . . . 669
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > . . . 685
cxx::Base_slab< sizeof(Type), L4_PAGESIZE, 2, New_allocator > . . . 685
  cxx::Slab< Type, Slab_size, Max_free, Alloc > . . . 815
cxx::Base_slab_static< sizeof(Type), L4_PAGESIZE, 2, New_allocator > . . . 692
  cxx::Slab_static< Type, Slab_size, Max_free, Alloc > . . . 818
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item > > . . . 745
  cxx::H_list< T, POLICY > . . . 772
    cxx::H_list_t< Weak_ref_base > . . . 783
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > > . . . 745
  cxx::S_list< T, Bits::Basic_list_policy< T, S_list_item > > . . . 811
  cxx::S_list< T, POLICY > . . . 811
L4::Types::Bool< __Iface_conflict< I, I2 >::value||_Iface_conflict< I, LIST::type >::value > . . . 1226
L4::Types::Bool< false > . . . 1226
  L4::Types::False . . . 1228
    L4::Types::Same< A, B > . . . 1240
L4::Types::Bool< I1::Proto !=PROTO_EMPTY &&I1::Proto==I2::Proto > . . . 1226
L4::Types::Bool< Iface_conflict< I, L2::type >::value||_Conflict< L1::type, L2::type >::value > . . . 1226
L4::Types::Bool< true > . . . 1226
  L4::Types::True . . . 1242
    L4::Ipc::Msg::Is_valid_rpc_type< A * > . . . 996
    L4::Ipc::Msg::Is_valid_rpc_type< T > . . . 996

L4::Types::Bool< Typeid::Conflict< L1::type, L2::type >::value||Conflict< L1, LIST... >::value||Conflict< L2, LIST... >::value > . . . 1226
L4::Types::Bool< Typeid::Iface_conflict< I::type, L::type >::value||Iface_conflict< I, LIST... >::value > . 1226
cxx::Bits::Bst< Node, Get_key, Lt_functor< typename Get_key::Key_type > > . . . 749
cxx::Avl_tree< Node, Get_key, Compare > . . . 677
L4::Ipc::Msg::Clnt_val_ops< Detail::_Plain< T >::type, DIR, CLASS > . . . 980
L4Re::Util::Counter< unsigned char > . . . 1399
cxx::Auto_ptr< T > . . . 665
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > . . . 685
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > . . . 692
cxx::Bitfield< T, LSB, MSB > . . . 698
cxx::Bitfield< T, LSB, MSB >::Value_base< TT > . . . 710
cxx::Bitfield< T, LSB, MSB >::Value< TT > . . . 709
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > . . . 712
cxx::Bitmap_base . . . 717
cxx::Bitmap< BITS > . . . 713
cxx::Bitmap_base::Bit . . . 725
cxx::Bitmap_base::Char< BITS > . . . 726
cxx::Bitmap_base::Word< BITS > . . . 726
cxx::Bits::Avl_map_get_key< KEY_TYPE > . . . 728
cxx::Bits::Avl_set_get_key< KEY_TYPE > . . . 728
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > . . . 729
cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc > . . . 669
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node . . . 743
cxx::Bits::Basic_list< POLICY > . . . 745
cxx::H_list< Timeout > . . . 772
cxx::H_list< Observer > . . . 772
cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > > . . . 772
cxx::H_list_t< T > . . . 783
cxx::H_list< cxx::Base_slab::Slab_i > . . . 772
cxx::Bits::Bst< Node, Get_key, Compare > . . . 749
cxx::Avl_tree< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, Lt_functor< ITEM_TYPE > > . . . 677
cxx::Avl_tree< Entry, Names_get_key > . . . 677
cxx::Avl_tree< _Node, GET_KEY, COMPARE > . . . 677
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< KEY_TYPE >, Lt_functor< KEY_TYPE > > . . . 677
cxx::Bits::Bst_node . . . 763
cxx::Avl_tree_node . . . 682
cxx::Bits::Direction . . . 765
cxx::Bits::Smart_ptr_list< ITEM > . . . 768
cxx::Bits::Smart_ptr_list_item< T, STORE_T > . . . 770
cxx::Ref_obj_list_item< T > . . . 806
cxx::H_list_item_t< ELEM_TYPE > . . . 780
L4::Ipc_svr::Timeout . . . 1051
cxx::List< D, Alloc > . . . 786
cxx::List< D, Alloc >::Iter . . . 788
cxx::List_alloc . . . 789
cxx::List_item . . . 792
cxx::List_item::Iter . . . 795
cxx::List_item::T_iter< E > . . . 797
cxx::List_item::T_iter< T, Poly > . . . 797
cxx::Lt_functor< Obj > . . . 800
cxx::New_allocator< _Type > . . . 800
cxx::Nothrow . . . 802
cxx::Pair< First, Second > . . . 802
cxx::Pair_first_compare< Cmp, Typ > . . . 804

cxx::Ref_ptr< T, CNT > . . . 807
cxx::static_vector< T, IDX > . . . 823
cxx::String . . . 824
L4Re::Util::Names::Name . . . 1428
L4virtio::Svr::Device_t< Ds_data > . . . 1687
L4virtio::Svr::Block_dev_base< Ds_data > . . . 1656
L4virtio::Svr::Driver_mem_list_t< Ds_data > . . . 1694
L4virtio::Svr::Driver_mem_region_t< Ds_data > . . . 1704
Elf32_Auxv . . . 836
Elf32_Dyn . . . 837
Elf32_Ehdr . . . 838
Elf32_Phdr . . . 840
Elf32_Rel . . . 842
Elf32_Rela . . . 843
Elf32_Shdr . . . 843
Elf32_Sym . . . 845
Elf64_Auxv . . . 846
Elf64_Dyn . . . 847
Elf64_Ehdr . . . 848
Elf64_Phdr . . . 851
Elf64_Rel . . . 853
Elf64_Rela . . . 854
Elf64_Shdr . . . 854
Elf64_Sym . . . 856
L4::Epiface_t0< IFACE, BASE > . . . 904
L4::Epiface_t0< void, BASE > . . . 904
L4Re::Event_buffer_t< PAYLOAD > . . . 1335
L4Re::Util::Event_buffer_t< PAYLOAD > . . . 1417
L4Re::Util::Event_buffer_consumer_t< PAYLOAD > . . . 1414
L4::Types::Flags_ops_t< Flags > . . . 1234
L4::Types::Flags_ops_t< Flags_t< DT, T > > . . . 1234
L4::Types::Flags_t< DT, T > . . . 1236
gfxbitmap_offset . . . 857
cxx::H_list_item_t< Weak_ref_base > . . . 780
cxx::Weak_ref_base . . . 833
cxx::Weak_ref< T > . . . 831
L4::Kobject_2t< Console, Video::Goos, Event, L4::PROTO_EMPTY > . . . 1097
L4Re::Console . . . 1299
L4::Kobject_2t< Debug_obj_t< BASE >, BASE, Debug_obj, L4::PROTO_EMPTY > . . . 1097
L4::Kobject_x< Iommu, Proto_t< L4_PROTO_IOMMU >, Type_info::Demand_t< 1 > > . . . 1111
L4::Iommu . . . 939
L4::Alloc_list . . . 858
L4::Basic_registry . . . 863
L4Re::Util::Object_registry . . . 1433
L4::Cap_base . . . 873
L4::Cap< L4::Vcon > . . . 868
L4::Cap< L4::Factory > . . . 868
L4::Cap< L4Re::Rm > . . . 868
L4::Cap< L4::Irq > . . . 868
L4::Cap< L4::Thread > . . . 868
L4::Cap< L4Re::Namespace > . . . 868
L4::Cap< void > . . . 868
L4::Cap< L4::Semaphore > . . . 868
L4::Cap< L4Re::Dataspace > . . . 868
L4::Cap< L4virtio::Device > . . . 868
L4::Cap< L4vbus::Vbus > . . . 868

L4::Cap< L4Re::Video::Goos > . . . 868
L4::Cap< T > . . . 868
L4::Smart_cap< T, SMART > . . . 1168
L4::Epiface . . . 897
L4::Epiface_t0< void, Epiface > . . . 904
L4::Irqep_t< Irq_object > . . . 1080
L4::Irqep_t< Derived, BASE, bool > . . . 1080
L4::Epiface_t0< IFACE, L4::Epiface > . . . 904
L4::Epiface_t< Derived, IFACE, BASE, bool > . . . 901
L4::Epiface_t0< RPC_IFACE, BASE > . . . 904
L4::Epiface_t< Block_dev< Ds_data >, L4virtio::Device > . . . 901
L4::Epiface_t< Null_handler, L4::Kobject > . . . 901
L4::Server_object . . . 1156
L4::Server_object_t< Kobject > . . . 1161
L4::Irq_handler_object . . . 1074
L4::Server_object_t< IFACE, L4::Server_object > . . . 1161
L4::Server_object_x< Derived, IFACE, BASE > . . . 1165
L4::Server_object_t< IFACE, BASE > . . . 1161
L4::Server_object_x< Derived, IFACE, BASE > . . . 1165
L4::Exception_tracer . . . 907
L4::Base_exception . . . 861
L4::Invalid_capability . . . 934
L4::Runtime_error . . . 1139
L4::Bounds_error . . . 865
L4::Com_error . . . 883
L4::Element_already_exists . . . 893
L4::Element_not_found . . . 895
L4::Out_of_memory . . . 1116
L4::Unknown_error . . . 1244
L4::Factory::Lstr . . . 917
L4::Factory::Nil . . . 919
L4::Factory::S . . . 920
L4::IOModifier . . . 942
L4::Ipc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > . . . 946
L4::Ipc::Array_ref< ELEM_TYPE, LEN_TYPE > . . . 947
L4::Ipc::As_value< T > . . . 949
L4::Ipc::Buf_item . . . 950
L4::Ipc::Call . . . 951
L4::Ipc::Call_t< RIGHTS > . . . 952
L4::Ipc::Call_zero_send_timeout . . . 953
L4::Ipc::Cap< T > . . . 955
L4::Ipc::Gen_fpage< T > . . . 957
L4::Ipc::In_out< T > . . . 960
L4::Ipc::Istream . . . 970
L4::Ipc::Iostream . . . 961
L4::Ipc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS > . . . 980
L4::Ipc::Msg::Cls_buffer . . . 982
L4::Ipc::Msg::Do_rcv_buffers . . . 991
L4::Ipc::Msg::Cls_data . . . 983
L4::Ipc::Msg::Do_in_data . . . 987
L4::Ipc::Msg::Do_out_data . . . 989
L4::Ipc::Msg::Cls_item . . . 984
L4::Ipc::Msg::Do_in_items . . . 988
L4::Ipc::Msg::Do_out_items . . . 990
L4::Ipc::Msg::Dir_in . . . 985

L4::Ipc::Msg::Do_in_data . . . 987
L4::Ipc::Msg::Do_in_items . . . 988
L4::Ipc::Msg::Do_rcv_buffers . . . 991
L4::Ipc::Msg::Dir_out . . . 986
L4::Ipc::Msg::Do_out_data . . . 989
L4::Ipc::Msg::Do_out_items . . . 990
L4::Ipc::Msg::Elem< Array< A, LEN > & > . . . 993
L4::Ipc::Msg::Elem< Array< A, LEN > > . . . 994
L4::Ipc::Msg::Elem< Array_ref< A, LEN > & > . . . 995
L4::Ipc::Msg::Svr_arg_pack< IPC_TYPE > . . . 998
L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > . . . 998
L4::Ipc::Msg_ptr< T > . . . 1000
L4::Ipc::Opt< T > . . . 1001
L4::Ipc::Ostream . . . 1002
L4::Ipc::Iostream . . . 961
L4::Ipc::Out< T > . . . 1008
L4::Ipc::Ret_array< T > . . . 1009
L4::Ipc::Send_only . . . 1010
L4::Ipc::Small_buf . . . 1011
L4::Ipc::Snd_item . . . 1012
L4::Ipc::Str_cp_in< T > . . . 1013
L4::Ipc::Varg . . . 1014
L4::Ipc::Varg_list_ref . . . 1022
L4::Ipc::Varg_list< MAX > . . . 1020
L4::Ipc::Varg_list_ref::Iterator . . . 1024
L4::Ipc_svr::Compound_reply . . . 1032
L4::Ipc_svr::Default_loop_hooks . . . 1033
L4::Server< L4::Ipc_svr::Default_loop_hooks > . . . 1151
L4Re::Util::Registry_server< LOOP_HOOKS > . . . 1441
L4::Server< LOOP_HOOKS > . . . 1151
L4Re::Util::Br_manager_hooks . . . 1394
L4::Ipc_svr::Default_setup_wait . . . 1035
L4::Ipc_svr::Default_timeout . . . 1036
L4::Ipc_svr::Default_loop_hooks . . . 1033
L4Re::Util::Br_manager_hooks . . . 1394
L4::Ipc_svr::Direct_dispatch< R > . . . 1038
L4::Ipc_svr::Exc_dispatch< R, Exc > . . . 1041
L4::Ipc_svr::Direct_dispatch< R * > . . . 1040
L4::Ipc_svr::Ignore_errors . . . 1042
L4::Ipc_svr::Default_loop_hooks . . . 1033
L4Re::Util::Br_manager_hooks . . . 1394
L4Re::Util::Br_manager_timeout_hooks . . . 1395
L4::Ipc_svr::Server_iface . . . 1044
L4::Ipc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager > . . . 1060
L4Re::Util::Br_manager_timeout_hooks . . . 1395
L4::Ipc_svr::Br_manager_no_buffers . . . 1028
L4::Ipc_svr::Default_loop_hooks . . . 1033
L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > . . . 1060
L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > . . . 1060
L4Re::Util::Br_manager . . . 1388
L4Re::Util::Br_manager_hooks . . . 1394
L4::Ipc_svr::Timeout_queue . . . 1054
L4::Kip::Mem_desc . . . 1084
L4::Kobject . . . 1094
L4::Kobject_t< Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE, L4::Type_info::Demand_t< 1 > > . . . 1105

L4Re::Namespace . . . 1356
L4::Kobject_t< Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG > . . . 1105
L4Re::Debug_obj . . . 1311
L4::Kobject_t< Thread, Kobject, L4_PROTO_THREAD, Type_info::Demand_t< 1 > > . . . 1105
L4::Thread . . . 1180
L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > > . . . 1105
L4::Rcv_endpoint . . . 1131
L4::Kobject_2t< Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER > . . . 1097
L4::Irq . . . 1065
L4::Kobject_t< Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > > 1105
L4::Ipc_gate . . . 1025
L4::Kobject_t< Goos, L4::Kobject, L4RE_PROTO_GOOS > . . . 1105
L4Re::Video::Goos . . . 1495
L4::Kobject_t< Derived, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> > . . . 1105
L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY > . . . 1105
L4::Factory . . . 909
L4::Kobject_t< Mem_alloc, L4::Factory, L4::PROTO_EMPTY > . . . 1105
L4Re::Mem_alloc . . . 1347
L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR > . . . 1105
L4Re::Inhibitor . . . 1339
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event > . . . 1101
L4vbus::Vbus . . . 1587
L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > > . . . 1105
L4Re::Dataspace . . . 1301
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event > . . . 1101
L4::Kobject_t< Platform_control, Kobject, L4_PROTO_PLATFORM_CTL > . . . 1105
L4::Platform_control . . . 1123
L4::Kobject_t< Meta, Kobject, L4_PROTO_META > . . . 1105
L4::Meta . . . 1112
L4::Kobject_t< Debugger, Kobject, L4_PROTO_DEBUGGER > . . . 1105
L4::Debugger . . . 886
L4::Kobject_t< Parent, L4::Kobject, L4RE_PROTO_PARENT > . . . 1105
L4Re::Parent . . . 1364
L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > > . . . 1105
L4::Task . . . 1172
L4::Kobject_t< Vm, Task, L4_PROTO_VM > . . . 1105
L4::Vm . . . 1254
L4::Vm . . . 1254
L4::Kobject_t< Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE > . . . 1105
L4Re::Mmio_space . . . 1352
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > . . . 1097
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > . . . 1101
L4::Kobject_demand< T > . . . 1105
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > . . . 1105
L4::Kobject_typeid< T > . . . 1107
L4::Kobject_typeid< void > . . . 1110
L4::Kobject_x< Derived, ARGS > . . . 1111
L4::Poll_timeout_kipclock . . . 1127
L4::Proto_t< P > . . . 1130
L4::Registry_iface . . . 1135
L4Re::Util::Object_registry . . . 1433
L4::String . . . 1171
L4::Thread::Attr . . . 1191
L4::Thread::Modify_senders . . . 1196

L4::Type_info . . . 1201
L4::Type_info::Demand . . . 1202
L4::Type_info::Demand_t< __I::Max< D1::Caps, D2::Caps >::Res, D1::Flags|D2::Flags, __I::Max< D1::Mem, D2::Mem >::Res, __I::Max< D1::Ports, D2::Ports >::Res > . . . 1205
L4::Type_info::Demand_union_t< Kobject_typeid< T1 >::Demand, Kobject_demand< T2... > > . . . 1207
L4::Type_info::Demand_union_t< D1, D2 > . . . 1207
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > . . . 1205
L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y > . . . 1211
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... > . . . 1212
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y > . . . 1213
L4::Typeid::Detail::Rpcs_end . . . 1214
L4::Typeid::Detail::_Rpcs< OPCODE_TYPE, 0, RPCS... > . . . 1210
L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS > . . . 1222
L4::Typeid::Detail::_Rpcs< l4_umword_t, 0, ARG... > . . . 1210
L4::Typeid::Rpcs_sys< ARG > . . . 1224
L4::Typeid::Detail::_Rpcs< L4::Opcode, 0, RPCS... > . . . 1210
L4::Typeid::Rpcs< RPCS > . . . 1219
L4::Typeid::Detail::_Rpcs< void, 0, OPERATION > . . . 1210
L4::Typeid::Rpc_nocode< OPERATION > . . . 1216
L4::Typeid::Detail::_Rpcs< OPCODE, O, X > . . . 1210
L4::Typeid::P_dispatch< LIST > . . . 1215
L4::Typeid::Raw_ipc< CLASS > . . . 1216
L4::Typeid::Rpcs_code< OPCODE_TYPE > . . . 1221
L4::Types::Bool< V > . . . 1226
L4::Types::Flags< BITS_ENUM, UNDERLYING > . . . 1229
L4::Types::Flags_ops_t< DT > . . . 1234
L4::Types::Int_for_size< SIZE, bool > . . . 1238
L4::Types::Int_for_type< T > . . . 1239
l4_buf_regs_t . . . 1257
l4_exc_regs_t . . . 1258
l4_fpage_t . . . 1261
l4_icu_info_t . . . 1261
L4::Icu::Info . . . 932
l4_icu_msi_info_t . . . 1263
l4_kernel_info_mem_desc_t . . . 1264
l4_kernel_info_t . . . 1265
l4_msg_regs_t . . . 1267
l4_msgtag_t . . . 1268
l4_sched_cpu_set_t . . . 1270
l4_sched_param_t . . . 1273
l4_snd_fpage_t . . . 1274
l4_thread_regs_t . . . 1275
l4_timeout_s . . . 1276
l4_timeout_t . . . 1277
l4_vcon_attr_t . . . 1278
l4_vcpu_ipc_regs_t . . . 1280
l4_vcpu_regs_t . . . 1281
l4_vcpu_state_t . . . 1285
L4vcpu::Vcpu . . . 1600
l4_vhw_descriptor . . . 1287
l4_vhw_entry . . . 1289
l4_vm_svm_vmcb_control_area . . . 1290
l4_vm_svm_vmcb_state_save_area . . . 1291
l4_vm_svm_vmcb_state_save_area_seg . . . 1292
l4_vm_svm_vmcb_t . . . 1293
l4_vm_tz_state . . . 1294
L4Re::Cap_alloc . . . 1295

L4Re::Dataspace::F . . . 1309
L4Re::Dataspace::Stats . . . 1311
L4Re::Env . . . 1319
L4Re::Event_buffer_t< PAYLOAD > . . . 1335
L4Re::Event_buffer_t< PAYLOAD >::Event . . . 1338
L4Re::Rm::F . . . 1384
L4Re::Rm::Range . . . 1385
L4Re::Smart_cap_auto< Unmap_flags > . . . 1386
L4Re::Smart_count_cap< Unmap_flags > . . . 1387
L4Re::Util::Cap_alloc_base . . . 1398
L4Re::Util::Counter< COUNTER > . . . 1399
L4Re::Util::Counting_cap_alloc< COUNTERTYPE > . . . 1400
L4Re::Util::Dataspace_svr . . . 1408
L4Re::Util::Event_svr< SVR > . . . 1421
L4Re::Util::Event_t< PAYLOAD > . . . 1423
L4Re::Util::Item_alloc_base . . . 1427
L4Re::Util::Names::Name_space . . . 1429
L4Re::Util::Ref_cap< T > . . . 1439
L4Re::Util::Ref_del_cap< T > . . . 1440
L4Re::Util::Smart_cap_auto< Unmap_flags > . . . 1446
L4Re::Util::Smart_count_cap< Unmap_flags > . . . 1447
L4Re::Util::Vcon_svr< SVR > . . . 1448
L4Re::Util::Video::Goos_svr . . . 1449
L4Re::Vfs::Directory . . . 1460
  L4Re::Vfs::File . . . 1465
    L4Re::Vfs::Be_file . . . 1452
L4Re::Vfs::File_system . . . 1467
  L4Re::Vfs::Be_file_system . . . 1457
L4Re::Vfs::Fs . . . 1469
  L4Re::Vfs::Ops . . . 1480
L4Re::Vfs::Generic_file . . . 1474
  L4Re::Vfs::File . . . 1465
L4Re::Vfs::Mman . . . 1479
  L4Re::Vfs::Ops . . . 1480
L4Re::Vfs::Regular_file . . . 1483
  L4Re::Vfs::File . . . 1465
L4Re::Vfs::Special_file . . . 1488
  L4Re::Vfs::File . . . 1465
L4Re::Video::Color_component . . . 1491
L4Re::Video::Goos::Info . . . 1502
L4Re::Video::Pixel_info . . . 1503
L4Re::Video::View . . . 1512
L4Re::Video::View::Info . . . 1517
l4re_aux_t . . . 1520
l4re_ds_stats_t . . . 1521
l4re_elf_aux_mword_t . . . 1521
l4re_elf_aux_t . . . 1522
l4re_elf_aux_vma_t . . . 1523
l4re_env_cap_entry_t . . . 1524
l4re_env_t . . . 1526
l4re_event_t . . . 1527
l4re_video_color_component_t . . . 1528
l4re_video_goos_info_t . . . 1529
l4re_video_pixel_info_t . . . 1530
l4re_video_view_info_t . . . 1531
l4re_video_view_t . . . 1533

l4util_idt_desc_t . . . 1534
l4util_idt_header_t . . . 1535
l4util_l4mod_info . . . 1536
l4util_l4mod_mod . . . 1537
l4util_mb_addr_range_t . . . 1538
l4util_mb_apm_t . . . 1539
l4util_mb_drive_t . . . 1540
l4util_mb_info_t . . . 1542
l4util_mb_mod_t . . . 1543
l4util_mb_vbe_ctrl_t . . . 1544
l4util_mb_vbe_mode_t . . . 1545
L4vbus::Gpio_module::Pin_slice . . . 1563
L4vbus::Pm< DEC > . . . 1586
l4vbus_device_t . . . 1596
l4vbus_resource_t . . . 1597
L4vcpu::State . . . 1598
L4virtio::Driver::Block_device::Handle . . . 1627
L4virtio::Driver::Device . . . 1628
  L4virtio::Driver::Block_device . . . 1619
L4virtio::Ptr< T > . . . 1650
L4virtio::Svr::Bad_descriptor . . . 1654
L4virtio::Svr::Block_request< Ds_data > . . . 1665
L4virtio::Svr::Data_buffer . . . 1668
L4virtio::Svr::Dev_config . . . 1672
L4virtio::Svr::Dev_features . . . 1683
L4virtio::Svr::Dev_status . . . 1684
L4virtio::Svr::Device_t< DATA > . . . 1687
L4virtio::Svr::Driver_mem_list_t< DATA > . . . 1694
L4virtio::Svr::Driver_mem_region_t< DATA > . . . 1704
L4virtio::Svr::Request_processor . . . 1711
L4virtio::Svr::Virtqueue::Head_desc . . . 1724
L4virtio::Virtqueue . . . 1726
  L4virtio::Driver::Virtqueue . . . 1641
  L4virtio::Svr::Virtqueue . . . 1718
L4virtio::Virtqueue::Avail . . . 1742
L4virtio::Virtqueue::Avail::Flags . . . 1744
L4virtio::Virtqueue::Desc . . . 1745
L4virtio::Virtqueue::Desc::Flags . . . 1747
L4virtio::Virtqueue::Used . . . 1749
L4virtio::Virtqueue::Used::Flags . . . 1750
L4virtio::Virtqueue::Used_elem . . . 1752
l4virtio_block_config_t . . . 1753
l4virtio_block_discard_t . . . 1754
l4virtio_block_header_t . . . 1755
l4virtio_config_hdr_t . . . 1756
l4virtio_config_queue_t . . . 1757
l4virtio_input_absinfo_t . . . 1759
l4virtio_input_config_t . . . 1759
l4virtio_input_devids_t . . . 1760
l4virtio_input_event_t . . . 1761
cxx::New_allocator< _Node > . . . 800
cxx::New_allocator< Node > . . . 800
cxx::New_allocator< Slab_i > . . . 800
L4vbus::Pm< Device > . . . 1586
  L4vbus::Device . . . 1549
    L4vbus::Gpio_module . . . 1557
    L4vbus::Gpio_pin . . . 1564
    L4vbus::Icu . . . 1572

L4vbus::Pci_dev 1575
L4vbus::Pci_host_bridge 1580
L4virtio::Ptr< void > 1650
cxx::Ref_ptr< L4Re::Vfs::File > 807
cxx::Ref_ptr< Mount_tree > 807
L4::Ipc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_in, CLASS > 998
L4::Ipc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_out, CLASS > 998
L4::Ipc::Msg::Svr_val_ops< L4::Ipc::Snd_fpage, Dir_in, CLASS > 998
L4::Ipc::Msg::Svr_val_ops< typename ELEM::svr_type, typename Direction< T >::type, typename Class< typename Detail::_Plain< T >::type >::type > 998
cxx::Bitmap_base::Word< Bits > 726
cxx::Bitmap_base::Word< Size > 726

Chapter 11

Data Structure Index

11.1 Data Structures

Here are the data structures with brief descriptions:

cxx::Auto_ptr< T >
Smart pointer with automatic deletion 665
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >
AVL tree based associative container 669
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >
AVL set for simple compareable items 674
cxx::Avl_tree< Node, Get_key, Compare >
A generic AVL tree 677
cxx::Avl_tree_node
Node of an AVL tree 682
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >
Basic slab allocator 685
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i
Type of a slab 691
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >
Merged slab allocator (allocators for objects of the same size are merged together) 692
cxx::Bitfield< T, LSB, MSB >
Definition for a member (part) of a bit field 698
cxx::Bitfield< T, LSB, MSB >::Value< TT >
Internal helper type 709
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >
Internal helper type 710
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >
Internal helper type 712
cxx::Bitmap< BITS >
A static bit map 713
cxx::Bitmap_base
Basic bitmap abstraction 717
cxx::Bitmap_base::Bit
A writeable bit in a bitmap 725
cxx::Bitmap_base::Char< BITS >
Helper abstraction for a byte contained in the bitmap 726
cxx::Bitmap_base::Word< BITS >
Helper abstraction for a word contained in the bitmap 726
cxx::Bits::Avl_map_get_key< KEY_TYPE >
Key-getter for Avl_map 728

cxx::Bits::Avl_set_get_key< KEY_TYPE >
Internal, key-getter for Avl_set nodes . . . 728
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >
Internal: AVL set with internally managed nodes . . . 729
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node
A smart pointer to a tree item . . . 743
cxx::Bits::Basic_list< POLICY >
Internal: Common functions for all head-based list implementations . . . 745
cxx::Bits::Bst< Node, Get_key, Compare >
Basic binary search tree (BST) . . . 749
cxx::Bits::Bst_node
Basic type of a node in a binary search tree (BST) . . . 763
cxx::Bits::Direction
The direction to go in a binary search tree . . . 765
cxx::Bits::Smart_ptr_list< ITEM >
List of smart-pointer-managed objects . . . 768
cxx::Bits::Smart_ptr_list_item< T, STORE_T >
List item for an arbitrary item in a Smart_ptr_list . . . 770
cxx::H_list< T, POLICY >
General double-linked list of unspecified cxx::H_list_item elements . . . 772
cxx::H_list_item_t< ELEM_TYPE >
Basic element type for a double-linked H_list . . . 780
cxx::H_list_t< T >
Double-linked list of typed H_list_item_t elements . . . 783
cxx::List< D, Alloc >
Doubly linked list, with internal allocation . . . 786
cxx::List< D, Alloc >::Iter
Iterator . . . 788
cxx::List_alloc
Standard list-based allocator . . . 789
cxx::List_item
Basic list item . . . 792
cxx::List_item::Iter
Iterator for a list of ListItem-s . . . 795
cxx::List_item::T_iter< T, Poly >
Iterator for derived classes from ListItem . . . 797
cxx::Lt_functor< Obj >
Generic comparator class that defaults to the less-than operator . . . 800
cxx::New_allocator< _Type >
Standard allocator based on `operator new ()` . . . 800
cxx::Nothrow
Helper type to distinguish the `oeprator new` version that does not throw exceptions . . . 802
cxx::Pair< First, Second >
Pair of two values . . . 802
cxx::Pair_first_compare< Cmp, Typ >
Comparison functor for Pair . . . 804
cxx::Ref_obj_list_item< T >
Item for list linked via cxx::Ref_ptr with default refence counting . . . 806
cxx::Ref_ptr< T, CNT >
A reference-counting pointer with automatic cleanup . . . 807
cxx::S_list< T, POLICY >
Simple single-linked list . . . 811
cxx::Slab< Type, Slab_size, Max_free, Alloc >
Slab allocator for object of type `Type` . . . 815
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >
Merged slab allocator (allocators for objects of the same size are merged together) . . . 818
cxx::static_vector< T, IDX >
Simple encapsulation for a dynamically allocated array . . . 823

cxx::String
Allocation free string class with explicit length field . . . 824
cxx::Weak_ref< T >
Typed weak reference to an object of type T . . . 831
cxx::Weak_ref_base
Generic (base) weak reference to some object . . . 833
Elf32_Auxv
Auxiliary vector (32-bit) . . . 836
Elf32_Dyn
ELF32 dynamic entry . . . 837
Elf32_Ehdr
ELF32 header . . . 838
Elf32_Phdr
ELF32 program header . . . 840
Elf32_Rel
ELF32 relocation entry w/o addend . . . 842
Elf32_Rela
ELF32 relocation entry w/ addend . . . 843
Elf32_Shdr
ELF32 section header . . . 843
Elf32_Sym
ELF32 symbol table entry . . . 845
Elf64_Auxv
Auxiliary vector (64-bit) . . . 846
Elf64_Dyn
ELF64 dynamic entry . . . 847
Elf64_Ehdr
ELF64 header . . . 848
Elf64_Phdr
ELF64 program header . . . 851
Elf64_Rel
ELF64 relocation entry w/o addend . . . 853
Elf64_Rela
ELF64 relocation entry w/ addend . . . 854
Elf64_Shdr
ELF64 section header . . . 854
Elf64_Sym
ELF64 symbol table entry . . . 856
gfxbitmap_offset
Offsets in pmap[] and bmap[] . . . 857
L4::Alloc_list
A simple list-based allocator . . . 858
L4::Arm_smccc
Wrapper for function calls that follow the ARM SMC/HVC calling convention . . . 859
L4::Base_exception
Base class for all exceptions, thrown by the L4Re framework . . . 861
L4::Basic_registry
This registry returns the corresponding server object based on the label of an Ipc_gate . . . 863
L4::Bounds_error
Access out of bounds . . . 865
L4::Cap< T >
C++ interface for capabilities . . . 868
L4::Cap_base
Base class for all kinds of capabilities . . . 873
L4::Com_error
Error conditions during IPC . . . 883
L4::Debugger
C++ kernel debugger API . . . 886

L4::Element_already_exists
Exception for duplicate element insertions . . . 893
L4::Element_not_found
Exception for a failed lookup (element not found) . . . 895
L4::Epiface
Base class for interface implementations . . . 897
L4::Epiface_t< Derived, IFACE, BASE, bool >
Epiface implementation for Kobject-based interface implementations . . . 901
L4::Epiface_t0< RPC_IFACE, BASE >
Epiface mixin for generic Kobject-based interfaces . . . 904
L4::Exception
Exception interface . . . 906
L4::Exception_tracer
Back-trace support for exceptions . . . 907
L4::Factory
C++ Factory interface, see Factory for the C interface . . . 909
L4::Factory::Lstr
Special type to add a pascal string into the factory create stream . . . 917
L4::Factory::Nil
Special type to add a void argument into the factory create stream . . . 919
L4::Factory::S
Stream class for the create() argument stream . . . 920
L4::Icu
C++ Icu interface, see Interrupt controller for the C interface . . . 924
L4::Icu::Info
This class encapsulates information about an ICU . . . 932
L4::Invalid_capability
Indicates that an invalid object was invoked . . . 934
L4::Io_pager
Io_pager interface . . . 937
L4::Iommu
Interface for IO-MMUs used for DMA remapping . . . 939
L4::IOModifier
Modifier class for the IO stream . . . 942
L4::Ipc::Array< ELEM_TYPE, LEN_TYPE >
Array data type for dynamically sized arrays in RPCs . . . 943
L4::Ipc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >
Server-side copy in buffer for Array . . . 946
L4::Ipc::Array_ref< ELEM_TYPE, LEN_TYPE >
Array reference data type for arrays located in the message . . . 947
L4::Ipc::As_value< T >
Pass the argument as plain data value . . . 949
L4::Ipc::Buf_item
RPC warpper for a receive item . . . 950
L4::Ipc::Call
RPC attribute for a standard RPC call . . . 951
L4::Ipc::Call_t< RIGHTS >
RPC attribute for an RPC call with required rights . . . 952
L4::Ipc::Call_zero_send_timeout
RPC attribute for an RPC call, with zero send timeout . . . 953
L4::Ipc::Cap< T >
Capability type for RPC interfaces (see `L4::Cap<T>`) . . . 955
L4::Ipc::Gen_fpage< T >
Generic RPC wrapper for L4 flex-pages . . . 957
L4::Ipc::In_out< T >
Mark an argument as in-out argument . . . 960
L4::Ipc::Iostream
Input/Output stream for IPC [un]marshalling . . . 961

L4::Ipc::Istream
Input stream for IPC unmarshalling . . . 970
L4::Ipc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >
Defines client-side handling of 'MTYPE' as RPC argument . . . 980
L4::Ipc::Msg::Cls_buffer
Marker type for receive buffer values . . . 982
L4::Ipc::Msg::Cls_data
Marker type for data values . . . 983
L4::Ipc::Msg::Cls_item
Marker type for item values . . . 984
L4::Ipc::Msg::Dir_in
Marker type for input values . . . 985
L4::Ipc::Msg::Dir_out
Marker type for output values . . . 986
L4::Ipc::Msg::Do_in_data
Marker for Input data . . . 987
L4::Ipc::Msg::Do_in_items
Marker for Input items . . . 988
L4::Ipc::Msg::Do_out_data
Marker for Output data . . . 989
L4::Ipc::Msg::Do_out_items
Marker for Output items . . . 990
L4::Ipc::Msg::Do_rcv_buffers
Marker for receive buffers . . . 991
L4::Ipc::Msg::Elem< Array< A, LEN > & >
Array as output argument . . . 993
L4::Ipc::Msg::Elem< Array< A, LEN > >
Array as input arguments . . . 994
L4::Ipc::Msg::Elem< Array_ref< A, LEN > & >
Array_ref as output argument . . . 995
L4::Ipc::Msg::Is_valid_rpc_type< T >
Type trait defining a valid RPC parameter type . . . 996
L4::Ipc::Msg::Svr_arg_pack< IPC_TYPE >
Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function . . . 998
L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >
Defines server-side handling for `MTYPE` server arguments . . . 998
L4::Ipc::Msg_ptr< T >
Pointer to an element of type T in an Ipc::Istream . . . 1000
L4::Ipc::Opt< T >
Attribute for defining an optional RPC argument . . . 1001
L4::Ipc::Ostream
Output stream for IPC marshalling . . . 1002
L4::Ipc::Out< T >
Mark an argument as a output value in an RPC signature . . . 1008
L4::Ipc::Ret_array< T >
Dynamically sized output array of type T . . . 1009
L4::Ipc::Send_only
RPC attribute for a send-only RPC . . . 1010
L4::Ipc::Small_buf
A receive item for receiving a single capability . . . 1011
L4::Ipc::Snd_item
RPC wrapper for a send item . . . 1012
L4::Ipc::Str_cp_in< T >
Abstraction for extracting a zero-terminated string from an Ipc::Istream . . . 1013
L4::Ipc::Varg
Variably sized RPC argument . . . 1014

L4::Ipc::Varg_list< MAX >
Self-contained list of variable-sized RPC parameters . . . 1020
L4::Ipc::Varg_list_ref
List of variable-sized RPC parameters as received by the server . . . 1022
L4::Ipc::Varg_list_ref::Iterator
Iterator for Valists . . . 1024
L4::Ipc_gate
The C++ IPC gate interface, see IPC-Gate API for the C interface . . . 1025
L4::Ipc_svr::Br_manager_no_buffers
Empty implementation of Server_iface . . . 1028
L4::Ipc_svr::Compound_reply
Mix in for LOOP_HOOKS to always use compound reply and wait . . . 1032
L4::Ipc_svr::Default_loop_hooks
Default LOOP_HOOKS . . . 1033
L4::Ipc_svr::Default_setup_wait
Mix in for LOOP_HOOKS for setup_wait no op . . . 1035
L4::Ipc_svr::Default_timeout
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout . . . 1036
L4::Ipc_svr::Direct_dispatch< R >
Direct dispatch helper, for forwarding dispatch calls to a registry *R* . . . 1038
L4::Ipc_svr::Direct_dispatch< R * >
Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry *R* . . . 1040
L4::Ipc_svr::Exc_dispatch< R, Exc >
Dispatch helper wrapping try {} catch {} around the dispatch call . . . 1041
L4::Ipc_svr::Ignore_errors
Mix in for LOOP_HOOKS to ignore IPC errors . . . 1042
L4::Ipc_svr::Server_iface
Interface for server-loop related functions . . . 1044
L4::Ipc_svr::Timeout
Callback interface for Timeout_queue . . . 1051
L4::Ipc_svr::Timeout_queue
Timeout queue to be used in l4re server loop . . . 1054
L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >
Loop hooks mixin for integrating a timeout queue into the server loop . . . 1060
L4::Irq
C++ Irq interface, see IRQs for the C interface . . . 1065
L4::Irq_eoi
Interface for sending an unmask message to an object . . . 1071
L4::Irq_handler_object
Server object base class for handling IRQ messages . . . 1074
L4::Irq_mux
IRQ multiplexer for shared IRQs . . . 1076
L4::Irqep_t< Derived, BASE, bool >
Epiface implementation for interrupt handlers . . . 1080
L4::Kip::Mem_desc
Memory descriptors stored in the kernel interface page . . . 1084
L4::Kobject
Base class for all kinds of kernel objects and remote objects, referenced by capabilities . . . 1094
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >
Helper class to create an L4Re interface class that is derived from two base classes (see `L4::Kobject_t`) . . . 1097
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >
Helper class to create an L4Re interface class that is derived from three base classes (see `L4::Kobject_t`) . . . 1101
L4::Kobject_demand< T >
Get the combined server-side resource requirements for all type T.. . . . 1105
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >
Helper class to create an L4Re interface class that is derived from a single base class . . . 1105

L4::Kobject_typeid< T >
Meta object for handling access to type information of Kobjects 1107
L4::Kobject_typeid< void >
Minimalistic ID for `void` interface 1110
L4::Kobject_x< Derived, ARGS >
Generic Kobject inheritance template 1111
L4::Meta
Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects 1112
L4::Out_of_memory
Exception signalling insufficient memory 1116
L4::Pager
Pager interface including the Io_pager interface 1119
L4::Platform_control
L4 C++ interface for controlling platform-wide properties, see Platform Control C API for the C interface 1123
L4::Poll_timeout_kipclock
A polling timeout based on the L4Re clock 1127
L4::Proto_t< P >
Data type for defining protocol numbers 1130
L4::Rcv_endpoint
Interface for kernel objects that allow to receive IPC from them 1131
L4::Registry_iface
Abstract interface for object registries 1135
L4::Runtime_error
Exception for an abstract runtime error 1139
L4::Scheduler
C++ interface of the Scheduler kernel object, see Scheduler for the C interface 1142
L4::Semaphore
C++ Kernel-provided semaphore interface, see Kernel-provided semaphore for the C interface 1147
L4::Server< LOOP_HOOKS >
Basic server loop for handling client requests 1151
L4::Server_object
Abstract server object to be used with L4::Server and L4::Basic_registry 1156
L4::Server_object_t< IFACE, BASE >
Base class (template) for server implementing server objects 1161
L4::Server_object_x< Derived, IFACE, BASE >
Helper class to implement p_dispatch based server objects 1165
L4::Smart_cap< T, SMART >
Smart capability class 1168
L4::String
A null-terminated string container class 1171
L4::Task
C++ interface of the Task kernel object, see Task for the C interface 1172
L4::Thread
C++ L4 kernel thread interface, see Thread for the C interface 1180
L4::Thread::Attr
Thread attributes used for control_commit() 1191
L4::Thread::Modify_senders
Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread 1196
L4::Triggerable
Interface that allows an object to be triggered by some source 1198
L4::Type_info
Dynamic Type Information for L4Re Interfaces 1201
L4::Type_info::Demand
Data type for expressing the needed receive buffers at the server-side of an interface 1202

L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >
Template type statically describing demand of receive buffers . . . 1205
L4::Type_info::Demand_union_t< D1, D2 >
Template type statically describing the combination of two Demand object . . . 1207
L4::Typeid::Detail::_Rpcs< OPCODE, O, X >
Empty list of RPCs . . . 1210
L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y >
Find the given RPC in the list . . . 1211
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >
Non-empty list of RPCs . . . 1212
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y >
Find the given RPC in the list . . . 1213
L4::Typeid::Detail::Rpcs_end
Internal end-of-list marker . . . 1214
L4::Typeid::P_dispatch< LIST >
Use for protocol based dispatch stage . . . 1215
L4::Typeid::Raw_ipc< CLASS >
RPCs list for passing raw incoming IPC to the server object . . . 1216
L4::Typeid::Rpc_nocode< OPERATION >
List of RPCs of an interface using a single operation without an opcode . . . 1216
L4::Typeid::Rpcs< RPCS >
Standard list of RPCs of an interface . . . 1219
L4::Typeid::Rpcs_code< OPCODE_TYPE >
List of RPCs of an interface using a special opcode type . . . 1221
L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS > . . . 1222
L4::Typeid::Rpcs_sys< ARG >
List of RPCs typically used for kernel interfaces . . . 1224
L4::Types::Bool< V >
Boolean meta type . . . 1226
L4::Types::False
False meta value . . . 1228
L4::Types::Flags< BITS_ENUM, UNDERLYING >
Template for defining typical Flags bitmaps . . . 1229
L4::Types::Flags_ops_t< DT >
Mixin class to define a set of friend bitwise operators on `DT` . . . 1234
L4::Types::Flags_t< DT, T >
Template type to define a flags type with bitwise operations . . . 1236
L4::Types::Int_for_size< SIZE, bool >
Metafunction to get an unsigned integral type for the given size . . . 1238
L4::Types::Int_for_type< T >
Metafunction to get an integral type of the same size as `T` . . . 1239
L4::Types::Same< A, B >
Compare two data types for equality . . . 1240
L4::Types::True
True meta value . . . 1242
L4::Unknown_error
Exception for an unknown condition . . . 1244
L4::Vcon
C++ L4 Vcon interface, see Virtual Console for the C interface . . . 1247
L4::Vm
Virtual machine host address space . . . 1254
l4_buf_regs_t
Encapsulation of the buffer-registers block in the UTCB . . . 1257
l4_exc_regs_t
UTCB structure for exceptions . . . 1258
l4_fpage_t
L4 flexpage type . . . 1261

l4_icu_info_t
Info structure for an ICU . . . 1261
l4_icu_msi_info_t
Info to use for a specific MSI . . . 1263
l4_kernel_info_mem_desc_t
Memory descriptor data structure . . . 1264
l4_kernel_info_t
L4 Kernel Interface Page . . . 1265
l4_msg_regs_t
Encapsulation of the message-register block in the UTCB . . . 1267
l4_msgtag_t
Message tag data structure . . . 1268
l4_sched_cpu_set_t
CPU sets . . . 1270
l4_sched_param_t
Scheduler parameter set . . . 1273
l4_snd_fpage_t
Send-flex-page types . . . 1274
l4_thread_regs_t
Encapsulation of the thread-control-register block of the UTCB . . . 1275
l4_timeout_s
Basic timeout specification . . . 1276
l4_timeout_t
Timeout pair . . . 1277
l4_vcon_attr_t
Vcon attribute structure . . . 1278
l4_vcpu_ipc_regs_t
VCPU message registers . . . 1280
l4_vcpu_regs_t
VCPU registers . . . 1281
l4_vcpu_state_t
State of a vCPU . . . 1285
l4_vhw_descriptor
Virtual hardware devices description . . . 1287
l4_vhw_entry
Description of a device . . . 1289
l4_vm_svm_vmcb_control_area
VMCB structure for SVM VMs . . . 1290
l4_vm_svm_vmcb_state_save_area
State save area structure for SVM VMs . . . 1291
l4_vm_svm_vmcb_state_save_area_seg
State save area segment selector struct . . . 1292
l4_vm_svm_vmcb_t
Control structure for SVM VMs . . . 1293
l4_vm_tz_state
State structure for TrustZone VMs . . . 1294
L4Re::Cap_alloc
Capability allocator interface . . . 1295
L4Re::Console
Console class . . . 1299
L4Re::Dataspace
Interface for memory-like objects . . . 1301
L4Re::Dataspace::F
Dataspace flags definitions . . . 1309
L4Re::Dataspace::Stats
Information about the dataspace . . . 1311
L4Re::Debug_obj
Debug interface . . . 1311

L4Re::Dma_space
DMA Address Space . . . 1314
L4Re::Env
C++ interface of the initial environment that is provided to an L4 task . . . 1319
L4Re::Event
Event class . . . 1332
L4Re::Event_buffer_t< PAYLOAD >
Event buffer class . . . 1335
L4Re::Event_buffer_t< PAYLOAD >::Event
Event structure used in buffer . . . 1338
L4Re::Inhibitor
Set of inhibitor locks, which inhibit specific actions when held . . . 1339
L4Re::Log
Log interface class . . . 1344
L4Re::Mem_alloc
Memory allocation interface . . . 1347
L4Re::Mmio_space
Interface for memory-like address space accessible via IPC . . . 1352
L4Re::Namespace
Name-space interface . . . 1356
L4Re::Ned::Cmd_control
Direct control interface for Ned . . . 1362
L4Re::Parent
Parent interface . . . 1364
L4Re::Random
Low-bandwidth interface for random number generators . . . 1367
L4Re::Rm
Region map . . . 1371
L4Re::Rm::F
Rm flags definitions . . . 1384
L4Re::Rm::Range
A range of virtual addresses . . . 1385
L4Re::Smart_cap_auto< Unmap_flags >
Helper for Unique_cap and Unique_del_cap . . . 1386
L4Re::Smart_count_cap< Unmap_flags >
Helper for Ref_cap and Ref_del_cap . . . 1387
L4Re::Util::Br_manager
Buffer-register (BR) manager for L4::Server . . . 1388
L4Re::Util::Br_manager_hooks
Predefined server-loop hooks for a server loop using the Br_manager . . . 1394
L4Re::Util::Br_manager_timeout_hooks
Predefined server-loop hooks for a server with using the Br_manager and a timeout queue . . . 1395
L4Re::Util::Cap_alloc_base
Capability allocator . . . 1398
L4Re::Util::Counter< COUNTER >
Counter for Counting_cap_alloc with variable data width . . . 1399
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >
Internal reference-counting cap allocator . . . 1400
L4Re::Util::Dataspace_svr
Dataspace server class . . . 1408
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >
An event buffer consumer . . . 1414
L4Re::Util::Event_buffer_t< PAYLOAD >
Event_buffer utility class . . . 1417
L4Re::Util::Event_svr< SVR >
Convenience wrapper for implementing an event server . . . 1421
L4Re::Util::Event_t< PAYLOAD >
Convenience wrapper for getting access to an event object . . . 1423

L4Re::Util::Item_alloc_base
Item allocator . . . 1427
L4Re::Util::Names::Name
Name class . . . 1428
L4Re::Util::Names::Name_space
Abstract server-side implementation of the L4::Namespace interface . . . 1429
L4Re::Util::Object_registry
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread . . . 1433
L4Re::Util::Ref_cap< T >
Automatic capability that implements automatic free and unmap of the capability selector . . . 1439
L4Re::Util::Ref_del_cap< T >
Automatic capability that implements automatic free and unmap+delete of the capability selector 1440
L4Re::Util::Registry_server< LOOP_HOOKS >
A server loop object which has a Object_registry included . . . 1441
L4Re::Util::Smart_cap_auto< Unmap_flags >
Helper for Unique_cap and Unique_del_cap . . . 1446
L4Re::Util::Smart_count_cap< Unmap_flags >
Helper for Ref_cap and Ref_del_cap . . . 1447
L4Re::Util::Vcon_svr< SVR >
Console server template class . . . 1448
L4Re::Util::Video::Goos_svr
Goos server class . . . 1449
L4Re::Vfs::Be_file
Boiler plate class for implementing an open file for L4Re::Vfs . . . 1452
L4Re::Vfs::Be_file_system
Boilerplate class for implementing a L4Re::Vfs::File_system . . . 1457
L4Re::Vfs::Directory
Interface for a POSIX file that is a directory . . . 1460
L4Re::Vfs::File
The basic interface for an open POSIX file . . . 1465
L4Re::Vfs::File_system
Basic interface for an L4Re::Vfs file system . . . 1467
L4Re::Vfs::Fs
POSIX File-system related functionality . . . 1469
L4Re::Vfs::Generic_file
The common interface for an open POSIX file . . . 1474
L4Re::Vfs::Mman
Interface for the POSIX memory management . . . 1479
L4Re::Vfs::Ops
Interface for the POSIX backends for an application . . . 1480
L4Re::Vfs::Regular_file
Interface for a POSIX file that provides regular file semantics . . . 1483
L4Re::Vfs::Special_file
Interface for a POSIX file that provides special file semantics . . . 1488
L4Re::Video::Color_component
A color component . . . 1491
L4Re::Video::Goos
Class that abstracts framebuffers . . . 1495
L4Re::Video::Goos::Info
Information structure of a Goos . . . 1502
L4Re::Video::Pixel_info
Pixel information . . . 1503
L4Re::Video::View
View of a framebuffer . . . 1512
L4Re::Video::View::Info
Information structure of a view . . . 1517

l4re_aux_t
Auxiliary descriptor . . . 1520
l4re_ds_stats_t
Information about the data space . . . 1521
l4re_elf_aux_mword_t
Auxiliary vector element for a single unsigned data word . . . 1521
l4re_elf_aux_t
Generic header for each auxiliary vector element . . . 1522
l4re_elf_aux_vma_t
Auxiliary vector element for a reserved virtual memory area . . . 1523
l4re_env_cap_entry_t
Entry in the L4Re environment array for the named inital objects . . . 1524
l4re_env_t
Initial environment data structure . . . 1526
l4re_event_t
Event structure used in buffer . . . 1527
l4re_video_color_component_t
Color component structure . . . 1528
l4re_video_goos_info_t
Goos information structure . . . 1529
l4re_video_pixel_info_t
Pixel_info structure . . . 1530
l4re_video_view_info_t
View information structure . . . 1531
l4re_video_view_t
C representation of a goos view . . . 1533
l4util_idt_desc_t
IDT entry . . . 1534
l4util_idt_header_t
Header of an IDT table . . . 1535
l4util_l4mod_info
Base module structure . . . 1536
l4util_l4mod_mod
A single module . . . 1537
l4util_mb_addr_range_t
INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached . . . 1538
l4util_mb_apm_t
APM BIOS info . . . 1539
l4util_mb_drive_t
Drive Info structure . . . 1540
l4util_mb_info_t
MultiBoot Info description . . . 1542
l4util_mb_mod_t
The structure type "mod_list" is used by the multiboot_info structure . . . 1543
l4util_mb_vbe_ctrl_t
VBE controller information . . . 1544
l4util_mb_vbe_mode_t
VBE mode information . . . 1545
L4vbus::Device
Device on a L4vbus::Vbus . . . 1549
L4vbus::Gpio_module
A Gpio_module groups multiple GPIO pins together . . . 1557
L4vbus::Gpio_module::Pin_slice
A slice of the pins provided by this module . . . 1563
L4vbus::Gpio_pin
A GPIO pin . . . 1564

L4vbus::Icu
Vbus Interrupt controller API . . . 1572
L4vbus::Pci_dev
A PCI device . . . 1575
L4vbus::Pci_host_bridge
A Pci host bridge . . . 1580
L4vbus::Pm< DEC >
Power-management API mixin . . . 1586
L4vbus::Vbus
The virtual bus (Vbus) interface . . . 1587
l4vbus_device_t
Detailed information about a vbus device . . . 1596
l4vbus_resource_t
Description of a single vbus resource . . . 1597
L4vcpu::State
C++ implementation of state word in the vCPU area . . . 1598
L4vcpu::Vcpu
C++ implementation of the vCPU save state area . . . 1600
L4virtio::Device
IPC interface for virtio over L4 IPC . . . 1612
L4virtio::Driver::Block_device
Simple class for accessing a virtio block device synchronously . . . 1619
L4virtio::Driver::Block_device::Handle
Handle to an ongoing request . . . 1627
L4virtio::Driver::Device
Client-side implementation for a general virtio device . . . 1628
L4virtio::Driver::Virtqueue
Driver-side implementation of a Virtqueue . . . 1641
L4virtio::Ptr< T >
Pointer used in virtio descriptors . . . 1650
L4virtio::Svr::Bad_descriptor
Exception used by Queue to indicate descriptor errors . . . 1654
L4virtio::Svr::Block_dev_base< Ds_data >
Base class for virtio block devices . . . 1656
L4virtio::Svr::Block_request< Ds_data >
A request to read or write data . . . 1665
L4virtio::Svr::Data_buffer
Abstract data buffer . . . 1668
L4virtio::Svr::Dev_config
Abstraction for L4-Virtio device config memory . . . 1672
L4virtio::Svr::Dev_features
Type for device feature bitmap . . . 1683
L4virtio::Svr::Dev_status
Type of the device status register . . . 1684
L4virtio::Svr::Device_t< DATA >
Server-side L4-VIRTIO device stub . . . 1687
L4virtio::Svr::Driver_mem_list_t< DATA >
List of driver memory regions assigned to a single L4-VIRTIO transport instance . . . 1694
L4virtio::Svr::Driver_mem_region_t< DATA >
Region of driver memory, that shall be managed locally . . . 1704
L4virtio::Svr::Request_processor
Encapsulate the state for processing a VIRTIO request . . . 1711
L4virtio::Svr::Virtqueue
Virtqueue implementation for the device . . . 1718
L4virtio::Svr::Virtqueue::Head_desc
VIRTIO request, essentially a descriptor from the available ring . . . 1724
L4virtio::Virtqueue
Low-level Virtqueue . . . 1726

L4virtio::Virtqueue::Avail
Type of available ring, this is read-only for the host . . . 1742
L4virtio::Virtqueue::Avail::Flags
Flags of the available ring . . . 1744
L4virtio::Virtqueue::Desc
Descriptor in the descriptor table . . . 1745
L4virtio::Virtqueue::Desc::Flags
Type for descriptor flags . . . 1747
L4virtio::Virtqueue::Used
Used ring . . . 1749
L4virtio::Virtqueue::Used::Flags
Flags for the used ring . . . 1750
L4virtio::Virtqueue::Used_elem
Type of an element of the used ring . . . 1752
l4virtio_block_config_t
Device configuration for block devices . . . 1753
l4virtio_block_discard_t
Structure used for the write zeroes and discard commands . . . 1754
l4virtio_block_header_t
Header structure of a request for a block device . . . 1755
l4virtio_config_hdr_t
L4-VIRTIO config header, provided in shared data space . . . 1756
l4virtio_config_queue_t
Queue configuration entry . . . 1757
l4virtio_input_absinfo_t
Information about the absolute axis in the underlying evdev implementation . . . 1759
l4virtio_input_config_t
Device configuration for input devices . . . 1759
l4virtio_input_devids_t
Device ID information for the device . . . 1760
l4virtio_input_event_t
Single event in event or status queue . . . 1761

Chapter 12

File Index

12.1 File List

Here is a list of all documented files with brief descriptions:

pkg/l4re-core/ned/lib/include/**cmd_control** . . . . . ??
pkg/l4re-core/ned/lib/include/**Makefile** . . . . . ??
amd64/l4/sys/**__kip-arch.h** . . . . . ??
amd64/l4/sys/**__vcpu-arch.h** . . . . . ??
amd64/l4/sys/cache.h
Cache functions 2119
amd64/l4/sys/consts.h
Common L4 constants, amd64 version 2134
amd64/l4/sys/**ktrace_events.h** . . . . . ??
amd64/l4/sys/l4int.h
Fixed sized integer types, amd64 version 2227
amd64/l4/sys/linkage.h
Linkage 1828
amd64/l4/sys/segment.h
Segment handling 1809
amd64/l4/sys/utcb.h
UTCB definitions for amd64 2289
amd64/l4/sys/**vm.h** . . . . . ??
amd64/l4/util/apic.h
APIC for AMD64 1763
amd64/l4/util/bitops_arch.h
Amd64 bit manipulation functions 1834
amd64/l4/util/cpu.h
CPU related functions 1842
amd64/l4/util/idt.h
IDT related functions 1774
amd64/l4/util/irq.h
Some PIC and hardware interrupt related functions 1933
amd64/l4/util/l4_macros.h
Main function 1846
amd64/l4/util/mbi_argv.h
Command line handling 1850
amd64/l4/util/perform.h
Perfomance Monitoring using P5/P6 Measurement Counters 1777
amd64/l4/util/port_io.h
Port I/O functions 1819

amd64/l4/util/rdtsc.h
Time stamp counter related functions . . . 1789
amd64/l4/util/spin.h
Spinning for amd64 . . . 1798
amd64/l4/util/util.h
Utilities, amd64 version . . . 1800
amd64/l4f/l4/sys/**ipc.h** . . . **??**
amd64/l4f/l4/sys/segment.h
L4f specific fs/gs manipulation . . . 1806
amd64/l4f/l4/util/port_io.h
Port I/O functions . . . 1818
arm/l4/sys/**__kip-arch.h** . . . **??**
arm/l4/sys/**__vcpu-arch.h** . . . **??**
arm/l4/sys/atomic.h
Atomic memory modifications . . . 2318
arm/l4/sys/cache.h
Cache functions . . . 2116
arm/l4/sys/consts.h
Common L4 constants, arm version . . . 2133
arm/l4/sys/**ktrace_events.h** . . . **??**
arm/l4/sys/l4int.h
Fixed sized integer types, arm version . . . 2227
arm/l4/sys/linkage.h
Linkage . . . 1827
arm/l4/sys/mem_op.h
Memory access functions (ARM specific) . . . 1830
arm/l4/sys/**task.h** . . . **??**
arm/l4/sys/thread.h
ARM-specific thread related definitions . . . 2382
arm/l4/sys/utcb.h
UTCB definitions for ARM . . . 2286
arm/l4/sys/**vm** . . . **??**
arm/l4/sys/vm.h
ARM virtualization interface . . . 1832
arm/l4/util/bitops_arch.h
ARM specific implementation of bitops functions . . . 1833
arm/l4/util/cpu.h
CPU related functions . . . 1841
arm/l4/util/irq.h
ARM specific implementation of irq functions . . . 1931
arm/l4/util/l4_macros.h
Main function . . . 1846
arm/l4/util/mbi_argv.h
Multiboot . . . 1849
arm/l4f/l4/sys/ipc.h
L4 IPC System Calls, ARM . . . 2206
arm/l4f/l4/sys/syscall_defs.h
Syscall entry definitions . . . 1853
contrib/libio-io/l4/io/io.h . . . 1853
contrib/libio-io/l4/io/**types.h** . . . **??**
l4/cxx/alloc.h
Alloc list . . . 1857
l4/cxx/**arith** . . . **??**
l4/cxx/atomic.h
Atomic template . . . 2319
l4/cxx/**auto_ptr** . . . **??**
l4/cxx/avl_map
AVL map . . . 1858

l4/cxx/avl_set
AVL set . . . 1861
l4/cxx/avl_tree
AVL tree . . . 1866
l4/cxx/basic_ostream
Basic IO stream . . . 1872
l4/cxx/basic_vector.h
Basic vector . . . 1876
l4/cxx/**bitfield** . . . ??
l4/cxx/**bitmap** . . . ??
l4/cxx/**dlist** . . . ??
l4/cxx/exceptions
Base exceptions . . . 1890
l4/cxx/**hlist** . . . ??
l4/cxx/iostream
IO Stream . . . 1894
l4/cxx/ipc_helper
IPC helper . . . 1896
l4/cxx/ipc_server
IPC server loop . . . 2153
l4/cxx/ipc_stream
IPC stream . . . 1898
l4/cxx/**ipc_timeout_queue** . . . ??
l4/cxx/l4iostream
L4 IO stream . . . 1916
l4/cxx/l4types.h
L4 Types . . . 1917
l4/cxx/**list** . . . ??
l4/cxx/**list_alloc** . . . ??
l4/cxx/main_thread
Main thread . . . 1919
l4/cxx/**minmax** . . . ??
l4/cxx/**observer** . . . ??
l4/cxx/pair
Pair implementation . . . 1920
l4/cxx/**ref_ptr** . . . ??
l4/cxx/ref_ptr_list
Implementation of a list of ref-ptr-managed objects . . . 1922
l4/cxx/**slab_alloc** . . . ??
l4/cxx/**slist** . . . ??
l4/cxx/**static_container** . . . ??
l4/cxx/**static_vector** . . . ??
l4/cxx/**std_alloc** . . . ??
l4/cxx/std_exc_io
Base exceptions std stream operator . . . 1923
l4/cxx/**std_ops** . . . ??
l4/cxx/**string** . . . ??
l4/cxx/string.h
String . . . 1925
l4/cxx/thread
Thread implementation . . . 2272
l4/cxx/**type_list** . . . ??
l4/cxx/**type_traits** . . . ??
l4/cxx/**unique_ptr** . . . ??
l4/cxx/unique_ptr_list
Implementation of a list of unique-ptr-managed objects . . . 1927
l4/cxx/**utils** . . . ??
l4/cxx/**weak_ref** . . . ??

l4/cxx/bits/bst.h
AVL tree . . . 1877
l4/cxx/bits/bst_base.h
AVL tree . . . 1881
l4/cxx/bits/bst_iter.h
AVL tree . . . 1884
l4/cxx/bits/**list_basics.h** . . . **??**
l4/cxx/bits/smart_ptr_list.h
Implementation of a list of smart-pointer-managed objects . . . 1887
l4/cxx/bits/**type_traits.h** . . . **??**
l4/irq/irq.h
IRQ handling routines . . . 1929
l4/l4re_vfs/**backend** . . . **??**
l4/l4re_vfs/**vfs.h** . . . **??**
l4/l4re_vfs/impl/**default_ops_impl.h** . . . **??**
l4/l4re_vfs/impl/**fd_store.h** . . . **??**
l4/l4re_vfs/impl/**fd_store_impl.h** . . . **??**
l4/l4re_vfs/impl/**ns_fs.h** . . . **??**
l4/l4re_vfs/impl/**ns_fs_impl.h** . . . **??**
l4/l4re_vfs/impl/**ro_file.h** . . . **??**
l4/l4re_vfs/impl/**ro_file_impl.h** . . . **??**
l4/l4re_vfs/impl/**vcon_stream.h** . . . **??**
l4/l4re_vfs/impl/**vcon_stream_impl.h** . . . **??**
l4/l4re_vfs/impl/**vfs_impl.h** . . . **??**
l4/l4virtio/**l4virtio** . . . **??**
l4/l4virtio/**virtio.h** . . . **??**
l4/l4virtio/**virtio_block.h** . . . **??**
l4/l4virtio/**virtio_input.h** . . . **??**
l4/l4virtio/**virtqueue** . . . **??**
l4/l4virtio/client/**virtio-block** . . . **??**
l4/l4virtio/server/**l4virtio** . . . **??**
l4/l4virtio/server/**virtio** . . . **??**
l4/l4virtio/server/**virtio-block** . . . **??**
l4/libedid/edid.h . . . 1942
l4/libgfxbitmap/bitmap.h
Bitmap renderer header file . . . 1943
l4/libgfxbitmap/font.h
Bitmap font renderer header file . . . 1949
l4/libgfxbitmap/support
Terminal support functionality . . . 1955
l4/re/cap_alloc
Abstract capability-allocator interface . . . 1991
l4/re/**console** . . . **??**
l4/re/consts
Constants . . . 2137
l4/re/consts.h
Constants . . . 2135
l4/re/dataspace
Dataspace interface . . . 1997
l4/re/dataspace-sys.h
Dataspace protocol defintion . . . 2000
l4/re/debug
Debug interface . . . 2001
l4/re/dma_space . . . 2003
l4/re/elf_aux.h
Auxiliary information for binaries . . . 2006
l4/re/env
Environment interface . . . 2008

l4/re/env.h
Environment interface . . . 2010
l4/re/error_helper
Error helper . . . 2014
l4/re/**event** . . . **??**
l4/re/**event-sys.h** . . . **??**
l4/re/event.h
Events . . . 1966
l4/re/**event_enums.h** . . . **??**
l4/re/**inhibitor** . . . **??**
l4/re/**inhibitor-sys.h** . . . **??**
l4/re/l4aux.h
Auxiliary definitions . . . 2028
l4/re/log
Log interface . . . 2030
l4/re/log-sys.h
Log protocol definition . . . 2031
l4/re/mem_alloc
Memory allocator interface . . . 2032
l4/re/mem_alloc-sys.h
Memory allocator protocol definitions . . . 2034
l4/re/mmio_space
Interface definition to emit MMIO-like accesses via IPC . . . 2035
l4/re/namespace
Namespace interface . . . 2037
l4/re/namespace-sys.h
Namespace protocol definitions . . . 2040
l4/re/parent
Parent interface . . . 2041
l4/re/parent-sys.h
Parent protocol definition . . . 2043
l4/re/protocols.h
L4Re Protocol Constants (C version) . . . 2044
l4/re/random
Random number generator interface definition . . . 2046
l4/re/rm
Region mapper interface . . . 2047
l4/re/rm-sys.h
Region mapper protocol definitions . . . 2053
l4/re/shared_cap
Shared_cap / Shared_del_cap . . . 2054
l4/re/unique_cap
Unique_cap / Unique_del_cap . . . 2059
l4/re/c/dataspace.h
Data space C interface . . . 1957
l4/re/c/debug.h
Debug C interface . . . 1959
l4/re/c/dma_space.h
DMA space C interface . . . 1961
l4/re/c/event.h
Event C interface . . . 1964
l4/re/c/**event_buffer.h** . . . **??**
l4/re/c/**inhibitor.h** . . . **??**
l4/re/c/log.h
Log C interface . . . 1968
l4/re/c/mem_alloc.h
Memory allocator C interface . . . 1969

l4/re/c/namespace.h
Namespace functions, C interface . . . 1971
l4/re/c/rm.h
Region map interface, C interface . . . 1973
l4/re/c/util/cap_alloc.h
Capability allocator C interface . . . 1977
l4/re/c/util/kumem_alloc.h
Kumem allocator utility C interface . . . 1979
l4/re/c/util/video/goos_fb.h
Framebuffer utility functionality . . . 1981
l4/re/c/video/colors.h . . . 1982
l4/re/c/video/goos.h . . . 1985
l4/re/c/video/view.h . . . 1988
l4/re/impl/dataspace_impl.h
Dataspace client stub implementation . . . 2019
l4/re/impl/mem_alloc_impl.h
Memory allocator client stub implementation . . . 2022
l4/re/impl/namespace_impl.h
Namespace client stub implementation . . . 2023
l4/re/impl/rm_impl.h
Region map client stub implementation . . . 2026
l4/re/util/bitmap_cap_alloc
Bitmap capability allocator . . . 2063
l4/re/util/**br_manager** . . . **??**
l4/re/util/cap
Capability utility functions . . . 2066
l4/re/util/cap_alloc
Capability allocator . . . 1994
l4/re/util/cap_alloc_impl.h
Capability allocator implementation . . . 2067
l4/re/util/counting_cap_alloc
Reference-counting capability allocator . . . 2069
l4/re/util/**dataspace_svr** . . . **??**
l4/re/util/**debug** . . . **??**
l4/re/util/**env_ns** . . . **??**
l4/re/util/event . . . 2017
l4/re/util/**event_buffer** . . . **??**
l4/re/util/**event_svr** . . . **??**
l4/re/util/**icu_svr** . . . **??**
l4/re/util/item_alloc
Item allocator . . . 2073
l4/re/util/kumem_alloc
Kumem allocator helper . . . 2075
l4/re/util/**meta** . . . **??**
l4/re/util/**name_space_svr** . . . **??**
l4/re/util/**object_registry** . . . **??**
l4/re/util/**poll_timeout_kipclock** . . . **??**
l4/re/util/region_mapping
Region handling . . . 2076
l4/re/util/**region_mapping_svr_2** . . . **??**
l4/re/util/shared_cap
Shared_cap / Shared_del_cap . . . 2056
l4/re/util/unique_cap
Unique_cap / Unique_del_cap . . . 2061
l4/re/util/**vcon_svr** . . . **??**
l4/re/util/video/**goos_fb** . . . **??**
l4/re/util/video/**goos_svr** . . . **??**
l4/re/video/**colors** . . . **??**

l4/re/video/**goos** . . . . . ??
l4/re/video/goos-sys.h
Goos protocol definition 2083
l4/re/video/**view** . . . . . ??
l4/shmc/shmc.h
Shared memory library header file 2084
l4/sigma0/sigma0.h
Sigma0 interface 2088
l4/sys/__kernel_object_impl.h
Low-level kernel debugger functions 2092
l4/sys/**__kip-32bit.h** . . . . . ??
l4/sys/**__kip-64bit.h** . . . . . ??
l4/sys/__ktrace-impl.h
L4 kernel event tracing 2093
l4/sys/**__l4_fpage.h** . . . . . ??
l4/sys/**__task-arm.h** . . . . . ??
l4/sys/**__timeout.h** . . . . . ??
l4/sys/__typeinfo.h
Type information handling 2095
l4/sys/**__vcpu-arm.h** . . . . . ??
l4/sys/__vm-arm.h
Virtualization interface 2108
l4/sys/**__vm-svm.h** . . . . . ??
l4/sys/**__vm-vmx.h** . . . . . ??
l4/sys/arm_smccc
ARM secure monitor call functions 2110
l4/sys/arm_smccc.h
ARM secure monitor call functions 2111
l4/sys/assert.h
Low-level assert implementation 2308
l4/sys/cache.h
Cache-consistency functions 2114
l4/sys/capability
L4::Cap related definitions 2122
l4/sys/compiler.h
L4 compiler related defines 2125
l4/sys/consts.h
Common constants 2129
l4/sys/debugger
The debugger interface specifies common debugging related definitions 2172
l4/sys/debugger.h
Debugger related definitions 2174
l4/sys/err.h
Error codes 2181
l4/sys/exception
Exception C++ interface 2182
l4/sys/factory
Common factory related definitions 2184
l4/sys/factory.h
Common factory related definitions 2188
l4/sys/icu
Interrupt controller 2193
l4/sys/icu.h
Interrupt controller 2195
l4/sys/**iommu** . . . . . ??
l4/sys/ipc.h
Common IPC interface 2201

l4/sys/ipc_gate
The C++ IPC gate interface . . . 2209
l4/sys/ipc_gate.h
The C IPC gate interface, see L4::Ipc_gate for the C++ interface . . . 2211
l4/sys/irq
C++ Irq interface . . . 2215
l4/sys/irq.h
C Irq interface . . . 1938
l4/sys/**kdebug.h** . . . **??**
l4/sys/kernel_object.h
Kernel object system calls . . . 2218
l4/sys/kip . . . 2220
l4/sys/kip.h
Kernel Info Page access functions . . . 2350
l4/sys/**kobject** . . . **??**
l4/sys/ktrace.h
L4 kernel event tracing . . . 2223
l4/sys/l4int.h
Fixed sized integer types, generic version . . . 2225
l4/sys/memdesc.h
Memory description functions . . . 2229
l4/sys/meta
Meta interface for getting dynamic type information about objects behind capabilities . . . 2233
l4/sys/pager
Pager and Io_pager C++ interface . . . 2235
l4/sys/platform_control
Platform control object . . . 2237
l4/sys/platform_control.h
Platform control object . . . 2239
l4/sys/rcv_endpoint
The C++ Receive endpoint interface . . . 2243
l4/sys/rcv_endpoint.h
Receive endpoint C interface . . . 2245
l4/sys/scheduler
Scheduler object functions . . . 2248
l4/sys/scheduler.h
Scheduler object functions . . . 2249
l4/sys/semaphore
Semaphore class definition . . . 2254
l4/sys/semaphore.h
C semaphore interface . . . 2256
l4/sys/smart_capability
L4::Capability class . . . 2259
l4/sys/task
Common task related definitions . . . 2262
l4/sys/task.h
Common task related definitions . . . 2265
l4/sys/thread
Common thread related definitions . . . 2269
l4/sys/thread.h
Common thread related definitions . . . 2373
l4/sys/typeinfo_svr
Type information server template . . . 2274
l4/sys/types.h
Common L4 ABI Data Types . . . 2277
l4/sys/utcb.h
UTCB definitions . . . 2282

l4/sys/vcon
C++ Virtual console interface . . . 2294
l4/sys/vcon.h
Virtual console interface . . . 2296
l4/sys/**vcpu.h** . . . **??**
l4/sys/vhw.h
Descriptors for virtual hardware (under UX) . . . 2302
l4/sys/vm
Virtualization interface . . . 2305
l4/sys/cxx/**capability.h** . . . **??**
l4/sys/cxx/**consts** . . . **??**
l4/sys/cxx/**ipc_array** . . . **??**
l4/sys/cxx/**ipc_basics** . . . **??**
l4/sys/cxx/ipc_client . . . 2139
l4/sys/cxx/**ipc_epiface** . . . **??**
l4/sys/cxx/ipc_iface
Interface Definition Language . . . 2142
l4/sys/cxx/**ipc_legacy** . . . **??**
l4/sys/cxx/**ipc_ret_array** . . . **??**
l4/sys/cxx/**ipc_server** . . . **??**
l4/sys/cxx/**ipc_server_loop** . . . **??**
l4/sys/cxx/**ipc_string** . . . **??**
l4/sys/cxx/ipc_types . . . 2156
l4/sys/cxx/**ipc_varg** . . . **??**
l4/sys/cxx/smart_capability_1x . . . 2164
l4/sys/cxx/types . . . 2168
l4/util/assert.h
Some useful assert-style macros . . . 2306
l4/util/atomic.h
Atomic operations header and generic implementations . . . 2311
l4/util/backtrace.h
Backtrace . . . 2320
l4/util/base64.h
Base 64 encoding and decoding functions adapted from Bob Trower 08/04/01 . . . 2322
l4/util/bitops.h
Bit manipulation functions . . . 2324
l4/util/elf.h
ELF definition . . . 2329
l4/util/getopt.h
Getopt . . . 2345
l4/util/keymap.h
Event to ASCII key mapping . . . 2347
l4/util/kip.h . . . 2348
l4/util/kprintf.h
Printf using the kernel debugger . . . 2353
l4/util/l4_macros.h
Some useful generic macros, L4f version . . . 1847
l4/util/**l4mod.h** . . . **??**
l4/util/list_alloc.h
Simple list-based allocator . . . 2354
l4/util/**llulc.h** . . . **??**
l4/util/lock.h
Simple lock implementation . . . 2357
l4/util/mb_info.h
Multiboot info structure as defined by GRUB . . . 2358
l4/util/parse_cmd.h
Comfortable command-line parsing . . . 2366

l4/util/rand.h
Simple Pseudo-Random Number Generator . . . 2367
l4/util/splitlog2.h
Split a range in log2 aligned and size-aligned chunks . . . 2369
l4/util/thread.h
Low-level Thread Functions . . . 2371
l4/util/**util.h** . . . **??**
l4/vbus/**vbus** . . . **??**
l4/vbus/vbus.h
Description of the vbus C API . . . 2383
l4/vbus/**vbus_generic** . . . **??**
l4/vbus/**vbus_gpio** . . . **??**
l4/vbus/**vbus_gpio-ops.h** . . . **??**
l4/vbus/**vbus_gpio.h** . . . **??**
l4/vbus/**vbus_i2c.h** . . . **??**
l4/vbus/**vbus_inhibitor.h** . . . **??**
l4/vbus/vbus_interfaces.h
This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs . . . 2387
l4/vbus/**vbus_mcspi.h** . . . **??**
l4/vbus/**vbus_pci** . . . **??**
l4/vbus/**vbus_pci-ops.h** . . . **??**
l4/vbus/**vbus_pci.h** . . . **??**
l4/vbus/**vbus_pm-ops.h** . . . **??**
l4/vbus/**vbus_pm.h** . . . **??**
l4/vbus/vbus_types.h
This header file contains descriptions of vbus related data types and constants . . . 2390
l4/vbus/**vdevice-ops.h** . . . **??**
l4/vcpu/**vcpu** . . . **??**
l4/vcpu/**vcpu.h** . . . **??**
x86/l4/sys/**__kip-arch.h** . . . **??**
x86/l4/sys/**__vcpu-arch.h** . . . **??**
x86/l4/sys/cache.h
Cache functions . . . 2120
x86/l4/sys/consts.h
Common L4 constants, x86 version . . . 2135
x86/l4/sys/**ipc-invoke.h** . . . **??**
x86/l4/sys/**ktrace_events.h** . . . **??**
x86/l4/sys/l4int.h
Fixed sized integer types, x86 version . . . 2228
x86/l4/sys/linkage.h
Linkage . . . 1829
x86/l4/sys/segment.h
Segment handling . . . 1815
x86/l4/sys/utcb.h
UTCB definitions for X86 . . . 2292
x86/l4/sys/**vm.h** . . . **??**
x86/l4/util/apic.h
APIC for X86 . . . 1768
x86/l4/util/bitops_arch.h
X86 bit manipulation functions . . . 1837
x86/l4/util/cpu.h
CPU related functions . . . 1843
x86/l4/util/idt.h
IDT related functions . . . 1776
x86/l4/util/irq.h
Some PIC and hardware interrupt related functions . . . 1935

x86/l4/util/l4_macros.h
Main function 1848
x86/l4/util/mbi_argv.h
Command line handling 1851
x86/l4/util/perform.h
Perfomance Monitoring using P5/P6 Measurement Counters 1783
x86/l4/util/port_io.h
X86 port I/O 1823
x86/l4/util/rdtsc.h
Time stamp counter related functions 1793
x86/l4/util/spin.h
Spinning for x86 1799
x86/l4/util/util.h
Utilities for x86 1803
x86/l4f/l4/sys/**ipc-l42-gcc3-nopic.h** . . . . . **??**
x86/l4f/l4/sys/ipc.h
L4 IPC System Calls, x86 2208
x86/l4f/l4/sys/segment.h
L4f specific segment manipulation 1814
x86/l4f/l4/util/port_io.h
Port I/O functions 1820

Chapter 13

Module Documentation

13.1 ARM Virtual Registers (UTCB)

Collaboration diagram for ARM Virtual Registers (UTCB):

Data Structures

- struct l4_exc_regs_t

  UTCB structure for exceptions.

Typedefs

- typedef struct l4_exc_regs_t l4_exc_regs_t

  UTCB structure for exceptions.

Enumerations

- enum L4_utcb_consts_arm

  UTCB constants for ARM.

13.1.1 Detailed Description

13.2 Atomic Instructions

Collaboration diagram for Atomic Instructions:

Files

- file atomic.h

  atomic operations header and generic implementations

Functions

- int l4util_cmpxchg64 (volatile l4_uint64_t *dest, l4_uint64_t cmp_val, l4_uint64_t new_val)

  Atomic compare and exchange (64 bit version)
- int l4util_cmpxchg32 (volatile l4_uint32_t *dest, l4_uint32_t cmp_val, l4_uint32_t new_val)

  Atomic compare and exchange (32 bit version)
- int l4util_cmpxchg16 (volatile l4_uint16_t *dest, l4_uint16_t cmp_val, l4_uint16_t new_val)

  Atomic compare and exchange (16 bit version)
- int l4util_cmpxchg8 (volatile l4_uint8_t *dest, l4_uint8_t cmp_val, l4_uint8_t new_val)

  Atomic compare and exchange (8 bit version)
- int l4util_cmpxchg (volatile l4_umword_t *dest, l4_umword_t cmp_val, l4_umword_t new_val)

  Atomic compare and exchange (machine wide fields)
- l4_uint32_t l4util_xchg32 (volatile l4_uint32_t *dest, l4_uint32_t val)

  Atomic exchange (32 bit version)
- l4_uint16_t l4util_xchg16 (volatile l4_uint16_t *dest, l4_uint16_t val)

  Atomic exchange (16 bit version)
- l4_uint8_t l4util_xchg8 (volatile l4_uint8_t *dest, l4_uint8_t val)

  Atomic exchange (8 bit version)
- l4_umword_t l4util_xchg (volatile l4_umword_t *dest, l4_umword_t val)

  Atomic exchange (machine wide fields)
- void l4util_atomic_add (volatile long *dest, long val)

  Atomic add.
- void l4util_atomic_inc (volatile long *dest)

  Atomic increment.

Atomic add/sub/and/or (8,16,32 bit version) without result

- void l4util_add8 (volatile l4_uint8_t *dest, l4_uint8_t val)
- void l4util_add16 (volatile l4_uint16_t *dest, l4_uint16_t val)
- void l4util_add32 (volatile l4_uint32_t *dest, l4_uint32_t val)
- void l4util_sub8 (volatile l4_uint8_t *dest, l4_uint8_t val)
- void l4util_sub16 (volatile l4_uint16_t *dest, l4_uint16_t val)
- void l4util_sub32 (volatile l4_uint32_t *dest, l4_uint32_t val)
- void l4util_and8 (volatile l4_uint8_t *dest, l4_uint8_t val)
- void l4util_and16 (volatile l4_uint16_t *dest, l4_uint16_t val)
- void l4util_and32 (volatile l4_uint32_t *dest, l4_uint32_t val)
- void l4util_or8 (volatile l4_uint8_t *dest, l4_uint8_t val)
- void l4util_or16 (volatile l4_uint16_t *dest, l4_uint16_t val)
- void l4util_or32 (volatile l4_uint32_t *dest, l4_uint32_t val)

Atomic add/sub/and/or operations (8,16,32 bit) with result

- l4_uint8_t l4util_add8_res (volatile l4_uint8_t *dest, l4_uint8_t val)
- l4_uint16_t l4util_add16_res (volatile l4_uint16_t *dest, l4_uint16_t val)
- l4_uint32_t l4util_add32_res (volatile l4_uint32_t *dest, l4_uint32_t val)
- l4_uint8_t l4util_sub8_res (volatile l4_uint8_t *dest, l4_uint8_t val)
- l4_uint16_t l4util_sub16_res (volatile l4_uint16_t *dest, l4_uint16_t val)
- l4_uint32_t l4util_sub32_res (volatile l4_uint32_t *dest, l4_uint32_t val)
- l4_uint8_t l4util_and8_res (volatile l4_uint8_t *dest, l4_uint8_t val)
- l4_uint16_t l4util_and16_res (volatile l4_uint16_t *dest, l4_uint16_t val)
- l4_uint32_t l4util_and32_res (volatile l4_uint32_t *dest, l4_uint32_t val)
- l4_uint8_t l4util_or8_res (volatile l4_uint8_t *dest, l4_uint8_t val)
- l4_uint16_t l4util_or16_res (volatile l4_uint16_t *dest, l4_uint16_t val)
- l4_uint32_t l4util_or32_res (volatile l4_uint32_t *dest, l4_uint32_t val)

Atomic inc/dec (8,16,32 bit) without result

- void l4util_inc8 (volatile l4_uint8_t *dest)
- void l4util_inc16 (volatile l4_uint16_t *dest)
- void l4util_inc32 (volatile l4_uint32_t *dest)
- void l4util_dec8 (volatile l4_uint8_t *dest)
- void l4util_dec16 (volatile l4_uint16_t *dest)
- void l4util_dec32 (volatile l4_uint32_t *dest)

Atomic inc/dec (8,16,32 bit) with result

- l4_uint8_t l4util_inc8_res (volatile l4_uint8_t *dest)
- l4_uint16_t l4util_inc16_res (volatile l4_uint16_t *dest)
- l4_uint32_t l4util_inc32_res (volatile l4_uint32_t *dest)
- l4_uint8_t l4util_dec8_res (volatile l4_uint8_t *dest)
- l4_uint16_t l4util_dec16_res (volatile l4_uint16_t *dest)
- l4_uint32_t l4util_dec32_res (volatile l4_uint32_t *dest)

13.2.1 Detailed Description

13.2.2 Function Documentation

13.2.2.1 l4util_add16()

```
void l4util_add16 (
            volatile l4_uint16_t * dest,
            l4_uint16_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Definition at line 460 of file atomic.h.

13.2.2.2 l4util_add16_res()

```
l4_uint16_t l4util_add16_res (
            volatile l4_uint16_t * dest,
            l4_uint16_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Returns

res

Definition at line 512 of file atomic.h.

13.2.2.3 l4util_add32()

```
void l4util_add32 (
            volatile l4_uint32_t * dest,
            l4_uint32_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Definition at line 464 of file atomic.h.

13.2.2.4 l4util_add32_res()

```
l4_uint32_t l4util_add32_res (
            volatile l4_uint32_t * dest,
            l4_uint32_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Returns

res

Definition at line 516 of file atomic.h.

13.2.2.5 l4util_add8()

```
void l4util_add8 (
            volatile l4_uint8_t * dest,
            l4_uint8_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Definition at line 456 of file atomic.h.

13.2.2.6 l4util_add8_res()

```
l4_uint8_t l4util_add8_res (
            volatile l4_uint8_t * dest,
            l4_uint8_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Returns

res

Definition at line 508 of file atomic.h.

13.2.2.7 l4util_and16()

```
void l4util_and16 (
            volatile l4_uint16_t * dest,
            l4_uint16_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Definition at line 488 of file atomic.h.

13.2.2.8 l4util_and16_res()

```
l4_uint16_t l4util_and16_res (
            volatile l4_uint16_t * dest,
            l4_uint16_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Returns

res

Definition at line 536 of file atomic.h.

13.2.2.9 l4util_and32()

```
void l4util_and32 (
            volatile l4_uint32_t * dest,
            l4_uint32_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Definition at line 492 of file atomic.h.

13.2.2.10 l4util_and32_res()

```
l4_uint32_t l4util_and32_res (
            volatile l4_uint32_t * dest,
            l4_uint32_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Returns

res

Definition at line 540 of file atomic.h.

13.2.2.11 l4util_and8()

```
void l4util_and8 (
            volatile l4_uint8_t * dest,
            l4_uint8_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Definition at line 484 of file atomic.h.

13.2.2.12 l4util_and8_res()

```
l4_uint8_t l4util_and8_res (
            volatile l4_uint8_t * dest,
            l4_uint8_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Returns

res

Definition at line 532 of file atomic.h.

13.2.2.13 l4util_atomic_add()

```
void l4util_atomic_add (
            volatile long * dest,
            long val )  [inline]
```

Atomic add.

Parameters

dest	destination operand
val	value to add

Definition at line 468 of file atomic.h.

13.2.2.14 l4util_atomic_inc()

```
void l4util_atomic_inc (
            volatile long * dest )  [inline]
```

Atomic increment.

Parameters

dest	destination operand

Definition at line 411 of file atomic.h.

13.2.2.15 l4util_cmpxchg()

```
int l4util_cmpxchg (
            volatile l4_umword_t * dest,
            l4_umword_t cmp_val,
            l4_umword_t new_val )  [inline]
```

Atomic compare and exchange (machine wide fields)

Parameters

dest	destination operand
cmp_val	compare value
new_val	new value for dest

Returns

0 if comparison failed, 1 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 367 of file atomic.h.

Referenced by l4util_btr(), l4util_bts(), l4util_clear_bit(), and l4util_set_bit().

Here is the caller graph for this function:

13.2.2.16 l4util_cmpxchg16()

```
int l4util_cmpxchg16 (
            volatile l4_uint16_t * dest,
            l4_uint16_t cmp_val,
            l4_uint16_t new_val )  [inline]
```

Atomic compare and exchange (16 bit version)

Parameters

dest	destination operand
cmp_val	compare value
new_val	new value for dest

Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 351 of file atomic.h.

13.2.2.17 l4util_cmpxchg32()

```
int l4util_cmpxchg32 (
            volatile l4_uint32_t * dest,
            l4_uint32_t cmp_val,
            l4_uint32_t new_val )  [inline]
```

Atomic compare and exchange (32 bit version)

Parameters

dest	destination operand
cmp_val	compare value
new_val	new value for dest

Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 343 of file atomic.h.

13.2.2.18 l4util_cmpxchg64()

```
int l4util_cmpxchg64 (
            volatile l4_uint64_t * dest,
            l4_uint64_t cmp_val,
            l4_uint64_t new_val )  [inline]
```

Atomic compare and exchange (64 bit version)

Parameters

dest	destination operand
cmp_val	compare value
new_val	new value for dest

Returns

0 if comparison failed, 1 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 335 of file atomic.h.

13.2.2.19 l4util_cmpxchg8()

```
int l4util_cmpxchg8 (
            volatile l4_uint8_t * dest,
            l4_uint8_t cmp_val,
            l4_uint8_t new_val )  [inline]
```

Atomic compare and exchange (8 bit version)

Parameters

dest	destination operand
cmp_val	compare value
new_val	new value for dest

Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 359 of file atomic.h.

13.2.2.20 l4util_dec16()

```
void l4util_dec16 (
            volatile l4_uint16_t * dest )  [inline]
```

Parameters

dest	destination operand

Definition at line 419 of file atomic.h.

13.2.2.21 l4util_dec16_res()

```
l4_uint16_t l4util_dec16_res (
            volatile l4_uint16_t * dest )  [inline]
```

Parameters

dest	destination operand

Returns

res

Definition at line 444 of file atomic.h.

13.2.2.22 l4util_dec32()

```
void l4util_dec32 (
            volatile l4_uint32_t * dest )  [inline]
```

Parameters

dest	destination operand

Definition at line 423 of file atomic.h.

13.2.2.23 l4util_dec32_res()

```
l4_uint32_t l4util_dec32_res (
            volatile l4_uint32_t * dest )  [inline]
```

Parameters

dest	destination operand

Returns

res

Definition at line 448 of file atomic.h.

13.2.2.24 l4util_dec8()

```
void l4util_dec8 (
            volatile l4_uint8_t * dest )  [inline]
```

Parameters

dest	destination operand

Definition at line 415 of file atomic.h.

13.2.2.25 l4util_dec8_res()

```
l4_uint8_t l4util_dec8_res (
            volatile l4_uint8_t * dest )  [inline]
```

Parameters

dest	destination operand

Returns

res

Definition at line 440 of file atomic.h.

13.2.2.26 l4util_inc16()

```
void l4util_inc16 (
            volatile l4_uint16_t * dest )  [inline]
```

Parameters

dest	destination operand

Definition at line 403 of file atomic.h.

13.2.2.27 l4util_inc16_res()

```
l4_uint16_t l4util_inc16_res (
```

```
            volatile l4_uint16_t * dest )  [inline]
```

Parameters

dest	destination operand

Returns

res

Definition at line 432 of file atomic.h.

13.2.2.28 l4util_inc32()

```
void l4util_inc32 (
            volatile l4_uint32_t * dest )  [inline]
```

Parameters

dest	destination operand

Definition at line 407 of file atomic.h.

13.2.2.29 l4util_inc32_res()

```
l4_uint32_t l4util_inc32_res (
            volatile l4_uint32_t * dest )  [inline]
```

Parameters

dest	destination operand

Returns

res

Definition at line 436 of file atomic.h.

13.2.2.30 l4util_inc8()

```
void l4util_inc8 (
            volatile l4_uint8_t * dest )  [inline]
```

Parameters

dest	destination operand

Definition at line 399 of file atomic.h.

13.2.2.31 l4util_inc8_res()

```
l4_uint8_t l4util_inc8_res (
            volatile l4_uint8_t * dest )  [inline]
```

Parameters

dest	destination operand

Returns

res

Definition at line 428 of file atomic.h.

13.2.2.32 l4util_or16()

```
void l4util_or16 (
            volatile l4_uint16_t * dest,
            l4_uint16_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Definition at line 500 of file atomic.h.

13.2.2.33 l4util_or16_res()

```
l4_uint16_t l4util_or16_res (
            volatile l4_uint16_t * dest,
            l4_uint16_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Returns

res

Definition at line 548 of file atomic.h.

13.2.2.34 l4util_or32()

```
void l4util_or32 (
            volatile l4_uint32_t * dest,
            l4_uint32_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Definition at line 504 of file atomic.h.

13.2.2.35 l4util_or32_res()

```
l4_uint32_t l4util_or32_res (
            volatile l4_uint32_t * dest,
            l4_uint32_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Returns

res

Definition at line 552 of file atomic.h.

13.2.2.36 l4util_or8()

```
void l4util_or8 (
            volatile l4_uint8_t * dest,
            l4_uint8_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Definition at line 496 of file atomic.h.

13.2.2.37 l4util_or8_res()

```
l4_uint8_t l4util_or8_res (
            volatile l4_uint8_t * dest,
            l4_uint8_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Returns

res

Definition at line 544 of file atomic.h.

13.2.2.38 l4util_sub16()

```
void l4util_sub16 (
            volatile l4_uint16_t * dest,
            l4_uint16_t val )  [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Definition at line 476 of file atomic.h.

13.2.2.39 l4util_sub16_res()

```
l4_uint16_t l4util_sub16_res (
            volatile l4_uint16_t * dest,
            l4_uint16_t val ) [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Returns

res

Definition at line 524 of file atomic.h.

13.2.2.40 l4util_sub32()

```
void l4util_sub32 (
            volatile l4_uint32_t * dest,
            l4_uint32_t val ) [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Definition at line 480 of file atomic.h.

13.2.2.41 l4util_sub32_res()

```
l4_uint32_t l4util_sub32_res (
            volatile l4_uint32_t * dest,
            l4_uint32_t val ) [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Returns

res

Definition at line 528 of file atomic.h.

13.2.2.42 l4util_sub8()

```
void l4util_sub8 (
            volatile l4_uint8_t * dest,
            l4_uint8_t val ) [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Definition at line 472 of file atomic.h.

13.2.2.43 l4util_sub8_res()

```
l4_uint8_t l4util_sub8_res (
            volatile l4_uint8_t * dest,
            l4_uint8_t val ) [inline]
```

Parameters

dest	destination operand
val	value to add/sub/and/or

Returns

res

Definition at line 520 of file atomic.h.

13.2.2.44 l4util_xchg()

```
l4_umword_t l4util_xchg (
            volatile l4_umword_t * dest,
            l4_umword_t val ) [inline]
```

Atomic exchange (machine wide fields)

Parameters

dest	destination operand
val	new value for dest

Returns

old value at destination

Definition at line 393 of file atomic.h.

13.2.2.45 l4util_xchg16()

```
l4_uint16_t l4util_xchg16 (
            volatile l4_uint16_t * dest,
            l4_uint16_t val ) [inline]
```

Atomic exchange (16 bit version)

Parameters

dest	destination operand
val	new value for dest

Returns

old value at destination

Definition at line 381 of file atomic.h.

13.2.2.46 l4util_xchg32()

```
l4_uint32_t l4util_xchg32 (
            volatile l4_uint32_t * dest,
            l4_uint32_t val ) [inline]
```

Atomic exchange (32 bit version)

Parameters

dest	destination operand
val	new value for dest

Returns

old value at destination

Definition at line 375 of file atomic.h.

13.2.2.47 l4util_xchg8()

```
l4_uint8_t l4util_xchg8 (
            volatile l4_uint8_t * dest,
            l4_uint8_t val )  [inline]
```

Atomic exchange (8 bit version)

Parameters

dest	destination operand
val	new value for dest

Returns

old value at destination

Definition at line 387 of file atomic.h.

13.3 Auxiliary data

Collaboration diagram for Auxiliary data:

Data Structures

- struct l4re_aux_t

  Auxiliary descriptor.

Typedefs

- typedef struct l4re_aux_t l4re_aux_t

  Auxiliary descriptor.

Enumerations

- enum l4re_aux_ldr_flags_t

  Flags for program loading.

13.3.1 Detailed Description

13.4 Base API

Interfaces for all kinds of base functionality.

Collaboration diagram for Base API:

Modules

- Basic Macros

  L4 standard macros for header files, function definitions, and public APIs etc.
- C++ IPC Interface Definition.

  APIs for defining IPC interfaces using C++ as language.
- Cache Consistency

  Various functions for cache consistency.
- Capabilities

  C interface for capabilities.
- Error codes

  Common error codes.
- Fiasco extensions

  Kernel debugger extensions of the Fiasco L4 implementation.
- Flex pages

  Flex-page related API.
- Integer Types
- Kernel Interface Page

  Kernel Interface Page.
- Kernel Objects

  API of kernel objects.
- Memory operations.

  Operations for memory access.
- Memory related

  Memory related constants, data types and functions.
- Object Invocation

  API for L4 object invocation.

Files

- file cache.h

  Cache-consistency functions.
- file compiler.h

  L4 compiler related defines.
- file consts.h

  Common constants.
- file debugger.h

  Debugger related definitions.
- file factory.h

  Common factory related definitions.
- file icu

  Interrupt controller.
- file icu.h

  Interrupt controller.
- file ipc.h

  Common IPC interface.
- file irq.h

  C Irq interface.
- file kip
- file kip.h

Kernel Info Page access functions.

- file memdesc.h

  Memory description functions.
- file semaphore.h

  C semaphore interface.
- file types.h

  Common L4 ABI Data Types.
- file vhw.h

  Descriptors for virtual hardware (under UX).
- file consts.h

  Common L4 constants, arm version.
- file consts.h

  Common L4 constants, amd64 version.
- file ipc.h

  L4 IPC System Calls, x86.
- file consts.h

  Common L4 constants, x86 version.

13.4.1 Detailed Description

Interfaces for all kinds of base functionality.

Some notes on Inter Process Communication (IPC)

IPC in L4 is always synchronous and unbuffered: a message is transferred from the sender to the recipient if and only if the recipient has invoked a corresponding IPC operation. The sender blocks until this happens or a timeout specified by the sender elapsed without the destination becoming ready to receive.

13.5 Basic Macros

L4 standard macros for header files, function definitions, and public APIs etc.

Collaboration diagram for Basic Macros:

Macros

- #define L4_ALWAYS_INLINE

  L4 Inline function attribute.
- #define __END_DECLS

  End section with C types and functions.
- #define EXTERN_C_BEGIN

  Start section with C types and functions.
- #define EXTERN_C_END

  End section with C types and functions.
- #define EXTERN_C

  Mark C types and functions.
- #define L4_NOTHROW

  Mark a function declaration and definition as never throwing an exception.
- #define L4_HIDDEN

  Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.
- #define L4_NORETURN

  Noreturn function attribute.
- #define L4_NOINSTRUMENT

  No instrumentation function attribute.
- #define L4_LIKELY(x)

  Expression is likely to execute.
- #define L4_UNLIKELY(x)

  Expression is unlikely to execute.
- #define L4_STICKY(x)

  Mark symbol sticky (even not there)
- #define L4_DEPRECATED(s)

  Mark symbol deprecated.
- #define L4_stringify_helper(x)

  stringify helper.
- #define L4_stringify(x)

  stringify.
- #define L4_CV

  Define calling convention.
- #define L4_CV

  Define calling convention.
- #define L4_CV __attribute__((regparm(0)))

  Define calling convention.

Functions

- unsigned long l4_align_stack_for_direct_fncall (unsigned long stack)

  Specify the desired alignment of the stack pointer.
- void l4_barrier (void)

  Memory barrier.
- void l4_mb (void)

  Memory barrier.
- void l4_wmb (void)

  Write memory barrier.

13.5.1 Detailed Description

L4 standard macros for header files, function definitions, and public APIs etc.

Include File

```
#include <l4/sys/compiler.h>
```

13.5.2 Macro Definition Documentation

13.5.2.1 L4_ALWAYS_INLINE

```
#define L4_ALWAYS_INLINE
```

L4 Inline function attribute.

Always inline a function

Definition at line 67 of file compiler.h.

13.5.2.2 L4_HIDDEN

```
#define L4_HIDDEN
```

Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.

This attribute is intended for functions, data, and data types that shall never be visible outside of a library. In particular, for shared libraries this may result in much faster code within the library and short linking times.

```
class L4_HIDDEN My_class
{
  ...
};
int L4_HIDDEN function(void);
int L4_HIDDEN global_data; // global data is not recommended
```

Definition at line 216 of file compiler.h.

13.5.2.3 L4_NOTHROW

```
#define L4_NOTHROW
```

Mark a function declaration and definition as never throwing an exception.

(Also for C code).

This macro shall be used to mark C and C++ functions that never throw any exception. Note that also C functions may throw exceptions according to the compilers ABI and shall be marked with L4_NOTHROW if they never do. In C++ this is equivalent to `throw()`.

```
int foo() L4_NOTHROW;
...
int foo() L4_NOTHROW
{
  ...
  return result;
}
```

Definition at line 186 of file compiler.h.

13.5.3 Function Documentation

13.5.3.1 l4_align_stack_for_direct_fncall()

```
unsigned long l4_align_stack_for_direct_fncall (
            unsigned long stack )  [inline]
```

Specify the desired alignment of the stack pointer.

BIGGEST_ALIGNMENT provides the largest alignment ever used for any data type on the target machine. This is normally identical to desired stack alignment. Align stack pointer for directly invoked functions.

The stack needs to be aligned to L4_STACK_ALIGN for being able to access certain data on the stack. On x86/↩ AMD64, a function call is performed using the 'call' instruction decrementing the stack pointer and writing the return address onto the stack. The called function considers this when adapting the stack pointer after function entry. If the called function was not invoked by a 'call' instruction, the stack pointer is actually off by a machine word leading to stack alignment issues when executing SSE instructions.

This function fixes the stack pointer for directly invoked functions. For architectures not automatically pushing the stack pointer during a function call, just enforce the L4_STACK_ALIGN alignment.

Definition at line 272 of file compiler.h.

13.6 Bit Manipulation

Collaboration diagram for Bit Manipulation:

Files

- file bitops.h

  bit manipulation functions

Functions

- void l4util_set_bit (int b, volatile l4_umword_t *dest)

  Set bit in memory.
- void l4util_clear_bit (int b, volatile l4_umword_t *dest)

  Clear bit in memory.
- void l4util_complement_bit (int b, volatile l4_umword_t *dest)

  Complement bit in memory.
- int l4util_test_bit (int b, const volatile l4_umword_t *dest)

  Test bit (return value of bit)
- int l4util_bts (int b, volatile l4_umword_t *dest)

  Bit test and set.
- int l4util_btr (int b, volatile l4_umword_t *dest)

  Bit test and reset.
- int l4util_btc (int b, volatile l4_umword_t *dest)

  Bit test and complement.
- int l4util_bsr (l4_umword_t word)

  Bit scan reverse.
- int l4util_bsf (l4_umword_t word)

  Bit scan forward.
- int l4util_find_first_set_bit (const void *dest, l4_size_t size)

  Find the first set bit in a memory region.
- int l4util_find_first_zero_bit (const void *dest, l4_size_t size)

  Find the first zero bit in a memory region.
- int l4util_next_power2 (unsigned long val)

  Find the next power of 2 for a given number.

13.6.1 Detailed Description

13.6.2 Function Documentation

13.6.2.1 l4util_bsf()

```
int l4util_bsf (
            l4_umword_t word ) [inline]
```

Bit scan forward.

Parameters

word	value (machine size)

Returns

index of least significant bit set in word, -1 if no bit is set (word == 0)

"bit scan forward", find least significant bit set in word.

Definition at line 316 of file bitops.h.

Referenced by l4util_splitlog2_size().

Here is the caller graph for this function:

13.6.2.2 l4util_bsr()

```
int l4util_bsr (
            l4_umword_t word ) [inline]
```

Bit scan reverse.

Parameters

word	value (machine size)

Returns

index of most significant set bit in word, -1 if no bit is set (word == 0)

"bit scan reverse", find most significant set bit in word (-> LOG2(word))

Definition at line 299 of file bitops.h.

Referenced by l4util_splitlog2_size(), and L4virtio::Driver::Block_device::setup_device().

Here is the caller graph for this function:

13.6.2.3 l4util_btc()

```
int l4util_btc (
            int b,
            volatile l4_umword_t * dest )  [inline]
```

Bit test and complement.

Parameters

b	bit position
dest	destination operand

Returns

Old value of bit *b*.

Complement bit *b* and return old value.

Definition at line 394 of file bitops.h.

13.6.2.4 l4util_btr()

```
int l4util_btr (
            int b,
            volatile l4_umword_t * dest )  [inline]
```

Bit test and reset.

Parameters

b	bit position
dest	destination operand

Returns

Old value of bit *b*.

Reset bit *b* and return old value.

Definition at line 278 of file bitops.h.

References l4util_cmpxchg().

Here is the call graph for this function:

13.6.2.5 l4util_bts()

```
int l4util_bts (
            int b,
            volatile l4_umword_t * dest )  [inline]
```

Bit test and set.

Parameters

b	bit position
dest	destination operand

Returns

Old value of bit *b*.

Set the *b* bit of *dest* to 1 and return the old value.

Definition at line 256 of file bitops.h.

References l4util_cmpxchg().

Here is the call graph for this function:

13.6.2.6 l4util_clear_bit()

```
void l4util_clear_bit (
            int b,
            volatile l4_umword_t * dest )  [inline]
```

Clear bit in memory.

Parameters

b	bit position
dest	destination operand

Definition at line 226 of file bitops.h.

References l4util_cmpxchg().

Here is the call graph for this function:

13.6.2.7 l4util_complement_bit()

```
void l4util_complement_bit (
            int b,
            volatile l4_umword_t * dest )  [inline]
```

Complement bit in memory.

Parameters

b	bit position
dest	destination operand

Definition at line 359 of file bitops.h.

13.6.2.8 l4util_find_first_set_bit()

```
int l4util_find_first_set_bit (
```

```
            const void * dest,
            l4_size_t size )  [inline]
```

Find the first set bit in a memory region.

Parameters

dest	bit string
size	size of string in bits (must be a multiple of 32!)

Returns

number of the first set bit, >= size if no bit is set

Definition at line 400 of file bitops.h.

13.6.2.9 l4util_find_first_zero_bit()

```
int l4util_find_first_zero_bit (
            const void * dest,
            l4_size_t size )  [inline]
```

Find the first zero bit in a memory region.

Parameters

dest	bit string
size	size of string in bits (must be a multiple of 32!)

Returns

number of the first zero bit, >= size if no bit is set

Definition at line 333 of file bitops.h.

13.6.2.10 l4util_next_power2()

```
int l4util_next_power2 (
            unsigned long val )  [inline]
```

Find the next power of 2 for a given number.

Parameters

val	initial value

Returns

next-highest power of 2

Definition at line 373 of file bitops.h.

13.6.2.11 l4util_set_bit()

```
void l4util_set_bit (
            int b,
            volatile l4_umword_t * dest )  [inline]
```

Set bit in memory.

Parameters

b	bit position
dest	destination operand

Definition at line 207 of file bitops.h.

References l4util_cmpxchg().

Here is the call graph for this function:

13.6.2.12 l4util_test_bit()

```
int l4util_test_bit (
            int b,
            const volatile l4_umword_t * dest )  [inline]
```

Test bit (return value of bit)

Parameters

b	bit position
dest	destination operand

Returns

Value of bit *b*.

Definition at line 244 of file bitops.h.

13.7 Bitmap graphics and fonts

This library provides some functions for bitmap handling in frame buffers.

Collaboration diagram for Bitmap graphics and fonts:

Modules

- Functions for rendering bitmap data in frame buffers
- Functions for rendering bitmap fonts to frame buffers

13.7.1 Detailed Description

This library provides some functions for bitmap handling in frame buffers.

Includes simple functions like filling or copying an area of the frame buffer going up to rendering text into the frame buffer using bitmap fonts.

13.8 Buffer Registers (BRs)

Collaboration diagram for Buffer Registers (BRs):

Data Structures

- struct l4_buf_regs_t

  Encapsulation of the buffer-registers block in the UTCB.

Typedefs

- typedef struct l4_buf_regs_t l4_buf_regs_t

  Encapsulation of the buffer-registers block in the UTCB.

Enumerations

- enum l4_buffer_desc_consts_t { L4_BDR_MEM_SHIFT = 0 , L4_BDR_IO_SHIFT = 5 , L4_BDR_OBJ_SHIFT = 10 , **L4_BDR_OFFSET_MASK** = (1UL << 20) - 1 }

  Constants for buffer descriptors.

Functions

- void l4_utcb_inherit_fpu (int switch_on) L4_NOTHROW

  Enable or disable inheritance of FPU state to receiver.

13.8.1 Detailed Description

13.8.2 Enumeration Type Documentation

13.8.2.1 l4_buffer_desc_consts_t

`enum l4_buffer_desc_consts_t`

Constants for buffer descriptors.

Enumerator

L4_BDR_MEM_SHIFT	Bit offset for the memory-buffer index.
L4_BDR_IO_SHIFT	Bit offset for the IO-buffer index.
L4_BDR_OBJ_SHIFT	Bit offset for the capability-buffer index.

Definition at line 225 of file consts.h.

13.9 C++ Exceptions

Collaboration diagram for C++ Exceptions:

Files

- file exceptions

  Base exceptions.
- file std_exc_io

  Base exceptions std stream operator.

Data Structures

- class L4::Exception_tracer

  Back-trace support for exceptions.
- class L4::Base_exception

  Base class for all exceptions, thrown by the L4Re framework.
- class L4::Runtime_error

  Exception for an abstract runtime error.
- class L4::Out_of_memory

  Exception signalling insufficient memory.
- class L4::Element_already_exists

  Exception for duplicate element insertions.
- class L4::Unknown_error

  Exception for an unknown condition.
- class L4::Element_not_found

  Exception for a failed lookup (element not found).
- class L4::Invalid_capability

  Indicates that an invalid object was invoked.
- class L4::Com_error

  Error conditions during IPC.
- class L4::Bounds_error

  Access out of bounds.

Macros

- #define L4_CXX_EXCEPTION_BACKTRACE 20

  Number of instruction pointers in backtrace.

13.9.1 Detailed Description

13.10 C++ IPC Interface Definition.

APIs for defining IPC interfaces using C++ as language.

Collaboration diagram for C++ IPC Interface Definition.:

Modules

- Internal Helpers

13.10.1 Detailed Description

APIs for defining IPC interfaces using C++ as language.

13.11 CPU related functions

Collaboration diagram for CPU related functions:

Functions

- int l4util_cpu_has_cpuid (void)

  Check whether the CPU supports the "cpuid" instruction.
- unsigned int l4util_cpu_capabilities (void)

  Returns the CPU capabilities if the "cpuid" instruction is available.
- unsigned int l4util_cpu_capabilities_nocheck (void)

  Returns the CPU capabilities.
- void l4util_cpu_cpuid (unsigned long mode, unsigned long ∗eax, unsigned long ∗ebx, unsigned long ∗ecx, unsigned long ∗edx)

  Generic CPUID access function.

13.11.1 Detailed Description

13.11.2 Function Documentation

13.11.2.1 l4util_cpu_capabilities()

```
unsigned int l4util_cpu_capabilities (
            void  )  [inline]
```

Returns the CPU capabilities if the "cpuid" instruction is available.

Returns

CPU capabilities if the "cpuid" instruction is available, 0 if the "cpuid" instruction is not supported.

Definition at line 97 of file cpu.h.

References l4util_cpu_capabilities_nocheck(), and l4util_cpu_has_cpuid().

Here is the call graph for this function:

13.11.2.2 l4util_cpu_capabilities_nocheck()

```
unsigned int l4util_cpu_capabilities_nocheck (
            void  )  [inline]
```

Returns the CPU capabilities.

Returns

CPU capabilities.

Definition at line 86 of file cpu.h.

References l4util_cpu_cpuid().

Referenced by l4util_cpu_capabilities().

Here is the call graph for this function:

Here is the caller graph for this function:

13.11.2.3 l4util_cpu_has_cpuid()

```
int l4util_cpu_has_cpuid (
            void  )  [inline]
```

Check whether the CPU supports the "cpuid" instruction.

Returns

1 if it has, 0 if it has not

Definition at line 66 of file cpu.h.

Referenced by l4util_cpu_capabilities().

Here is the caller graph for this function:

13.12 Cache Consistency

Various functions for cache consistency.

Collaboration diagram for Cache Consistency:

Functions

- int l4_cache_clean_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache clean a range in D-cache.
- int l4_cache_flush_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache flush a range.
- int l4_cache_inv_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache invalidate a range.
- int l4_cache_coherent (unsigned long start, unsigned long end) L4_NOTHROW

  Make memory coherent between I-cache and D-cache.
- int l4_cache_dma_coherent (unsigned long start, unsigned long end) L4_NOTHROW

  Make memory coherent for use with external memory.
- int l4_cache_dma_coherent_full (void) L4_NOTHROW

  Make memory coherent for use with external memory.

13.12.1 Detailed Description

Various functions for cache consistency.

Include File

```
#include <l4/sys/cache.h>
```

13.12.2 Function Documentation

13.12.2.1 l4_cache_clean_data()

```
int l4_cache_clean_data (
            unsigned long start,
            unsigned long end )  [inline]
```

Cache clean a range in D-cache.

Parameters

start	Start of range (inclusive)
end	End of range (exclusive)

Return values

0	on success
-EFAULT	in the case of an unresolved page fault in the given area

Writes back any dirty cache lines in the range but leaves them in the cache.

Examples

examples/libs/l4re/c++/shared_ds/ds_clnt.cc.

Definition at line 81 of file cache.h.

13.12.2.2 l4_cache_coherent()

```
int l4_cache_coherent (
            unsigned long start,
            unsigned long end )  [inline]
```

Make memory coherent between I-cache and D-cache.

Parameters

start	Start of range (inclusive)
end	End of range (exclusive)

Return values

0	on success
-EFAULT	in the case of an unresolved page fault in the given area

Definition at line 105 of file cache.h.

13.12.2.3 l4_cache_dma_coherent()

```
int l4_cache_dma_coherent (
            unsigned long start,
            unsigned long end )  [inline]
```

Make memory coherent for use with external memory.

Parameters

start	Start of range (inclusive)
end	End of range (exclusive)

Return values

0	on success
-EFAULT	in the case of an unresolved page fault in the given area

Definition at line 113 of file cache.h.

13.12.2.4 l4_cache_flush_data()

```
int l4_cache_flush_data (
            unsigned long start,
            unsigned long end )  [inline]
```

Cache flush a range.

Parameters

start	Start of range (inclusive)
end	End of range (exclusive)

Return values

0	on success
-EFAULT	in the case of an unresolved page fault in the given area

Writes back any dirty cache lines and invalidates all cache entries in the range.

Definition at line 89 of file cache.h.

13.12.2.5 l4_cache_inv_data()

```
int l4_cache_inv_data (
            unsigned long start,
            unsigned long end )  [inline]
```

Cache invalidate a range.

Parameters

start	Start of range (inclusive)
end	End of range (exclusive)

Return values

0	on success
-EFAULT	in the case of an unresolved page fault in the given area

Invalidates all cache entries in the range but does not necessarily write back dirty cache lines.

Note

Implementations may choose to write back dirty lines nonetheless if this is more efficient.

Definition at line 97 of file cache.h.

13.13 Capabilities

C interface for capabilities.

Collaboration diagram for Capabilities:

Typedefs

- typedef unsigned long l4_cap_idx_t

  L4 Capability selector Type.

Enumerations

- enum l4_cap_consts_t { L4_CAP_SHIFT , L4_CAP_SIZE , **L4_CAP_OFFSET** = 1UL << L4_CAP_SHIFT , L4_CAP_MASK , L4_INVALID_CAP , **L4_INVALID_CAP_BIT** = 1UL << (L4_CAP_SHIFT - 1) }

  Constants related to capability selectors.
- enum l4_default_caps_t { L4_BASE_TASK_CAP , L4_BASE_FACTORY_CAP , L4_BASE_THREAD_CAP , L4_BASE_PAGER_CAP , L4_BASE_LOG_CAP , L4_BASE_ICU_CAP , L4_BASE_SCHEDULER_CAP , L4_BASE_IOMMU_CAP , L4_BASE_DEBUGGER_CAP , L4_BASE_ARM_SMCCC_CAP , **L4_BASE_CAPS_LAST_P1** , L4_BASE_CAPS_LAST = L4_BASE_CAPS_LAST_P1 - 1 }

  Default capabilities setup for the initial tasks.

Functions

- unsigned l4_is_invalid_cap (l4_cap_idx_t c) L4_NOTHROW

  Test if a capability selector is the invalid capability.
- unsigned l4_is_valid_cap (l4_cap_idx_t c) L4_NOTHROW

  Test if a capability selector is a valid selector.
- unsigned l4_capability_equal (l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW

  Test if two capability selectors are equal.

13.13.1 Detailed Description

C interface for capabilities.

Add

```
#include <l4/sys/types.h>
#include <l4/sys/consts.h>
```

to your code to use the functions and definitions explained here.

13.13.2 Enumeration Type Documentation

13.13.2.1 l4_cap_consts_t

```
enum l4_cap_consts_t
```

Constants related to capability selectors.

Enumerator

L4_CAP_SHIFT	Capability index shift.
L4_CAP_SIZE	Offset of two consecutive capability selectors.
L4_CAP_MASK	Mask to get only the relevant bits of an l4_cap_idx_t.
L4_INVALID_CAP	Invalid capability selector.

Definition at line 128 of file consts.h.

13.13.2.2 l4_default_caps_t

`enum l4_default_caps_t`

Default capabilities setup for the initial tasks.

These capability selectors are setup per default by the micro kernel for the two initial tasks, the Root-Pager (Sigma0) and the Root-Task (Moe).

Attention

These constants do not have any particular meaning for applications started by Moe, see Initial Environment for this kind of information.

See also

Initial Environment for information useful for normal user applications.

Enumerator

L4_BASE_TASK_CAP	Capability selector for the current task.
L4_BASE_FACTORY_CAP	Capability selector for the factory.
L4_BASE_THREAD_CAP	Capability selector for the first thread.
L4_BASE_PAGER_CAP	Capability selector for the pager gate. For Sigma0, the pager is not present since it never raises page faults. For Moe, the pager is set to Sigma0.
L4_BASE_LOG_CAP	Capability selector for the log object. Present if the corresponding feature is turned on in the microkernel configuration.
L4_BASE_ICU_CAP	Capability selector for the base icu object.
L4_BASE_SCHEDULER_CAP	Capability selector for the scheduler cap.
L4_BASE_IOMMU_CAP	Capability selector for the IO-MMU cap. Present if the microkernel detected an IO-MMU.
L4_BASE_DEBUGGER_CAP	Capability selector for the debugger cap. Present if the corresponding feature is turned on in the microkernel configuration.
L4_BASE_ARM_SMCCC_CAP	Capability selector for the ARM SMCCC cap. Present if the microkernel detected an ARM SMC capable trusted execution environment.
L4_BASE_CAPS_LAST	Last capability index used for base capabilities.

Definition at line 246 of file consts.h.

13.13.3 Function Documentation

13.13.3.1 l4_capability_equal()

```
unsigned l4_capability_equal (
            l4_cap_idx_t c1,
            l4_cap_idx_t c2 )  [inline]
```

Test if two capability selectors are equal.

Parameters

c1	Capability
c2	Capability

Return values

0	The given capabilities are not in the same slot.
1	The given capabilities are in the same slot.

Definition at line 403 of file types.h.

References L4_CAP_SHIFT.

13.13.3.2 l4_is_invalid_cap()

```
unsigned l4_is_invalid_cap (
            l4_cap_idx_t c )  [inline]
```

Test if a capability selector is the invalid capability.

Parameters

c	Capability selector

Return values

0	The capability selector is not the invalid capability.
>0	The capability selector is the invalid capability.

Examples

examples/libs/l4re/c/ma+rm.c, examples/sys/aliens/main.c, examples/sys/isr/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 395 of file types.h.

13.13.3.3 l4_is_valid_cap()

```
unsigned l4_is_valid_cap (
            l4_cap_idx_t c ) [inline]
```

Test if a capability selector is a valid selector.

Parameters

c	Capability selector

Return values

0	The capability selector is not valid.
>0	The capability selector is valid.

Definition at line 399 of file types.h.

Referenced by L4Re::Util::Cap_alloc_base::free(), and L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free().

Here is the caller graph for this function:

13.14 Capability allocator

Capability allocator C interface.

Collaboration diagram for Capability allocator:

Functions

- l4_cap_idx_t l4re_util_cap_alloc (void) L4_NOTHROW

  Get free capability index at capability allocator.
- void l4re_util_cap_free (l4_cap_idx_t cap) L4_NOTHROW

  Return capability index to capability allocator.
- void l4re_util_cap_free_um (l4_cap_idx_t cap) L4_NOTHROW

  Return capability index to capability allocator, and unmaps the object.
- long l4re_util_cap_last (void) L4_NOTHROW

  Return last capability index the allocator can return.

13.14.1 Detailed Description

Capability allocator C interface.

13.14.2 Function Documentation

13.14.2.1 l4re_util_cap_last()

```
long l4re_util_cap_last (
            void  )
```

Return last capability index the allocator can return.

Returns

last/biggest capability index the allocator can return

13.15 Chunks

Collaboration diagram for Chunks:

Modules

- Consumer
- Producer

Functions

- long l4shmc_add_chunk (l4shmc_area_t ∗shmarea, const char ∗chunk_name, l4_umword_t chunk_capacity, l4shmc_chunk_t ∗chunk)

  Add a chunk in the shared memory area.
- long l4shmc_get_chunk (l4shmc_area_t ∗shmarea, const char ∗chunk_name, l4shmc_chunk_t ∗chunk)

  Get chunk out of shared memory area.
- long l4shmc_get_chunk_to (l4shmc_area_t ∗shmarea, const char ∗chunk_name, l4_umword_t timeout_ms, l4shmc_chunk_t ∗chunk)

  Get chunk out of shared memory area, with timeout.
- long l4shmc_iterate_chunk (l4shmc_area_t ∗shmarea, const char ∗∗chunk_name, long offs)

  Iterate over names of all existing chunks.
- void ∗ l4shmc_chunk_ptr (l4shmc_chunk_t ∗chunk)

  Get data pointer to chunk.
- long l4shmc_chunk_capacity (l4shmc_chunk_t ∗chunk)

  Get capacity of a chunk.
- l4shmc_signal_t ∗ l4shmc_chunk_signal (l4shmc_chunk_t ∗chunk)

  Get the signal of a chunk.

13.15.1 Detailed Description

13.15.2 Function Documentation

13.15.2.1 l4shmc_add_chunk()

```
long l4shmc_add_chunk (
            l4shmc_area_t * shmarea,
            const char * chunk_name,
            l4_umword_t chunk_capacity,
            l4shmc_chunk_t * chunk )
```

Add a chunk in the shared memory area.

Parameters

	shmarea	The shared memory area to put the chunk in.
	chunk_name	Name of the chunk.
	chunk_capacity	Capacity for payload of the chunk in bytes.
out	*chunk*	Chunk structure to fill in.

Return values

0	Success.
<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.15.2.2 l4shmc_chunk_capacity()

```
long l4shmc_chunk_capacity (
            l4shmc_chunk_t * chunk )  [inline]
```

Get capacity of a chunk.

Parameters

chunk	Chunk.

Return values

0	Success.
<0	Error.

13.15.2.3 l4shmc_chunk_ptr()

```
void* l4shmc_chunk_ptr (
            l4shmc_chunk_t * chunk )  [inline]
```

Get data pointer to chunk.

Parameters

chunk	Chunk.

Return values

0	Success.
<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.15.2.4 l4shmc_chunk_signal()

```
l4shmc_signal_t* l4shmc_chunk_signal (
            l4shmc_chunk_t * chunk )  [inline]
```

Get the signal of a chunk.

Parameters

chunk	Chunk.

Return values

0	No signal has been registered with this chunk.
>0	Pointer to signal otherwise.

13.15.2.5 l4shmc_get_chunk()

```
long l4shmc_get_chunk (
            l4shmc_area_t * shmarea,
            const char * chunk_name,
            l4shmc_chunk_t * chunk )  [inline]
```

Get chunk out of shared memory area.

Parameters

	shmarea	Shared memory area.
	chunk_name	Name of the chunk.
out	*chunk*	Chunk data structure to fill.

Return values

0	Success.
<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.15.2.6 l4shmc_get_chunk_to()

```
long l4shmc_get_chunk_to (
            l4shmc_area_t * shmarea,
            const char * chunk_name,
            l4_umword_t timeout_ms,
            l4shmc_chunk_t * chunk )
```

Get chunk out of shared memory area, with timeout.

Parameters

	shmarea	Shared memory area.
	chunk_name	Name of the chunk.
	timeout_ms	Timeout in milliseconds to wait for the chunk to appear in the shared memory area.
out	*chunk*	Chunk data structure to fill.

Return values

0	Success.
<0	Error.

13.15.2.7 l4shmc_iterate_chunk()

```
long l4shmc_iterate_chunk (
            l4shmc_area_t * shmarea,
            const char ** chunk_name,
            long offs )
```

Iterate over names of all existing chunks.

Parameters

shmarea	Shared memory area.
chunk_name	Where the name of the current chunk will be stored
offs	0 to start iteration, return value of previous call to l4shmc_iterate_chunk() to get next chunk

Return values

0	No more chunks available.
<0	Error.
>0	Iterator value for the next call.

13.16 Comfortable Command Line Parsing

Collaboration diagram for Comfortable Command Line Parsing:

Typedefs

- typedef void(∗ parse_cmd_fn_t) (int)

  Function type for PARSE_CMD_FN.
- typedef void(∗ parse_cmd_fn_arg_t) (int, const char ∗, int)

  Function type for PARSE_CMD_FN_ARG.

Enumerations

- enum parse_cmd_type

  Types for parsing.

Functions

- int parse_cmdline (int ∗argc, const char ∗∗∗argv, char arg0,...)

  Parse the command-line for specified arguments and store the values into variables.

13.16.1 Detailed Description

13.16.2 Function Documentation

13.16.2.1 parse_cmdline()

```
int parse_cmdline (
            int * argc,
            const char *** argv,
            char arg0,
             ...  )
```

Parse the command-line for specified arguments and store the values into variables.

This Functions gets the command-line, and a list of command-descriptors. Then, the command-line is parsed according to the given descriptors, storing strings, switches and numeric arguments at given addresses, and possibly calling specified functions. A default help descriptor is added. Its purpose is to present a short command overview in the case the given command-line does not fit to the descriptors.

Each command-descriptor has the following form:

short option char, *long option name*, *comment*, *type*, *val*, *addr*.

The *short option char* specifies the short form of the described option. The short form will be recognized after a single dash, or in a group of short options preceeded by a single dash. Specify ' ' if no short form should be used.

The *long option name* specifies the long form of the described option. The long form will be recognized after two dashes. Specify 0 if no long form should be used for this option.

The *comment* is a string that will be used when presenting the short command-line help.

The *type* specifies, if the option should be recognized as

- a number (`PARSE_CMD_INT`),
- a switch (`PARSE_CMD_SWITCH`),
- a string (`PARSE_CMD_STRING`),
- a function call (`PARSE_CMD_FN`, `PARSE_CMD_FN_ARG`),
- an increment/decrement operator (`PARSE_CMD_INC`, `PARSE_CMD_DEC`).

If type is `PARSE_CMD_INT`, the option requires a second argument on the command-line after the option. This argument is parsed as a number. It can be preceeded by 0x to present a hex-value or by 0 to present an octal form. *addr* is interpreted as an int-pointer. The scanned argument from the command-line is stored in this pointer.

If *type* is `PARSE_CMD_SWITCH`, *addr* must be a pointer to int, and the value from *val* is stored at this pointer.

With `PARSE_CMD_STRING`, an additional argument is expected at the cmdline. *addr* must be a pointer to const char*, and a pointer to the argument on the command line is stored at this pointer. The value in *val* is a default value, which is stored at *addr* if the corresponding option is not given on the command line.

With `PARSE_CMD_FN_ARG`, *addr* is interpreted as a function pointer of type parse_cmd_fn_t. It will be called with *val* as argument if the corresponding option is found.

If *type* is `PARSE_CMD_FN_ARG`, *addr* is as a function pointer of type parse_cmd_fn_arg_t, and handled similar to `PARSE_CMD_FN`. An additional argument is expected at the command line, however. It is given to the called function as 2nd argument, and parsed as an integer as with `PARSE_CMD_INT` as a third argument.

If *type* is `PARSE_CMD_INC` or `PARSE_CMD_DEC`, *addr* is interpreted as an int-pointer. The value of *val* is stored to this pointer first. For every occurence of the option in the command line, the integer referenced by *addr* is incremented or decremented, respectively.

The list of command-descriptors is terminated by specifying a binary 0 for the short option char.

Note: The short option char 'h' and the long option name "help" must not be specified. They are used for the default help descriptor and produce a short command-options help when specified on the command-line.

Parameters

argc	pointer to number of command line parameters as passed to main
argv	pointer to array of command line parameters as passed to main
arg0	format list describing the command line options to parse for

Returns

0 if the command-line was successfully parsed, otherwise:

- -1 if the given descriptors are somehow wrong.
- -2 if not enough memory was available to hold temporary structs.
- -3 if the given command-line args did not meet the specified set.
- -4 if the help-option was given.

Upon return, argc and argv point to a list of arguments that were not scanned as arguments. See `getoptlong` for details on scanning.

13.17 Console API

Console interface.

Collaboration diagram for Console API:

Data Structures

- class L4Re::Console

  Console class.

13.17.1 Detailed Description

Console interface.

13.18 Consumer

Collaboration diagram for Consumer:

Functions

- long l4shmc_enable_chunk (l4shmc_chunk_t ∗chunk)

  Enable a signal connected with a chunk.
- long l4shmc_wait_chunk (l4shmc_chunk_t ∗chunk)

  Wait on a specific chunk.
- long l4shmc_wait_chunk_to (l4shmc_chunk_t ∗chunk, l4_timeout_t timeout)

  Check whether a specific chunk has an event pending, with timeout.
- long l4shmc_wait_chunk_try (l4shmc_chunk_t ∗chunk)

  Check whether a specific chunk has an event pending.
- long l4shmc_chunk_consumed (l4shmc_chunk_t ∗chunk)

  Mark a chunk as free.
- long l4shmc_is_chunk_ready (l4shmc_chunk_t ∗chunk)

  Check whether data is available.
- long l4shmc_chunk_size (l4shmc_chunk_t ∗chunk)

  Get current size of a chunk.

13.18.1 Detailed Description

13.18.2 Function Documentation

13.18.2.1 l4shmc_chunk_consumed()

```
long l4shmc_chunk_consumed (
            l4shmc_chunk_t * chunk )  [inline]
```

Mark a chunk as free.

Parameters

chunk	Chunk to mark as free.

Return values

0	Success.
<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.18.2.2 l4shmc_chunk_size()

```
long l4shmc_chunk_size (
            l4shmc_chunk_t * chunk )  [inline]
```

Get current size of a chunk.

Parameters

chunk	Chunk.

Return values

0	Success.
<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.18.2.3 l4shmc_enable_chunk()

```
long l4shmc_enable_chunk (
            l4shmc_chunk_t * chunk )
```

Enable a signal connected with a chunk.

Parameters

chunk	Chunk to enable.

Return values

0	Success.
<0	Error.

A signal must be enabled before waiting when the consumer waits on any signal. Enabling is not needed if the consumer waits for a specific signal or chunk.

13.18.2.4 l4shmc_is_chunk_ready()

```
long l4shmc_is_chunk_ready (
            l4shmc_chunk_t * chunk )  [inline]
```

Check whether data is available.

Parameters

chunk	Chunk to check.

Return values

0	Success.
<0	Error.

13.18.2.5 l4shmc_wait_chunk()

```
long l4shmc_wait_chunk (
            l4shmc_chunk_t * chunk )  [inline]
```

Wait on a specific chunk.

Parameters

chunk	Chunk to wait for.

Return values

0	Success.
<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.18.2.6 l4shmc_wait_chunk_to()

```
long l4shmc_wait_chunk_to (
            l4shmc_chunk_t * chunk,
            l4_timeout_t timeout )
```

Check whether a specific chunk has an event pending, with timeout.

Parameters

chunk	Chunk to check.
timeout	Timeout.

Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

13.18.2.7 l4shmc_wait_chunk_try()

```
long l4shmc_wait_chunk_try (
            l4shmc_chunk_t * chunk )  [inline]
```

Check whether a specific chunk has an event pending.

Parameters

chunk	Chunk to check.

Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

13.19 Consumer

Collaboration diagram for Consumer:

Functions

- long l4shmc_enable_signal (l4shmc_signal_t *signal)

  Enable a signal.
- long l4shmc_wait_any (l4shmc_signal_t **retsignal)

  Wait on any signal.
- long l4shmc_wait_any_try (l4shmc_signal_t **retsignal)

  Check whether any waited signal has an event pending.
- long l4shmc_wait_any_to (l4_timeout_t timeout, l4shmc_signal_t **retsignal)

  Wait for any signal with timeout.
- long l4shmc_wait_signal (l4shmc_signal_t *signal)

  Wait on a specific signal.
- long l4shmc_wait_signal_to (l4shmc_signal_t *signal, l4_timeout_t timeout)

  Wait on a specific signal, with timeout.
- long l4shmc_wait_signal_try (l4shmc_signal_t *signal)

  Check whether a specific signal has an event pending.

13.19.1 Detailed Description

13.19.2 Function Documentation

13.19.2.1 l4shmc_enable_signal()

```
long l4shmc_enable_signal (
            l4shmc_signal_t * signal )
```

Enable a signal.

Parameters

signal	Signal to enable.

Return values

0	Success.
<0	Error.

A signal must be enabled before waiting when the consumer waits on any signal. Enabling is not needed if the consumer waits for a specific signal or chunk.

13.19.2.2 l4shmc_wait_any()

```
long l4shmc_wait_any (
            l4shmc_signal_t ** retsignal )  [inline]
```

Wait on any signal.

Parameters

out	*retsignal*	Signal received.

Return values

0	Success.
<0	Error.

13.19.2.3 l4shmc_wait_any_to()

```
long l4shmc_wait_any_to (
            l4_timeout_t timeout,
            l4shmc_signal_t ** retsignal )
```

Wait for any signal with timeout.

Parameters

	timeout	Timeout.
out	*retsignal*	Signal that has the event pending if any.

Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

13.19.2.4 l4shmc_wait_any_try()

```
long l4shmc_wait_any_try (
            l4shmc_signal_t ** retsignal )  [inline]
```

Check whether any waited signal has an event pending.

Parameters

out	*retsignal*	Signal that has the event pending if any.

Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

13.19.2.5 l4shmc_wait_signal()

```
long l4shmc_wait_signal (
            l4shmc_signal_t * signal )  [inline]
```

Wait on a specific signal.

Parameters

signal	Signal to wait for.

Return values

0	Success.
<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.19.2.6 l4shmc_wait_signal_to()

```
long l4shmc_wait_signal_to (
            l4shmc_signal_t * signal,
            l4_timeout_t timeout )
```

Wait on a specific signal, with timeout.

Parameters

signal	Signal to wait for.
timeout	Timeout.

Return values

0	Success.
<0	Error.

13.19.2.7 l4shmc_wait_signal_try()

```
long l4shmc_wait_signal_try (
```

```
l4shmc_signal_t * signal )  [inline]
```

Check whether a specific signal has an event pending.

Parameters

signal	Signal to check.

Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

13.20 DMA Space Interface

DMA Space C interface.

Collaboration diagram for DMA Space Interface:

Typedefs

- typedef l4_cap_idx_t l4re_dma_space_t

  DMA space capability type.

Functions

- long l4re_dma_space_map (l4re_dma_space_t dma, l4re_ds_t src, l4re_ds_offset_t offset, l4_size_t *size, unsigned long attrs, enum l4re_dma_space_direction dir, l4re_dma_space_dma_addr_t *dma_addr) L4_NOTHROW

  Map the given part of this data space into the DMA address space.
- long l4re_dma_space_unmap (l4re_dma_space_t dma, l4re_dma_space_dma_addr_t dma_addr, l4_size_t size, unsigned long attrs, enum l4re_dma_space_direction dir) L4_NOTHROW

  Unmap the given part of this data space from the DMA address space.
- long l4re_dma_space_associate (l4re_dma_space_t dma, l4_cap_idx_t dma_task, unsigned long attr) L4_NOTHROW

  Associate a DMA task for a device to this Dma_space.
- long l4re_dma_space_disassociate (l4re_dma_space_t dma)

  Disassociate the DMA task from this Dma_space.

13.20.1 Detailed Description

DMA Space C interface.

13.20.2 Typedef Documentation

13.20.2.1 l4re_dma_space_t

```
typedef l4_cap_idx_t l4re_dma_space_t
```

DMA space capability type.

DMA Address Space. A Dma_space represents the DMA address space of a device. Whenever a device needs direct access to parts of an L4Re::Dataspace, that part of the data space must be mapped to the DMA address space that is assigned to that device. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a DMA address space, using map(), makes the given parts of the data space visible to the associated device at the returned DMA address. As long as the memory is mapped into a DMA space it is 'pinned' and cannot be subject to dynamic memory management such as swapping. Additionally, map() is responsible for the necessary syncing operations before the DMA.

unmap() is the reverse operation to map() and unmaps the given data-space part for the DMA address space. unmap() is responsible for the necessary sync operations after the DMA.

Definition at line 59 of file dma_space.h.

13.20.3 Function Documentation

13.20.3.1 l4re_dma_space_associate()

```
long l4re_dma_space_associate (
            l4re_dma_space_t dma,
            l4_cap_idx_t dma_task,
            unsigned long attr )
```

Associate a DMA task for a device to this Dma_space.

Parameters

dma	DMA space capability

Precondition

requires capability rights: {RW}

Parameters

in	*dma_task*	The DMA task used for the device that shall be associated with this DMA space. The dma_task might be an invalid capability when L4Re::Dma_space::Phys_space is set in `attr`, in this case the CPUs physical memory is used as DMA address space.
in	*attr*	Attributes for this DMA space. See L4Re::Dma_space::Space_attrib.

13.20.3.2 l4re_dma_space_disassociate()

```
long l4re_dma_space_disassociate (
            l4re_dma_space_t dma )
```

Disassociate the DMA task from this Dma_space.

Parameters

dma	DMA space capability

Precondition

requires capability rights: {RW}

13.20.3.3 l4re_dma_space_map()

```
long l4re_dma_space_map (
            l4re_dma_space_t dma,
            l4re_ds_t src,
            l4re_ds_offset_t offset,
            l4_size_t * size,
            unsigned long attrs,
            enum l4re_dma_space_direction dir,
            l4re_dma_space_dma_addr_t * dma_addr )
```

Map the given part of this data space into the DMA address space.

Parameters

dma	DMA space capability

Precondition

requires capability rights: {R}

Parameters

in	*src*	Source data space (that describes the memory). Caller needs write rights to the data space.
in	*offset*	The offset (bytes) within `src`.
in,out	*size*	The size (bytes) of the region to be mapped for DMA, after successful mapping the size returned is the size mapped for DMA as a single block. This size might be smaller than the original input size, in this case the caller might call map() again with a new offset and the remaining size.
in	*attrs*	The attributes used for this DMA mapping (a combination of Dma_space::Attribute values).
in	*dir*	The direction of the DMA transfer issued with this mapping. The same value must later be passed to unmap().
out	*dma_addr*	The DMA address to use for DMA with the associated device.

Returns

0 in the case of success, a negative error code otherwise.

13.20.3.4 l4re_dma_space_unmap()

```
long l4re_dma_space_unmap (
            l4re_dma_space_t dma,
            l4re_dma_space_dma_addr_t dma_addr,
            l4_size_t size,
            unsigned long attrs,
            enum l4re_dma_space_direction dir )
```

Unmap the given part of this data space from the DMA address space.

Parameters

dma	DMA space capability

Precondition

requires capability rights: {R}

Parameters

dma_addr	The DMA address (returned by Dma_space::map()).
size	The size (bytes) of the memory region to unmap.
attrs	The attributes for the unmap (currently none).
dir	The direction of the finished DMA operation.

13.21 DMA space

A DMA space represents a device memory address space managed by an IOMMU.

Collaboration diagram for DMA space:

A DMA space represents a device memory address space managed by an IOMMU.

That is, it manages the translation of virtual addresses used by devices to physical addresses. It is accessed via the L4::Task interface, but with the following caveats:

- No threads can be bound to it.
- No objects (and IO ports on IA32) can be mapped to it.
- No kernel-user memory can be added to it.
- It must be constructed by passing the L4_PROTO_DMA_SPACE protocoll constant to the kernel factory's L4::Factory.create() call.

A DMA space must be bound to an L4::Iommu to enable the address translation for specific devices.

The kernel factory allows to create DMA spaces only if the kernel has been configured with IOMMU support and if an IOMMU was detected.

13.22 Dataspace interface

Dataspace C interface.

Collaboration diagram for Dataspace interface:

Data Structures

- struct l4re_ds_stats_t

  Information about the data space.

Typedefs

- typedef l4_cap_idx_t l4re_ds_t

  Dataspace type.
- typedef l4_cap_idx_t l4re_namespace_t

  Namespace type.

Enumerations

- enum l4re_ds_map_flags { }

  Flags to specify the memory mapping type of a request.

Functions

- long l4re_ds_clear (l4re_ds_t ds, l4re_ds_offset_t offset, l4re_ds_size_t size) L4_NOTHROW
- long l4re_ds_allocate (l4re_ds_t ds, l4re_ds_offset_t offset, l4re_ds_size_t size) L4_NOTHROW
- int l4re_ds_copy_in (l4re_ds_t ds, l4re_ds_offset_t dst_offs, l4re_ds_t src, l4re_ds_offset_t src_offs, l4re_ds_size_t size) L4_NOTHROW
- l4re_ds_size_t l4re_ds_size (l4re_ds_t ds) L4_NOTHROW
- l4re_ds_flags_t l4re_ds_flags (l4re_ds_t ds) L4_NOTHROW
- int l4re_ds_info (l4re_ds_t ds, l4re_ds_stats_t *stats) L4_NOTHROW

13.22.1 Detailed Description

Dataspace C interface.

13.22.2 Enumeration Type Documentation

13.22.2.1 l4re_ds_map_flags

`enum l4re_ds_map_flags`

Flags to specify the memory mapping type of a request.

Enumerator

L4RE_DS_F_NORMAL	request normal memory mapping
L4RE_DS_F_CACHEABLE	request normal memory mapping
L4RE_DS_F_BUFFERABLE	request bufferable (write buffered) mappings
L4RE_DS_F_UNCACHEABLE	request uncacheable memory mappings
L4RE_DS_F_CACHING_MASK	mask for caching flags
L4RE_DS_F_CACHING_SHIFT	shift value for caching flags

Definition at line 58 of file dataspace.h.

13.22.3 Function Documentation

13.22.3.1 l4re_ds_allocate()

```
long l4re_ds_allocate (
            l4re_ds_t ds,
            l4re_ds_offset_t offset,
            l4re_ds_size_t size )
```

Returns

0 on success, <0 on errors

See also

L4Re::Dataspace::allocate

13.22.3.2 l4re_ds_clear()

```
long l4re_ds_clear (
            l4re_ds_t ds,
            l4re_ds_offset_t offset,
            l4re_ds_size_t size )
```

Returns

0 on success, <0 on errors

See also

L4Re::Dataspace::clear

13.22.3.3 l4re_ds_copy_in()

```
int l4re_ds_copy_in (
            l4re_ds_t ds,
            l4re_ds_offset_t dst_offs,
            l4re_ds_t src,
            l4re_ds_offset_t src_offs,
            l4re_ds_size_t size )
```

Returns

0 on success, <0 on errors

See also

L4Re::Dataspace::copy_in

13.22.3.4 l4re_ds_flags()

```
l4re_ds_flags_t l4re_ds_flags (
            l4re_ds_t ds )
```

See also

L4Re::Dataspace::flags

13.22.3.5 l4re_ds_info()

```
int l4re_ds_info (
            l4re_ds_t ds,
            l4re_ds_stats_t * stats )
```

See also

L4Re::Dataspace::info

13.22.3.6 l4re_ds_size()

```
l4re_ds_size_t l4re_ds_size (
            l4re_ds_t ds )
```

Returns

Size of the dataspace in bytes.

See also

L4Re::Dataspace::size

13.23 Debug interface

Collaboration diagram for Debug interface:

Functions

- long l4re_debug_obj_debug (l4_cap_idx_t srv, unsigned long function) L4_NOTHROW

  Call debug function of L4Re service.

13.23.1 Detailed Description

13.23.2 Function Documentation

13.23.2.1 l4re_debug_obj_debug()

```
long l4re_debug_obj_debug (
            l4_cap_idx_t srv,
            unsigned long function )
```

Call debug function of L4Re service.

Parameters

srv	Object to call.
function	Function to call.

See also

L4Re::Debug_obj::debug

13.24 Debugging API

Debugging Interface.

Collaboration diagram for Debugging API:

Data Structures

- class L4Re::Debug_obj

  Debug interface.

13.24.1 Detailed Description

Debugging Interface.

The debugging interface can be provided to retrieve, or log debugging information for an object. Each class may realize the debug interface to provide debugging functionality. For example, the region map objects provide a facility to dump the currently established memory regions.

See also

L4::Debug_obj for more information.

13.25 EDID parsing functionality

Enumerations

- enum Libedid_consts { Libedid_block_size = 128 }

  EDID constants.

Functions

- int libedid_check_header (const unsigned char ∗edid)

  Check for valid EDID header.
- int libedid_checksum (const unsigned char ∗edid)

  Calculates the EDID checksum.
- unsigned libedid_version (const unsigned char ∗edid)

  Returns the EDID version number.
- unsigned libedid_revision (const unsigned char ∗edid)

  Returns the EDID revision number.
- void libedid_pnp_id (const unsigned char ∗edid, unsigned char ∗id)

  Extracts the display's PnP ID.
- void libedid_prefered_resolution (const unsigned char ∗edid, unsigned ∗w, unsigned ∗h)

  Extract the display's prefered mode.
- unsigned libedid_num_ext_blocks (const unsigned char ∗edid)

  Get the number of EDID extension blocks.
- unsigned libedid_dump_standard_timings (const unsigned char ∗edid)

  Dump the standard timings to stdout.
- void libedid_dump (const unsigned char ∗edid)

  Dump raw EDID data to stdout.

13.25.1 Detailed Description

13.25.2 Enumeration Type Documentation

13.25.2.1 Libedid_consts

`enum Libedid_consts`

EDID constants.

Enumerator

Libedid_block_size	Size of one EDID block in bytes.

Definition at line 23 of file edid.h.

13.25.3 Function Documentation

13.25.3.1 libedid_check_header()

```
int libedid_check_header (
            const unsigned char * edid )
```

Check for valid EDID header.

Parameters

edid	Pointer to a 128byte EDID block

Returns

0 if the header is correct, -EINVAL otherwise

13.25.3.2 libedid_checksum()

```
int libedid_checksum (
            const unsigned char * edid )
```

Calculates the EDID checksum.

Parameters

edid	Pointer to a 128byte EDID block

Returns

0 if checksum is correct, -EINVAL otherwise

13.25.3.3 libedid_dump()

```
void libedid_dump (
            const unsigned char * edid )
```

Dump raw EDID data to stdout.

Parameters

edid	Pointer to a 128byte EDID block

13.25.3.4 libedid_dump_standard_timings()

```
unsigned libedid_dump_standard_timings (
            const unsigned char * edid )
```

Dump the standard timings to stdout.

Parameters

edid	Pointer to a 128byte EDID block

Returns

Number of standard timings stored in EDID

13.25.3.5 libedid_num_ext_blocks()

```
unsigned libedid_num_ext_blocks (
            const unsigned char * edid )
```

Get the number of EDID extension blocks.

Parameters

edid	Pointer to a 128byte EDID block

Returns

Number of EDID extension blocks

13.25.3.6 libedid_pnp_id()

```
void libedid_pnp_id (
            const unsigned char * edid,
            unsigned char * id )
```

Extracts the display's PnP ID.

Parameters

edid	Pointer to a 128byte EDID block

Return values

id	Return the PnP id. Must point to 4 bytes.

13.25.3.7 libedid_prefered_resolution()

```
void libedid_prefered_resolution (
            const unsigned char * edid,
            unsigned * w,
            unsigned * h )
```

Extract the display's prefered mode.

Parameters

edid	Pointer to a 128byte EDID block

Return values

w	X resolution of prefered video mode in pixels.
h	Y resolution of prefered video mode in pixels.

13.25.3.8 libedid_revision()

```
unsigned libedid_revision (
            const unsigned char * edid )
```

Returns the EDID revision number.

Parameters

edid	Pointer to a 128 EDID block

Returns

Revision number

13.25.3.9 libedid_version()

```
unsigned libedid_version (
            const unsigned char * edid )
```

Returns the EDID version number.

Parameters

edid	Pointer to a 128byte EDID block

Returns

Version number

13.26 ELF binary format

Functions and types related to ELF binaries.

Collaboration diagram for ELF binary format:

Files

- file elf.h

  ELF definition.

Data Structures

- struct Elf32_Ehdr

  ELF32 header.
- struct Elf64_Ehdr

  ELF64 header.
- struct Elf32_Shdr

  ELF32 section header.
- struct Elf64_Shdr

  ELF64 section header.
- struct Elf32_Phdr

  ELF32 program header.
- struct Elf64_Phdr

ELF64 program header.

- struct Elf32_Dyn

  ELF32 dynamic entry.
- struct Elf64_Dyn

  ELF64 dynamic entry.
- struct Elf32_Rel

  ELF32 relocation entry w/o addend.
- struct Elf32_Rela

  ELF32 relocation entry w/ addend.
- struct Elf64_Rel

  ELF64 relocation entry w/o addend.
- struct Elf64_Rela

  ELF64 relocation entry w/ addend.
- struct Elf32_Sym

  ELF32 symbol table entry.
- struct Elf64_Sym

  ELF64 symbol table entry.
- struct Elf32_Auxv

  Auxiliary vector (32-bit).
- struct Elf64_Auxv

  Auxiliary vector (64-bit).

Macros

- #define ElfW(type) _ElfW(Elf, 32, type)

  Use 64 or 32 bits types depending on the target architecture.
- #define ELF32_R_SYM(i) ((i)>>8)

  Symbol table index.
- #define ELF32_R_TYPE(i) ((unsigned char)(i))
- #define ELF32_R_INFO(s, t) (((s)<<8)+(unsigned char)(t))

  Create info from symbol table index + type.
- #define ELF64_R_SYM(i) ((i)>>32)

  Symbol table index.
- #define ELF64_R_TYPE(i) ((i)&0xffffffffL)
- #define ELF64_R_INFO(s, t) (((s)<<32)+(t)&0xffffffffL)

  Create info from symbol table index + type.
- #define ELF32_ST_BIND(i) ((i)>>4)
- #define ELF32_ST_TYPE(i) ((i)&0xf)
- #define ELF32_ST_INFO(b, t) (((b)<<4)+((t)&0xf))

  Make info from bind + type.
- #define ELF64_ST_BIND(i) ((i)>>4)
- #define ELF64_ST_TYPE(i) ((i)&0xf)
- #define ELF64_ST_INFO(b, t) (((b)<<4)+((t)&0xf))

  Make info from bind + type.

Typedefs

- typedef struct Elf32_Auxv Elf32_Auxv

  Auxiliary vector (32-bit).
- typedef struct Elf64_Auxv Elf64_Auxv

  Auxiliary vector (64-bit).

Enumerations

- enum { EI_NIDENT = 16 }
- enum Elf_ETs {
  ET_NONE = 0 , ET_REL = 1 , ET_EXEC = 2 , ET_DYN = 3 ,
  ET_CORE = 4 , ET_LOPROC = 0xff00 , ET_HIPROC = 0xffff }

  Object file type.
- enum Elf_EMs {
  EM_NONE = 0 , EM_M32 = 1 , EM_SPARC = 2 , EM_386 = 3 ,
  EM_68K = 4 , EM_88K = 5 , EM_860 = 7 , EM_MIPS = 8 ,
  EM_MIPS_RS4_BE = 10 , EM_SPARC64 = 11 , EM_PARISC = 15 , EM_VPP500 = 17 ,
  EM_SPARC32PLUS = 18 , EM_960 = 19 , EM_PPC = 20 , EM_V800 = 36 ,
  EM_FR20 = 37 , EM_RH32 = 38 , EM_RCE = 39 , EM_ARM = 40 ,
  EM_ALPHA = 41 , EM_SH = 42 , EM_SPARCV9 = 43 , EM_TRICORE = 44 ,
  EM_ARC = 45 , EM_H8_300 = 46 , EM_H8_300H = 47 , EM_H8S = 48 ,
  EM_H8_500 = 49 , EM_IA_64 = 50 , EM_MIPS_X = 51 , EM_COLDFIRE = 52 ,
  EM_68HC12 = 53 , EM_X86_64 = 62 , EM_PDSP = 63 , EM_FX66 = 66 ,
  EM_ST9PLUS = 67 , EM_ST7 = 68 , EM_68HC16 = 69 , EM_68HC11 = 70 ,
  EM_68HC08 = 71 , EM_68HC05 = 72 , EM_SVX = 73 , EM_ST19 = 74 ,
  EM_VAX = 75 , EM_CRIS = 76 , EM_JAVELIN = 77 , EM_FIREPATH = 78 ,
  EM_ZSP = 79 , EM_MMIX = 80 , EM_HUANY = 81 , EM_PRISM = 82 ,
  EM_AVR = 83 , EM_FR30 = 84 , EM_D10V = 85 , EM_D30V = 86 ,
  EM_V850 = 87 , EM_M32R = 88 , EM_MN10300 = 89 , EM_MN10200 = 90 ,
  EM_PJ = 91 , EM_OPENRISC = 92 , EM_ARC_A5 = 93 , EM_XTENSA = 94 ,
  EM_ALTERA_NIOS2 = 113 , EM_AARCH64 = 183 , EM_TILEPRO = 188 , EM_MICROBLAZE = 189 ,
  EM_TILEGX = 191 , **EM_NUM** = 192 }

  Required architecture.
- enum Elf_EVs { EV_NONE = 0 , EV_CURRENT = 1 }

  Object file version.
- enum Elf_EIs {
  EI_MAG0 = 0 , EI_MAG1 = 1 , EI_MAG2 = 2 , EI_MAG3 = 3 ,
  EI_CLASS = 4 , EI_DATA = 5 , EI_VERSION = 6 , EI_OSABI = 7 ,
  EI_ABIVERSION = 8 , EI_PAD = 9 }

  Identification Indices.
- enum Elf_MAGs { ELFMAG0 = 0x7f , ELFMAG1 = 'E' , ELFMAG2 = 'L' , ELFMAG3 = 'F' }

  Magic number.
- enum Elf_CIASSs { ELFCLASSNONE = 0 , ELFCLASS32 = 1 , ELFCLASS64 = 2 , ELFCLASSNUM = 3 }

  File class or capacity.
- enum Elf_DATAs { ELFDATANONE = 0 , ELFDATA2LSB = 1 , ELFDATA2MSB = 2 , ELFDATANUM = 3 }

  Data encoding.
- enum Elf_OSABIs {
  ELFOSABI_NONE = 0 , ELFOSABI_SYSV = 0 , ELFOSABI_HPUX = 1 , ELFOSABI_NETBSD = 2 ,
  ELFOSABI_LINUX = 3 , ELFOSABI_SOLARIS = 6 , ELFOSABI_AIX = 7 , ELFOSABI_IRIX = 8 ,
  ELFOSABI_FREEBSD = 9 , ELFOSABI_TRU64 = 10 , ELFOSABI_MODESTO = 11 , ELFOSABI_OPENBSD
  = 12 ,
  ELFOSABI_ARM = 97 , ELFOSABI_STANDALONE = 255 }

  Identify operating system and ABI to which the object is targeted.
- enum Elf_SHNs {
  SHN_UNDEF = 0 , SHN_LORESERVE = 0xff00 , SHN_LOPROC = 0xff00 , SHN_HIPROC = 0xff1f ,
  SHN_ABS = 0xfff1 , SHN_COMMON = 0xfff2 , SHN_HIRESERVE = 0xffff }

  Special section indexes.
- enum Elf_SHTs {
  SHT_NULL = 0 , SHT_PROGBITS = 1 , SHT_SYMTAB = 2 , SHT_STRTAB = 3 ,
  SHT_RELA = 4 , SHT_HASH = 5 , SHT_DYNAMIC = 6 , SHT_NOTE = 7 ,
  SHT_NOBITS = 8 , SHT_REL = 9 , SHT_SHLIB = 10 , SHT_DYNSYM = 11 ,

SHT_INIT_ARRAY = 14 , SHT_FINI_ARRAY = 15 , SHT_PREINIT_ARRAY = 16 , SHT_GROUP = 17 , SHT_SYMTAB_SHNDX = 18 , SHT_NUM = 19 , SHT_LOOS = 0x60000000 , SHT_HIOS = 0x6fffffff , SHT_LOPROC = 0x70000000 , SHT_HIPROC = 0x7fffffff , SHT_LOUSER = 0x80000000 , SHT_HIUSER = 0xffffffff }

Section type.

- enum Elf_SHFs {
  SHF_WRITE = 0x1 , SHF_ALLOC = 0x2 , SHF_EXECINSTR = 0x4 , SHF_MERGE = 0x10 , SHF_STRINGS = 0x20 , SHF_INFO_LINK = 0x40 , SHF_LINK_ORDER = 0x80 , SHF_OS_NONCONFORMING = 0x100 ,
  SHF_GROUP = 0x200 , SHF_TLS = 0x400 , SHF_MASKOS = 0x0ff00000 , SHF_MASKPROC = 0xf0000000 }

  Section attribute flags.

- enum Elf_PTs {
  PT_NULL = 0 , PT_LOAD = 1 , PT_DYNAMIC = 2 , PT_INTERP = 3 ,
  PT_NOTE = 4 , PT_SHLIB = 5 , PT_PHDR = 6 , PT_TLS = 7 ,
  PT_NUM = 8 , PT_LOOS = 0x60000000 , PT_HIOS = 0x6fffffff , PT_LOPROC = 0x70000000 ,
  PT_HIPROC = 0x7fffffff , PT_GNU_EH_FRAME = PT_LOOS + 0x474e550 , PT_GNU_STACK = PT_LOOS + 0x474e551 , PT_GNU_RELRO = PT_LOOS + 0x474e552 ,
  PT_L4_STACK = PT_LOOS + 0x12 , PT_L4_KIP = PT_LOOS + 0x13 , PT_L4_AUX = PT_LOOS + 0x14 }

  Segment types.

- enum ELF_PFs {
  PF_X = 0x1 , PF_W = 0x2 , PF_R = 0x4 , PF_MASKOS = 0x0ff00000 ,
  PF_MASKPROC = 0x7fffffff }

  Segment permissions.

- enum Elf_NTs_core {
  NT_PRSTATUS = 1 , NT_FPREGSET = 2 , NT_PRPSINFO = 3 , NT_PRXREG = 4 ,
  NT_TASKSTRUCT = 4 , NT_PLATFORM = 5 , NT_AUXV = 6 , NT_GWINDOWS = 7 ,
  NT_ASRS = 8 , NT_PSTATUS = 10 , NT_PSINFO = 13 , NT_PRCRED = 14 ,
  NT_UTSNAME = 15 , NT_LWPSTATUS = 16 , NT_LWPSINFO = 17 , NT_PRFPXREG = 20 }

  Legal values for note segment descriptor types for core files.

- enum Elf_NTs_obj { NT_VERSION = 1 }

  Legal values for the note segment descriptor types for object files.

- enum Elf_DTs {
  DT_NULL = 0 , DT_NEEDED = 1 , DT_PLTRELSZ = 2 , DT_PLTGOT = 3 ,
  DT_HASH = 4 , DT_STRTAB = 5 , DT_SYMTAB = 6 , DT_RELA = 7 ,
  DT_RELASZ = 8 , DT_RELAENT = 9 , DT_STRSZ = 10 , DT_SYMENT = 11 ,
  DT_INIT = 12 , DT_FINI = 13 , DT_SONAME = 14 , DT_RPATH = 15 ,
  DT_SYMBOLIC = 16 , DT_REL = 17 , DT_RELSZ = 18 , DT_RELENT = 19 ,
  DT_PTRREL = 20 , DT_DEBUG = 21 , DT_TEXTREL = 22 , DT_JMPREL = 23 ,
  DT_BIND_NOW = 24 , DT_INIT_ARRAY = 25 , DT_FINI_ARRAY = 26 , DT_INIT_ARRAYSZ = 27 ,
  DT_FINI_ARRAYSZ = 28 , DT_RUNPATH = 29 , DT_FLAGS = 30 , DT_ENCODING = 32 ,
  DT_PREINIT_ARRAY = 32 , DT_PREINIT_ARRAYSZ = 33 , DT_NUM = 34 , DT_LOOS = 0x6000000d ,
  DT_HIOS = 0x6ffff000 , DT_LOPROC = 0x70000000 , DT_HIPROC = 0x7fffffff }

  Dynamic Array Tags.

- enum Elf_DFs {
  DF_ORIGIN = 0x00000001 , DF_SYMBOLIC = 0x00000002 , DF_TEXTREL = 0x00000004 , DF_BIND_NOW = 0x00000008 ,
  DF_STATIC_TLS = 0x00000010 }

  Values of Elf32_Dyn.d_un.d_val, Elf64_Dyn.d_un.d_val in the DT_FLAGS entry.

- enum Elf_DF_1s {
  DF_1_NOW = 0x00000001 , DF_1_GLOBAL = 0x00000002 , DF_1_GROUP = 0x00000004 , DF_1_NODELETE = 0x00000008 ,
  DF_1_LOADFLTR = 0x00000010 , DF_1_INITFIRST = 0x00000020 , DF_1_NOOPEN = 0x00000040 , DF_1_ORIGIN = 0x00000080 ,
  DF_1_DIRECT = 0x00000100 , **DF_1_TRANS** = 0x00000200 , DF_1_INTERPOSE = 0x00000400 , DF_1_NODEFLIB = 0x00000800 ,

DF_1_NODUMP = 0x00001000 , DF_1_CONFALT = 0x00002000 , DF_1_ENDFILTEE = 0x00004000 , DF_1_DISPRELDNE = 0x00008000 , DF_1_DISPRELPND = 0x00010000 }

State flags selectable in the Elf32_Dyn.d_un.d_val / Elf64_Dyn.d_un.d_val element of the DT_FLAGS_1 entry in the dynamic section.

- enum Elf_DTF_1s

  Flags for the feature selection in DT_FEATURE_1.

- enum Elf_DF_P1s { DF_P1_LAZYLOAD = 0x00000001 , DF_P1_GROUPPERM = 0x00000002 }

  Flags in the DT_POSFLAG_1 entry effecting only the next DT_∗ entry.

- enum Elf_R_386_s {
  R_386_NONE = 0 , R_386_32 = 1 , R_386_PC32 = 2 , R_386_GOT32 = 3 , R_386_PLT32 = 4 , R_386_COPY = 5 , R_386_GLOB_DAT = 6 , R_386_JMP_SLOT = 7 , R_386_RELATIVE = 8 , R_386_GOTOFF = 9 , R_386_GOTPC = 10 , **R_386_32PLT** = 11 , R_386_TLS_TPOFF = 14 , R_386_TLS_IE = 15 , R_386_TLS_GOTIE = 16 , R_386_TLS_LE = 17 , R_386_TLS_GD = 18 , R_386_TLS_LDM = 19 , **R_386_16** = 20 , **R_386_PC16** = 21 , **R_386_8** = 22 , **R_386_PC8** = 23 , R_386_TLS_GD_32 = 24 , R_386_TLS_GD_PUSH = 25 , R_386_TLS_GD_CALL = 26 , R_386_TLS_GD_POP = 27 , R_386_TLS_LDM_32 = 28 , R_386_TLS_LDM_PUSH = 29 , R_386_TLS_LDM_CALL = 30 , R_386_TLS_LDM_POP = 31 , R_386_TLS_LDO_32 = 32 , R_386_TLS_IE_32 = 33 , R_386_TLS_LE_32 = 34 , R_386_TLS_DTPMOD32 = 35 , R_386_TLS_DTPOFF32 = 36 , R_386_TLS_TPOFF32 = 37 , R_386_NUM = 38 }

  Relocation types (processor specific).

- enum Elf_EF_ARM_s { }

  ARM specific declarations.

- enum Elf_STT_ARM_s

  Additional symbol types for Thumb.

- enum Elf_SHF_s_ARM { SHF_ARM_ENTRYSECT = 0x10000000 , SHF_ARM_COMDEF = 0x80000000 }

  ARM-specific values for Elf32_Shdr.sh_flags / Elf64_Shdr.sh_flags.

- enum Elf_ARM_SBs { PF_ARM_SB = 0x10000000 }

  ARM-specific program header flags.

- enum Elf_R_ARM_s {
  R_ARM_NONE = 0 , R_ARM_PC24 = 1 , R_ARM_ABS32 = 2 , R_ARM_REL32 = 3 , **R_ARM_PC13** = 4 , R_ARM_ABS16 = 5 , R_ARM_ABS12 = 6 , **R_ARM_THM_ABS5** = 7 , R_ARM_ABS8 = 8 , **R_ARM_SBREL32** = 9 , **R_ARM_THM_PC22** = 10 , **R_ARM_THM_PC8** = 11 , **R_ARM_AMP_VCALL9** = 12 , **R_ARM_SWI24** = 13 , **R_ARM_THM_SWI8** = 14 , **R_ARM_XPC25** = 15 , **R_ARM_THM_XPC22** = 16 , R_ARM_COPY = 20 , R_ARM_GLOB_DAT = 21 , R_ARM_JUMP_SLOT = 22 , R_ARM_RELATIVE = 23 , R_ARM_GOTOFF = 24 , R_ARM_GOTPC = 25 , R_ARM_GOT32 = 26 , R_ARM_PLT32 = 27 , **R_ARM_ALU_PCREL_7_0** = 32 , **R_ARM_ALU_PCREL_15_8** = 33 , **R_ARM_↩ALU_PCREL_23_15** = 34 , **R_ARM_LDR_SBREL_11_0** = 35 , **R_ARM_ALU_SBREL_19_12** = 36 , **R_ARM_ALU_SBREL_27_20** = 37 , **R_ARM_GNU_VTENTRY** = 100 , **R_ARM_GNU_VTINHERIT** = 101 , R_ARM_THM_PC11 = 102 , R_ARM_THM_PC9 = 103 , **R_ARM_↩RXPC25** = 249 , **R_ARM_RSBREL32** = 250 , **R_ARM_THM_RPC22** = 251 , **R_ARM_RREL32** = 252 , **R_ARM_RABS22** = 253 , **R_ARM_RPC24** = 254 , **R_ARM_RBASE** = 255 , R_ARM_NUM = 256 }

  ARM relocations.

- enum Elf_R_AARCH64_s { R_AARCH64_NONE = 0 , **R_AARCH64_RELATIVE** = 1027 }

  AARCH64 relocations.

- enum Elf_R_X86_64_s {
  R_X86_64_NONE = 0 , R_X86_64_64 = 1 , R_X86_64_PC32 = 2 , R_X86_64_GOT32 = 3 , R_X86_64_PLT32 = 4 , R_X86_64_COPY = 5 , R_X86_64_GLOB_DAT = 6 , R_X86_64_JUMP_SLOT = 7 , R_X86_64_RELATIVE = 8 , R_X86_64_GOTPCREL = 9 , R_X86_64_32 = 10 , R_X86_64_32S = 11 ,

R_X86_64_16 = 12 , R_X86_64_PC16 = 13 , R_X86_64_8 = 14 , R_X86_64_PC8 = 15 ,
R_X86_64_DTPMOD64 = 16 , R_X86_64_DTPOFF64 = 17 , R_X86_64_TPOFF64 = 18 , R_X86_64_TLSGD = 19 ,
R_X86_64_TLSLD = 20 , R_X86_64_DTPOFF32 = 21 , R_X86_64_GOTTPOFF = 22 , R_X86_64_TPOFF32 = 23 ,
R_X86_64_NUM = 24 }

AMD x86-64 relocations.

- enum Elf_STNs

  Symbol Table Entry.
- enum Elf_STBs {
  STB_LOCAL = 0 , STB_GLOBAL = 1 , STB_WEAK = 2 , STB_LOOS = 10 ,
  STB_HIOS = 12 , STB_LOPROC = 13 , STB_HIPROC = 15 }

  Symbol Binding.
- enum Elf_STTs {
  STT_NOTYPE = 0 , STT_OBJECT = 1 , STT_FUNC = 2 , STT_SECTION = 3 ,
  STT_FILE = 4 , STT_LOOS = 10 , STT_HIOS = 12 , STT_LOPROC = 13 ,
  STT_HIPROC = 15 }

  Symbol Types.
- enum Elf_ATs {
  AT_NULL = 0 , AT_IGNORE = 1 , AT_EXECFD = 2 , AT_PHDR = 3 ,
  AT_PHENT = 4 , AT_PHNUM = 5 , AT_PAGESZ = 6 , AT_BASE = 7 ,
  AT_FLAGS = 8 , AT_ENTRY = 9 , AT_NOTELF = 10 , AT_UID = 11 ,
  AT_EUID = 12 , AT_GID = 13 , AT_EGID = 14 , AT_L4_AUX = 0xf0 ,
  AT_L4_ENV = 0xf1 }

  Legal values for Elf32_Auxv.atype / Elf64_Auxv.atype.

ELF types

- typedef l4_uint32_t Elf32_Addr

  size 4 align 4
- typedef l4_uint32_t Elf32_Off

  size 4 align 4
- typedef l4_uint16_t Elf32_Half

  size 2 align 2
- typedef l4_uint32_t Elf32_Word

  size 4 align 4
- typedef l4_int32_t Elf32_Sword

  size 4 align 4
- typedef l4_uint64_t Elf64_Addr

  size 8 align 8
- typedef l4_uint64_t Elf64_Off

  size 8 align 8
- typedef l4_uint16_t Elf64_Half

  size 2 align 2
- typedef l4_uint32_t Elf64_Word

  size 4 align 4
- typedef l4_int32_t Elf64_Sword

  size 4 align 4
- typedef l4_uint64_t Elf64_Xword

  size 8 align 8
- typedef l4_int64_t Elf64_Sxword

  size 8 align 8

13.26.1 Detailed Description

Functions and types related to ELF binaries.

13.26.2 Macro Definition Documentation

13.26.2.1 ELF32_R_TYPE

```
#define ELF32_R_TYPE(
              i ) ((unsigned char)(i))
```

See also

Elf_R_386s.

Definition at line 659 of file elf.h.

13.26.2.2 ELF32_ST_BIND

```
#define ELF32_ST_BIND(
              i ) ((i)>>4)
```

See also

Elf_STBs.

Definition at line 889 of file elf.h.

13.26.2.3 ELF32_ST_TYPE

```
#define ELF32_ST_TYPE(
              i ) ((i)&0xf)
```

See also

Elf_STTs.

Definition at line 892 of file elf.h.

13.26.2.4 ELF64_R_TYPE

```
#define ELF64_R_TYPE(
              i ) ((i)&0xffffffffL)
```

See also

Elf_R_386s.

Definition at line 667 of file elf.h.

13.26.2.5 ELF64_ST_BIND

```
#define ELF64_ST_BIND(
              i ) ((i)>>4)
```

See also

Elf_STBs

Definition at line 898 of file elf.h.

13.26.2.6 ELF64_ST_TYPE

```
#define ELF64_ST_TYPE(
              i ) ((i)&0xf)
```

See also

Elf_STTs

Definition at line 901 of file elf.h.

13.26.3 Enumeration Type Documentation

13.26.3.1 anonymous enum

```
anonymous enum
```

Enumerator

EI_NIDENT	Number of characters.

Definition at line 113 of file elf.h.

13.26.3.2 Elf_ARM_SBs

`enum Elf_ARM_SBs`

ARM-specific program header flags.

Enumerator

PF_ARM_SB	Segment contains the location addressed by the static base.

Definition at line 766 of file elf.h.

13.26.3.3 Elf_ATs

`enum Elf_ATs`

Legal values for Elf32_Auxv.atype / Elf64_Auxv.atype.

Enumerator

AT_NULL	End of vector.
AT_IGNORE	Entry should be ignored.
AT_EXECFD	File descriptor of program.
AT_PHDR	Program headers for program.
AT_PHENT	Size of program header entry.
AT_PHNUM	Number of program headers.
AT_PAGESZ	System page size.
AT_BASE	Base address of interpreter.
AT_FLAGS	Flags.
AT_ENTRY	Entry point of program.
AT_NOTELF	Program is not ELF.
AT_UID	Real UID.
AT_EUID	Effective UID.
AT_GID	Real GID.
AT_EGID	Effective GID.
AT_L4_AUX	L4Re AUX section.
AT_L4_ENV	L4Re ENV section.

Definition at line 935 of file elf.h.

13.26.3.4 Elf_ClASSs

`enum Elf_ClASSs`

File class or capacity.

Enumerator

ELFCLASSNONE	Invalid class.
ELFCLASS32	32-bit object
ELFCLASS64	64-bit object
ELFCLASSNUM	Mask for 32-bit or 64-bit class.

Definition at line 293 of file elf.h.

13.26.3.5 Elf_DATAs

`enum Elf_DATAs`

Data encoding.

Enumerator

ELFDATANONE	invalid data encoding
ELFDATA2LSB	0x01020304 => [0x04|0x03|0x02|0x01]
ELFDATA2MSB	0x01020304 => [0x01|0x02|0x03|0x04]
ELFDATANUM	Mask for valid data encoding.

Definition at line 302 of file elf.h.

13.26.3.6 Elf_DF_1s

`enum Elf_DF_1s`

State flags selectable in the Elf32_Dyn.d_un.d_val / Elf64_Dyn.d_un.d_val element of the DT_FLAGS_1 entry in the dynamic section.

Enumerator

DF_1_NOW	Set RTLD_NOW for this object.

Enumerator

DF_1_GLOBAL	Set RTLD_GLOBAL for this object.
DF_1_GROUP	Set RTLD_GROUP for this object.
DF_1_NODELETE	Set RTLD_NODELETE for this object.
DF_1_LOADFLTR	Trigger filtee loading at runtime.
DF_1_INITFIRST	Set RTLD_INITFIRST for this object.
DF_1_NOOPEN	Set RTLD_NOOPEN for this object.
DF_1_ORIGIN	$ORIGIN must be handled.
DF_1_DIRECT	Direct binding enabled.
DF_1_INTERPOSE	Object is used to interpose.
DF_1_NODEFLIB	Ignore default lib search path.
DF_1_NODUMP	Object can't be dldump'ed.
DF_1_CONFALT	Configuration alternative created.
DF_1_ENDFILTEE	Filtee terminates filters search.
DF_1_DISPRELDNE	Disp reloc applied at build time.
DF_1_DISPRELPND	Disp reloc applied at run-time.

Definition at line 590 of file elf.h.

13.26.3.7 Elf_DF_P1s

`enum Elf_DF_P1s`

Flags in the DT_POSFLAG_1 entry effecting only the next DT_∗ entry.

Enumerator

DF_P1_LAZYLOAD	Lazyload following object.
DF_P1_GROUPPERM	Symbols from next object are not generally available.

Definition at line 619 of file elf.h.

13.26.3.8 Elf_DFs

`enum Elf_DFs`

Values of Elf32_Dyn.d_un.d_val, Elf64_Dyn.d_un.d_val in the DT_FLAGS entry.

Enumerator

DF_ORIGIN	Object may use DF_ORIGIN.
DF_SYMBOLIC	Symbol resolutions starts here.
DF_TEXTREL	Object contains text relocations.
DF_BIND_NOW	No lazy binding for this object.
DF_STATIC_TLS	Module uses the static TLS model.

Definition at line 577 of file elf.h.

13.26.3.9 Elf_DTs

`enum Elf_DTs`

Dynamic Array Tags.

See also

Elf32_Dyn.d_tag, Elf64_Dyn.d_tag.

Enumerator

DT_NULL	end of _DYNAMIC array
DT_NEEDED	name of a needed library
DT_PLTRELSZ	total size of relocation entry
DT_PLTGOT	address assoc with prog link table
DT_HASH	address of symbol hash table
DT_STRTAB	address of string table
DT_SYMTAB	address of symbol table
DT_RELA	address of relocation table
DT_RELASZ	total size of relocation table
DT_RELAENT	size of DT_RELA relocation entry
DT_STRSZ	size of the string table
DT_SYMENT	size of a symbol table entry
DT_INIT	address of initialization function
DT_FINI	address of termination function
DT_SONAME	name of the shared object
DT_RPATH	search library path
DT_SYMBOLIC	alter symbol resolution algorithm
DT_REL	address of relocation table
DT_RELSZ	total size of DT_REL relocation table
DT_RELENT	size of the DT_REL relocation entry
DT_PTRREL	type of relocation entry
DT_DEBUG	for debugging purposes
DT_TEXTREL	at least on entry changes r/o section
DT_JMPREL	address of relocation entries
DT_BIND_NOW	Process relocations of object.

Enumerator

DT_INIT_ARRAY	Array with addresses of init fct.
DT_FINI_ARRAY	Array with addresses of fini fct.
DT_INIT_ARRAYSZ	Size in bytes of DT_INIT_ARRAY.
DT_FINI_ARRAYSZ	Size in bytes of DT_FINI_ARRAY.
DT_RUNPATH	Library search path.
DT_FLAGS	Flags for the object being loaded.
DT_ENCODING	Start of encoded range.
DT_PREINIT_ARRAY	Array with addresses of preinit fct.
DT_PREINIT_ARRAYSZ	size in bytes of DT_PREINIT_ARRAY
DT_NUM	Number used.
DT_LOOS	Start of OS-specific.
DT_HIOS	End of OS-specific.
DT_LOPROC	processor-specific
DT_HIPROC	processor-specific

Definition at line 531 of file elf.h.

13.26.3.10 Elf_EF_ARM_s

`enum Elf_EF_ARM_s`

ARM specific declarations.

Processor specific flags for the ELF header e_flags field.

Enumerator

EF_ARM_ALIGN8	8-bit structure alignment is in use

Definition at line 726 of file elf.h.

13.26.3.11 Elf_EIs

`enum Elf_EIs`

Identification Indices.

See also

Elf32_Ehdr.e_ident, Elf64_Ehdr.e_ident

Enumerator

EI_MAG0	file id 0
EI_MAG1	file id 1
EI_MAG2	file id 2
EI_MAG3	file id 3
EI_CLASS	file class
EI_DATA	data encoding
EI_VERSION	file version
EI_OSABI	Operating system / ABI identification.
EI_ABIVERSION	ABI version.
EI_PAD	start of padding bytes

Definition at line 269 of file elf.h.

13.26.3.12 Elf_EMs

`enum Elf_EMs`

Required architecture.

See also

Elf32_Ehdr.e_machine, Elf64_Ehdr.e_machine

Enumerator

EM_NONE	no machine
EM_M32	AT&T WE 32100.
EM_SPARC	SPARC.
EM_386	Intel 80386.
EM_68K	Motorola 68000.
EM_88K	Motorola 88000.
EM_860	Intel 80860.
EM_MIPS	MIPS RS3000 big-endian.
EM_MIPS_RS4_BE	MIPS RS4000 big-endian.
EM_SPARC64	SPARC 64-bit.
EM_PARISC	HP PA-RISC.
EM_VPP500	Fujitsu VPP500.
EM_SPARC32PLUS	Sun's V8plus.
EM_960	Intel 80960.
EM_PPC	PowerPC.
EM_V800	NEC V800.
EM_FR20	Fujitsu FR20.
EM_RH32	TRW RH-32.
EM_RCE	Motorola RCE.
EM_ARM	Advanced RISC Machines ARM.
EM_ALPHA	Digital Alpha.

Enumerator

EM_SH	Hitachi SuperH.
EM_SPARCV9	SPARC v9 64-bit.
EM_TRICORE	Siemens Tricore embedded processor.
EM_ARC	Argonaut RISC Core, Argonaut Techn Inc.
EM_H8_300	Hitachi H8/300.
EM_H8_300H	Hitachi H8/300H.
EM_H8S	Hitachi H8/S.
EM_H8_500	Hitachi H8/500.
EM_IA_64	HP/Intel IA-64.
EM_MIPS_X	Stanford MIPS-X.
EM_COLDFIRE	Motorola Coldfire.
EM_68HC12	Motorola M68HC12.
EM_X86_64	Advanced Micro Devices x86-64.
EM_PDSP	Sony DSP Processor.
EM_FX66	Siemens FX66 microcontroller.
EM_ST9PLUS	STMicroelectronics ST9+ 8/16 mc.
EM_ST7	STmicroelectronics ST7 8 bit mc.
EM_68HC16	Motorola MC68HC16 microcontroller.
EM_68HC11	Motorola MC68HC11 microcontroller.
EM_68HC08	Motorola MC68HC08 microcontroller.
EM_68HC05	Motorola MC68HC05 microcontroller.
EM_SVX	Silicon Graphics SVx.
EM_ST19	STMicroelectronics ST19 8 bit mc.
EM_VAX	Digital VAX.
EM_CRIS	Axis Communications 32-bit embedded processor.
EM_JAVELIN	Infineon Technologies 32-bit embedded processor.
EM_FIREPATH	Element 14 64-bit DSP Processor.
EM_ZSP	LSI Logic 16-bit DSP Processor.
EM_MMIX	Donald Knuth's educational 64-bit processor.
EM_HUANY	Harvard University machine-independent object files.
EM_PRISM	SiTera Prism.
EM_AVR	Atmel AVR 8-bit microcontroller.
EM_FR30	Fujitsu FR30.
EM_D10V	Mitsubishi D10V.
EM_D30V	Mitsubishi D30V.
EM_V850	NEC v850.
EM_M32R	Mitsubishi M32R.
EM_MN10300	Matsushita MN10300.
EM_MN10200	Matsushita MN10200.
EM_PJ	picoJava
EM_OPENRISC	OpenRISC 32-bit embedded processor.
EM_ARC_A5	ARC Cores Tangent-A5.
EM_XTENSA	Tensilica Xtensa Architecture.
EM_ALTERA_NIOS2	Altera Nios II.
EM_AARCH64	ARM AARCH64.
EM_TILEPRO	Tilera TILEPro.
EM_MICROBLAZE	Xilinx MicroBlaze.
EM_TILEGX	Tilera TILE-Gx.

Definition at line 179 of file elf.h.

13.26.3.13 Elf_ETs

`enum Elf_ETs`

Object file type.

See also

Elf32_Ehdr.e_type, Elf64_Ehdr.e_type

Enumerator

ET_NONE	no file type
ET_REL	relocatable file
ET_EXEC	executable file
ET_DYN	shared object file
ET_CORE	core file
ET_LOPROC	processor-specific
ET_HIPROC	processor-specific

Definition at line 164 of file elf.h.

13.26.3.14 Elf_EVs

`enum Elf_EVs`

Object file version.

See also

Elf32_Ehdr.e_version, Elf64_Ehdr.e_version

Enumerator

EV_NONE	Invalid version.
EV_CURRENT	Current version.

Definition at line 261 of file elf.h.

13.26.3.15 Elf_MAGs

`enum Elf_MAGs`

Magic number.

Enumerator

ELFMAG0	e_ident[EI_MAG0]
ELFMAG1	e_ident[EI_MAG1]
ELFMAG2	e_ident[EI_MAG2]
ELFMAG3	e_ident[EI_MAG3]

Definition at line 284 of file elf.h.

13.26.3.16 Elf_NTs_core

`enum Elf_NTs_core`

Legal values for note segment descriptor types for core files.

Enumerator

NT_PRSTATUS	Contains copy of prstatus struct.
NT_FPREGSET	Contains copy of fpregset struct.
NT_PRPSINFO	Contains copy of prpsinfo struct.
NT_PRXREG	Contains copy of prxregset struct.
NT_TASKSTRUCT	Contains copy of task structure.
NT_PLATFORM	String from sysinfo(SI_PLATFORM)
NT_AUXV	Contains copy of auxv array.
NT_GWINDOWS	Contains copy of gwindows struct.
NT_ASRS	Contains copy of asrset struct.
NT_PSTATUS	Contains copy of pstatus struct.
NT_PSINFO	Contains copy of psinfo struct.
NT_PRCRED	Contains copy of prcred struct.
NT_UTSNAME	Contains copy of utsname struct.
NT_LWPSTATUS	Contains copy of lwpstatus struct.
NT_LWPSINFO	Contains copy of lwpinfo struct.
NT_PRFPXREG	Contains copy of fprxregset struct.

Definition at line 482 of file elf.h.

13.26.3.17 Elf_NTs_obj

`enum Elf_NTs_obj`

Legal values for the note segment descriptor types for object files.

Enumerator

NT_VERSION	Contains a version string.

Definition at line 503 of file elf.h.

13.26.3.18 Elf_OSABIs

`enum Elf_OSABIs`

Identify operating system and ABI to which the object is targeted.

Enumerator

ELFOSABI_NONE	UNIX System V ABI.
ELFOSABI_SYSV	Alias.
ELFOSABI_HPUX	HP-UX.
ELFOSABI_NETBSD	NetBSD.
ELFOSABI_LINUX	Linux.
ELFOSABI_SOLARIS	Sun Solaris.
ELFOSABI_AIX	IBM AIX.
ELFOSABI_IRIX	SGI Irix.
ELFOSABI_FREEBSD	FreeBSD.
ELFOSABI_TRU64	Compaq TRU64 UNIX.
ELFOSABI_MODESTO	Novell Modesto.
ELFOSABI_OPENBSD	OpenBSD.
ELFOSABI_ARM	ARM.
ELFOSABI_STANDALONE	Standalone (embedded) application.

Definition at line 311 of file elf.h.

13.26.3.19 ELF_PFs

`enum ELF_PFs`

Segment permissions.

Enumerator

PF_X	Executable.
PF_W	Write.
PF_R	Read.
PF_MASKOS	OS-specific.
PF_MASKPROC	Processor-specific.

Definition at line 472 of file elf.h.

13.26.3.20 Elf_PTs

`enum Elf_PTs`

Segment types.

Enumerator

PT_NULL	array is unused
PT_LOAD	loadable
PT_DYNAMIC	dynamic linking information
PT_INTERP	path to interpreter
PT_NOTE	auxiliary information
PT_SHLIB	reserved
PT_PHDR	location of the pht itself
PT_TLS	Thread-local storage segment.
PT_NUM	Number of defined types.
PT_LOOS	OS-specific.
PT_HIOS	OS-specific.
PT_LOPROC	processor-specific
PT_HIPROC	processor-specific
PT_GNU_EH_FRAME	EH frame information.
PT_GNU_STACK	Flags for stack.
PT_GNU_RELRO	Read only after reloc.
PT_L4_STACK	Address of the stack.
PT_L4_KIP	Address of the KIP.
PT_L4_AUX	Address of the AUX structures.

Definition at line 446 of file elf.h.

13.26.3.21 Elf_R_386_s

`enum Elf_R_386_s`

Relocation types (processor specific).

Enumerator

R_386_NONE	none
R_386_32	S + A.
R_386_PC32	S + A - P.
R_386_GOT32	G + A - P.
R_386_PLT32	L + A - P.
R_386_COPY	none
R_386_GLOB_DAT	S.
R_386_JMP_SLOT	S.
R_386_RELATIVE	B + A.
R_386_GOTOFF	S + A - GOT.
R_386_GOTPC	GOT + A - P.
R_386_TLS_TPOFF	Offset in static TLS block.
R_386_TLS_IE	Address of GOT entry for static TLS block offset.
R_386_TLS_GOTIE	GOT entry for static TLS block offset.
R_386_TLS_LE	Offset relative to static TLS block.
R_386_TLS_GD	Direct 32 bit for GNU version of general dynamic thread local data.
R_386_TLS_LDM	Direct 32 bit for GNU version of local dynamic thread local data in LE code.
R_386_TLS_GD_32	Direct 32 bit for general dynamic thread local data.
R_386_TLS_GD_PUSH	Tag for pushl in GD TLS code.
R_386_TLS_GD_CALL	Relocation for call to __tls_get_addr()
R_386_TLS_GD_POP	Tag for popl in GD TLS code.
R_386_TLS_LDM_32	Direct 32 bit for local dynamic thread local data in LE code.
R_386_TLS_LDM_PUSH	Tag for pushl in LDM TLS code.
R_386_TLS_LDM_CALL	Relocation for call to __tls_get_addr() in LDM code.
R_386_TLS_LDM_POP	Tag for popl in LDM TLS code.
R_386_TLS_LDO_32	Offset relative to TLS block.
R_386_TLS_IE_32	GOT entry for negated static TLS block offset.
R_386_TLS_LE_32	Negated offset relative to static TLS block.
R_386_TLS_DTPMOD32	ID of module containing symbol.
R_386_TLS_DTPOFF32	Offset in TLS block.
R_386_TLS_TPOFF32	Negated offset in static TLS block.
R_386_NUM	Keep this the last entry.

Definition at line 673 of file elf.h.

13.26.3.22 Elf_R_AARCH64_s

`enum Elf_R_AARCH64_s`

AARCH64 relocations.

Enumerator

R_AARCH64_NONE	No reloc.

Definition at line 821 of file elf.h.

13.26.3.23 Elf_R_ARM_s

`enum Elf_R_ARM_s`

ARM relocations.

Enumerator

R_ARM_NONE	No reloc.
R_ARM_PC24	PC relative 26 bit branch.
R_ARM_ABS32	Direct 32 bit
R_ARM_REL32	PC relative 32 bit.
R_ARM_ABS16	Direct 16 bit.
R_ARM_ABS12	Direct 12 bit.
R_ARM_ABS8	Direct 8 bit.
R_ARM_COPY	Copy symbol at runtime.
R_ARM_GLOB_DAT	Create GOT entry.
R_ARM_JUMP_SLOT	Create PLT entry.
R_ARM_RELATIVE	Adjust by program base.
R_ARM_GOTOFF	32 bit offset to GOT
R_ARM_GOTPC	32 bit PC relative offset to GOT
R_ARM_GOT32	32 bit GOT entry
R_ARM_PLT32	32 bit PLT address
R_ARM_THM_PC11	thumb unconditional branch
R_ARM_THM_PC9	thumb conditional branch
R_ARM_NUM	Keep this the last entry.

Definition at line 773 of file elf.h.

13.26.3.24 Elf_R_X86_64_s

`enum Elf_R_X86_64_s`

AMD x86-64 relocations.

Enumerator

R_X86_64_NONE	No reloc.
R_X86_64_64	Direct 64 bit
R_X86_64_PC32	PC relative 32 bit signed.
R_X86_64_GOT32	32 bit GOT entry
R_X86_64_PLT32	32 bit PLT address

Enumerator

R_X86_64_COPY	Copy symbol at runtime.
R_X86_64_GLOB_DAT	Create GOT entry.
R_X86_64_JUMP_SLOT	Create PLT entry.
R_X86_64_RELATIVE	Adjust by program base.
R_X86_64_GOTPCREL	32 bit signed PC relative offset to GOT
R_X86_64_32	Direct 32 bit zero extended.
R_X86_64_32S	Direct 32 bit sign extended.
R_X86_64_16	Direct 16 bit zero extended.
R_X86_64_PC16	16 bit sign extended pc relative
R_X86_64_8	Direct 8 bit sign extended
R_X86_64_PC8	8 bit sign extended pc relative
R_X86_64_DTPMOD64	ID of module containing symbol.
R_X86_64_DTPOFF64	Offset in module's TLS block.
R_X86_64_TPOFF64	Offset in initial TLS block.
R_X86_64_TLSGD	32 bit signed PC relative offset to two GOT entries for GD symbol
R_X86_64_TLSLD	32 bit signed PC relative offset to two GOT entries for LD symbol
R_X86_64_DTPOFF32	Offset in TLS block.
R_X86_64_GOTTPOFF	32 bit signed PC relative offset to GOT entry for IE symbol
R_X86_64_TPOFF32	Offset in initial TLS block.

Definition at line 828 of file elf.h.

13.26.3.25 Elf_SHF_s_ARM

`enum Elf_SHF_s_ARM`

ARM-specific values for Elf32_Shdr.sh_flags / Elf64_Shdr.sh_flags.

Enumerator

SHF_ARM_ENTRYSECT	Section contains an entry point.
SHF_ARM_COMDEF	Section may be multiply defined in the input to a link step.

Definition at line 758 of file elf.h.

13.26.3.26 Elf_SHFs

`enum Elf_SHFs`

Section attribute flags.

Enumerator

SHF_WRITE	writeable during execution
SHF_ALLOC	section occupies virt memory
SHF_EXECINSTR	code section
SHF_MERGE	Might be merged.
SHF_STRINGS	Contains nul-terminated strings.
SHF_INFO_LINK	'sh_info' contains SHT index
SHF_LINK_ORDER	Preserve order after combining.
SHF_OS_NONCONFORMING	Non-standard OS-specific handling required.
SHF_GROUP	Section is member of a group.
SHF_TLS	Section hold thread-local data.
SHF_MASKOS	OS-specific.
SHF_MASKPROC	processor-specific mask

Definition at line 401 of file elf.h.

13.26.3.27 Elf_SHNs

`enum Elf_SHNs`

Special section indexes.

Enumerator

SHN_UNDEF	undefined section header entry
SHN_LORESERVE	lower bound of reserved indexes
SHN_LOPROC	lower bound of proc spec entr
SHN_HIPROC	upper bound of proc spec entr
SHN_ABS	absolute values for ref
SHN_COMMON	common symbols
SHN_HIRESERVE	upper bound of reserved indexes

Definition at line 330 of file elf.h.

13.26.3.28 Elf_SHTs

`enum Elf_SHTs`

Section type.

Enumerator

SHT_NULL	inactive section header
SHT_PROGBITS	information defined by program
SHT_SYMTAB	symbol table
SHT_STRTAB	string table
SHT_RELA	reloc entries w/ explicit addens
SHT_HASH	symbol hash table
SHT_DYNAMIC	information for dynamic linking
SHT_NOTE	information that marks the file
SHT_NOBITS	occupies no space in the file
SHT_REL	reloc entries w/o explicit addens
SHT_SHLIB	reserved + unspecified semantics
SHT_DYNSYM	symbol table (dynamic
SHT_INIT_ARRAY	Array of constructors.
SHT_FINI_ARRAY	Array of destructors.
SHT_PREINIT_ARRAY	Array of pre-constructors.
SHT_GROUP	Section group.
SHT_SYMTAB_SHNDX	Extended section indices.
SHT_NUM	Number of defined types.
SHT_LOOS	Start OS-specific.
SHT_HIOS	End OS-specific.
SHT_LOPROC	Start processor-specific.
SHT_HIPROC	End processor-specific.
SHT_LOUSER	Start application-specific.
SHT_HIUSER	End application-specific.

Definition at line 372 of file elf.h.

13.26.3.29 Elf_STBs

```
enum Elf_STBs
```

Symbol Binding.

See also

ELF32_ST_BIND, ELF64_ST_BIND

Enumerator

STB_LOCAL	not visible outside object file
STB_GLOBAL	visible to all objects being combined
STB_WEAK	resemble global symbols
STB_LOOS	OS-specific.
STB_HIOS	OS-specific.
STB_LOPROC	Processor-specific.
STB_HIPROC	Processor-specific.

Definition at line 908 of file elf.h.

13.26.3.30 Elf_STTs

`enum Elf_STTs`

Symbol Types.

See also

ELF32_ST_TYPE, ELF64_ST_TYPE

Enumerator

STT_NOTYPE	symbol's type not specified
STT_OBJECT	associated with a data object
STT_FUNC	associated with a function or other code
STT_SECTION	associated with a section
STT_FILE	source file name associated with object
STT_LOOS	OS-specific.
STT_HIOS	OS-specific.
STT_LOPROC	processor-specific
STT_HIPROC	processor-specific

Definition at line 921 of file elf.h.

13.27 Error Handling

Error handling for L4 object invocation.

Collaboration diagram for Error Handling:

Enumerations

- enum l4_ipc_tcr_error_t {
  L4_IPC_ERROR_MASK = 0x1F , L4_IPC_SND_ERR_MASK = 0x01 , L4_IPC_ENOT_EXISTENT = 0x04 ,

L4_IPC_RETIMEOUT = 0x03 ,
L4_IPC_SETIMEOUT = 0x02 , L4_IPC_RECANCELED = 0x07 , L4_IPC_SECANCELED = 0x06 ,
L4_IPC_REMAPFAILED = 0x11 ,
L4_IPC_SEMAPFAILED = 0x10 , L4_IPC_RESNDPFTO = 0x0b , L4_IPC_SESNDPFTO = 0x0a ,
L4_IPC_RERCVPFTO = 0x0d ,
L4_IPC_SERCVPFTO = 0x0c , L4_IPC_REABORTED = 0x0f , L4_IPC_SEABORTED = 0x0e ,
L4_IPC_REMSGCUT = 0x09 ,
L4_IPC_SEMSGCUT = 0x08 }

Error codes in the error TCR.

Functions

- l4_umword_t l4_ipc_error (l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW

  Get the error code for an object invocation.
- long l4_error (l4_msgtag_t tag) L4_NOTHROW

  Return error code of a system call return message tag or the tag label.
- int l4_ipc_is_snd_error (l4_utcb_t *utcb) L4_NOTHROW

  Returns whether an error occurred in send phase of an invocation.
- int l4_ipc_is_rcv_error (l4_utcb_t *utcb) L4_NOTHROW

  Returns whether an error occurred in receive phase of an invocation.
- int l4_ipc_error_code (l4_utcb_t *utcb) L4_NOTHROW

  Get the error condition of the last invocation from the TCR.

13.27.1 Detailed Description

Error handling for L4 object invocation.

Include File

```
#include <l4/sys/ipc.h>
```

13.27.2 Enumeration Type Documentation

13.27.2.1 l4_ipc_tcr_error_t

```
enum l4_ipc_tcr_error_t
```

Error codes in the *error* TCR.

The error codes are accessible via the *error* TCR, see l4_thread_regs_t.error.

Enumerator

L4_IPC_ERROR_MASK	Mask for error bits.
L4_IPC_SND_ERR_MASK	Send error mask.
L4_IPC_ENOT_EXISTENT	Non-existing destination or source.
L4_IPC_RETIMEOUT	Timeout during receive operation.

Enumerator

L4_IPC_SETIMEOUT	Timeout during send operation.
L4_IPC_RECANCELED	Receive operation canceled.
L4_IPC_SECANCELED	Send operation canceled.
L4_IPC_REMAPFAILED	Map flexpage failed in receive operation.
L4_IPC_SEMAPFAILED	Map flexpage failed in send operation.
L4_IPC_RESNDPFTO	Send-pagefault timeout in receive operation.
L4_IPC_SESNDPFTO	Send-pagefault timeout in send operation.
L4_IPC_RERCVPFTO	Receive-pagefault timeout in receive operation.
L4_IPC_SERCVPFTO	Receive-pagefault timeout in send operation.
L4_IPC_REABORTED	Receive operation aborted.
L4_IPC_SEABORTED	Send operation aborted.
L4_IPC_REMSGCUT	Cut receive message, due to message buffer is too small.
L4_IPC_SEMSGCUT	Cut send message. due to message buffer is too small,

Definition at line 75 of file ipc.h.

13.27.3 Function Documentation

13.27.3.1 l4_error()

```
long l4_error (
            l4_msgtag_t tag )  [inline]
```

Return error code of a system call return message tag or the tag label.

Parameters

tag	System call return message type.

Returns

In case of IPC error a negative error code in the range of L4_EIPC_LO to L4_EIPC_HI, otherwise the tag label.

Examples

examples/libs/l4re/streammap/client.cc, examples/sys/aliens/main.c, examples/sys/isr/main.c, examples/sys/migrate/thread_mig
examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 535 of file ipc.h.

Referenced by L4Re::Mem_alloc::alloc(), L4virtio::Driver::Device::bind_notification_irq(), L4::cap_dynamic_cast(), L4virtio::Driver::Device::driver_connect(), fiasco_tbuf_log(), fiasco_tbuf_log_3val(), fiasco_tbuf_log_binary(), L4Re::Util::Event_t< PAYLOAD >::init(), l4_vcon_write_u(), and L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process().

Here is the caller graph for this function:

13.27.3.2 l4_ipc_error()

```
l4_umword_t l4_ipc_error (
            l4_msgtag_t tag,
            l4_utcb_t * utcb )  [inline]
```

Get the error code for an object invocation.

Parameters

tag	Return value of the invocation.
utcb	UTCB that was used for the invocation.

Returns

0 if no error condition is set, error code otherwise (see l4_ipc_tcr_error_t).

Examples

examples/sys/ipc/ipc_example.c, examples/sys/isr/main.c, and examples/sys/start-with-exc/main.c.

Definition at line 518 of file ipc.h.

References l4_msgtag_has_error().

Referenced by l4util_ioport_map(), L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process(), and L4virtio::Driver::Device::wait()

Here is the call graph for this function:

Here is the caller graph for this function:

13.27.3.3 l4_ipc_error_code()

```
int l4_ipc_error_code (
            l4_utcb_t * utcb )  [inline]
```

Get the error condition of the last invocation from the TCR.

Precondition

l4_msgtag_has_error(tag) == true

Parameters

utcb	UTCB to check.

Returns

Error condition of type l4_ipc_tcr_error_t.

Definition at line 547 of file ipc.h.

13.27.3.4 l4_ipc_is_rcv_error()

```
int l4_ipc_is_rcv_error (
            l4_utcb_t * utcb )  [inline]
```

Returns whether an error occurred in receive phase of an invocation.

Precondition

l4_msgtag_has_error(tag) == true

Parameters

utcb	UTCB to check.

Returns

Boolean value.

Definition at line 544 of file ipc.h.

13.27.3.5 l4_ipc_is_snd_error()

```
int l4_ipc_is_snd_error (
            l4_utcb_t * utcb )  [inline]
```

Returns whether an error occurred in send phase of an invocation.

Precondition

l4_msgtag_has_error(tag) == true

Parameters

utcb	UTCB to check.

Returns

Boolean value.

Definition at line 541 of file ipc.h.

13.28 Error codes

Common error codes.

Collaboration diagram for Error codes:

Enumerations

- enum l4_error_code_t {
  L4_EOK = 0 , L4_EPERM = 1 , L4_ENOENT = 2 , L4_EIO = 5 ,
  L4_ENXIO = 6 , L4_E2BIG = 7 , L4_EAGAIN = 11 , L4_ENOMEM = 12 ,
  L4_EACCESS = 13 , L4_EFAULT = 14 , L4_EBUSY = 16 , L4_EEXIST = 17 ,
  L4_ENODEV = 19 , L4_EINVAL = 22 , L4_ENOSPC = 28 , L4_ERANGE = 34 ,
  L4_ENAMETOOLONG = 36 , L4_ENOSYS = 38 , L4_EBADPROTO = 39 , L4_EADDRNOTAVAIL = 99 ,
  L4_ERRNOMAX = 100 , L4_ENOREPLY = 1000 , L4_EMSGTOOSHORT = 1001 , L4_EMSGTOOLONG = 1002 ,
  L4_EMSGMISSARG = 1003 , L4_EIPC_LO = 2000 , L4_EIPC_HI = 2000 + 0x1f }

  L4 error codes.

13.28.1 Detailed Description

Common error codes.

Include File

```
#include <l4/sys/err.h>
```

13.28.2 Enumeration Type Documentation

13.28.2.1 l4_error_code_t

```
enum l4_error_code_t
```

L4 error codes.

Those error codes are used by both the kernel and the user programs.

Enumerator

L4_EOK	Ok.
L4_EPERM	No permission.
L4_ENOENT	No such entity.
L4_EIO	I/O error.
L4_ENXIO	No such device or address.
L4_E2BIG	Argument value too big.
L4_EAGAIN	Try again.
L4_ENOMEM	No memory.
L4_EACCESS	Permission denied.
L4_EFAULT	Invalid memory address.
L4_EBUSY	Object currently busy, try later.
L4_EEXIST	Already exists.
L4_ENODEV	No such thing.
L4_EINVAL	Invalid argument.
L4_ENOSPC	No space left on device.
L4_ERANGE	Range error.
L4_ENAMETOOLONG	Name too long.
L4_ENOSYS	No sys.
L4_EBADPROTO	Unsupported protocol.
L4_EADDRNOTAVAIL	Address not available.
L4_ERRNOMAX	Maximum error value.
L4_ENOREPLY	No reply.
L4_EMSGTOOSHORT	Message too short.
L4_EMSGTOOLONG	Message too long.
L4_EMSGMISSARG	Message has invalid capability.
L4_EIPC_LO	Communication error-range low.
L4_EIPC_HI	Communication error-range high.

Definition at line 41 of file err.h.

13.29 Event API

Event API.

Collaboration diagram for Event API:

Data Structures

- class L4Re::Event

  Event class.
- class L4Re::Event_buffer_t< PAYLOAD >

  Event buffer class.

13.29.1 Detailed Description

Event API.

On top of a shared L4Re::Dataspace (and optionally using L4::Triggerable), the event API implements asynchronous event transmission from an event provider (server) to an event receiver (client). Events are put into an Event_buffer_t residing on the shared L4Re::Dataspace.

This interface is not usually used directly. Instead use L4Re::Util::Event_t for clients. An example server portion is implemented in L4Re::Util::Event_svr.

13.30 Event interface

Event C interface.

Collaboration diagram for Event interface:

Functions

- long l4re_event_get_buffer (const l4_cap_idx_t server, const l4re_ds_t ds) L4_NOTHROW

  Get an event signal buffer.
- long l4re_event_get_num_streams (const l4_cap_idx_t server) L4_NOTHROW

  Get number of streams.
- long l4re_event_get_stream_info (const l4_cap_idx_t server, int idx, l4re_event_stream_info_t *info) L4_NOTHROW

  Get information on a stream.
- long l4re_event_get_stream_info_for_id (const l4_cap_idx_t server, l4_umword_t stream_id, l4re_event_stream_info_t *info) L4_NOTHROW

  Get info for a stream given a stream id.
- long l4re_event_get_axis_info (const l4_cap_idx_t server, l4_umword_t id, unsigned naxes, unsigned const *axis, l4re_event_absinfo_t *info) L4_NOTHROW

  Get Axis information for a stream.

13.30.1 Detailed Description

Event C interface.

13.30.2 Function Documentation

13.30.2.1 l4re_event_get_axis_info()

```
long l4re_event_get_axis_info (
            const l4_cap_idx_t server,
            l4_umword_t id,
            unsigned naxes,
            unsigned const * axis,
            l4re_event_absinfo_t * info )
```

Get Axis information for a stream.

Parameters

	server	Server to talk to.
	id	Id of the stream to get information from.
	naxes	Number of axes in `axis` array.
`in`	*axis*	Array of axis IDs whose information should be retrieved.
`out`	*info*	Information buffer to store the retrieved axis infos.

Return values

0	Success
<0	Error

See also

L4Re::Event::get_axis_info

13.30.2.2 l4re_event_get_buffer()

```
long l4re_event_get_buffer (
            const l4_cap_idx_t server,
            const l4re_ds_t ds )
```

Get an event signal buffer.

Parameters

server	Server to talk to.
ds	Buffer to event data.

Returns

0 for success, <0 on error

See also

L4Re::Event::get_buffer

13.30.2.3 l4re_event_get_num_streams()

```
long l4re_event_get_num_streams (
            const l4_cap_idx_t server )
```

Get number of streams.

Parameters

server	Server to talk to.

Returns

0 for success, <0 on error

See also

L4Re::Event::get_num_streams

13.30.2.4 l4re_event_get_stream_info()

```
long l4re_event_get_stream_info (
            const l4_cap_idx_t server,
            int idx,
            l4re_event_stream_info_t * info )
```

Get information on a stream.

Parameters

server	Server to talk to.
idx	Index value.

Return values

info	Information buffer.

Returns

0 for success, <0 on error

See also

L4Re::Event::get_stream_info

13.30.2.5 l4re_event_get_stream_info_for_id()

```
long l4re_event_get_stream_info_for_id (
            const l4_cap_idx_t server,
            l4_umword_t stream_id,
            l4re_event_stream_info_t * info )
```

Get info for a stream given a stream id.

Parameters

server	Server to talk to.
stream↩ _id	Stream ID.

Return values

info	Information buffer.

Returns

0 for success, <0 on error

See also

L4Re::Event::get_stream_info_for_id

13.31 Exception registers

Overly definition of the MRs for exception messages.

Collaboration diagram for Exception registers:

Functions

- l4_exc_regs_t * l4_utcb_exc (void) L4_NOTHROW L4_PURE

  Get the message-register block of a UTCB (for an exception IPC).
- l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE

  Access function to get the program counter of the exception state.
- void l4_utcb_exc_pc_set (l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW

  Set the program counter register in the exception state.
- unsigned long l4_utcb_exc_typeval (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE

  Get the value out of an exception UTCB that describes the type of exception.
- int l4_utcb_exc_is_pf (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE

  Check whether an exception IPC is a page fault.
- l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE

  Function to get the L4 style page fault address out of an exception.
- int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE

  Check whether an exception IPC was triggered via l4_thread_ex_regs().

13.31.1 Detailed Description

Overly definition of the MRs for exception messages.

13.31.2 Function Documentation

13.31.2.1 l4_utcb_exc()

```
l4_exc_regs_t * l4_utcb_exc (
            void  )  [inline]
```

Get the message-register block of a UTCB (for an exception IPC).

Returns

A pointer to the exception message in `u`.

Examples

examples/sys/aliens/main.c, and examples/sys/singlestep/main.c.

Definition at line 361 of file utcb.h.

13.31.2.2 l4_utcb_exc_is_ex_regs_exception()

```
int l4_utcb_exc_is_ex_regs_exception (
            l4_exc_regs_t const * u )  [inline]
```

Check whether an exception IPC was triggered via l4_thread_ex_regs().

Return values

0	Exception was not triggered through ex_regs.
!=0	Exception was triggered through ex_regs.

This function checks if the exception was emitted by using the L4_THREAD_EX_REGS_TRIGGER_EXCEPTION flag in an l4_thread_ex_regs() call.

Definition at line 116 of file utcb.h.

References l4_utcb_exc_typeval().

Here is the call graph for this function:

13.31.2.3 l4_utcb_exc_is_pf()

```
int l4_utcb_exc_is_pf (
            l4_exc_regs_t const * u )  [inline]
```

Check whether an exception IPC is a page fault.

Returns

0 if not, != 0 if yes

Function to check whether an exception IPC is a page fault, also applies to I/O pagefaults.

Definition at line 106 of file utcb.h.

13.31.2.4 l4_utcb_exc_pc()

```
l4_umword_t l4_utcb_exc_pc (
            l4_exc_regs_t const * u )  [inline]
```

Access function to get the program counter of the exception state.

Parameters

u	UTCB

Returns

The program counter register out of the exception state.

Examples

examples/sys/aliens/main.c, and examples/sys/singlestep/main.c.

Definition at line 91 of file utcb.h.

13.31.2.5 l4_utcb_exc_pc_set()

```
void l4_utcb_exc_pc_set (
            l4_exc_regs_t * u,
            l4_addr_t pc )  [inline]
```

Set the program counter register in the exception state.

Parameters

u	UTCB
pc	The program counter to set.

Definition at line 96 of file utcb.h.

13.32 Extended vCPU support

extended vCPU handling functionality.

Collaboration diagram for Extended vCPU support:

Functions

- int l4vcpu_ext_alloc (l4_vcpu_state_t **vcpu, l4_addr_t *ext_state, l4_cap_idx_t task, l4_cap_idx_t regmgr) L4_NOTHROW

  Allocate state area for an extended vCPU.

13.32.1 Detailed Description

extended vCPU handling functionality.

13.32.2 Function Documentation

13.32.2.1 l4vcpu_ext_alloc()

```
int l4vcpu_ext_alloc (
            l4_vcpu_state_t ** vcpu,
            l4_addr_t * ext_state,
            l4_cap_idx_t task,
            l4_cap_idx_t regmgr )
```

Allocate state area for an extended vCPU.

Return values

vcpu	Allocated vcpu-state area.
ext_state	Allocated extended vcpu-state area.

Parameters

task	Task to use for allocation.
regmgr	Region manager to use for allocation.

Returns

0 for success, error code otherwise

13.33 Factory

C factory interface to create kernel objects, see L4::Factory for the C++ interface.

Collaboration diagram for Factory:

Functions

- l4_msgtag_t l4_factory_create_task (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_fpage_t utcb_area) L4_NOTHROW

  Create a new task.
- l4_msgtag_t l4_factory_create_thread (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW

  Create a new thread.
- l4_msgtag_t l4_factory_create_factory (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW

  Create a new factory.
- l4_msgtag_t l4_factory_create_gate (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW

  Create a new IPC gate.
- l4_msgtag_t l4_factory_create_irq (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW

  Create a new IRQ sender.
- l4_msgtag_t l4_factory_create_vm (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW

  Create a new virtual machine.
- l4_msgtag_t l4_factory_create (l4_cap_idx_t factory, long obj, l4_cap_idx_t target) L4_NOTHROW

  Create a new kernel object.

13.33.1 Detailed Description

C factory interface to create kernel objects, see L4::Factory for the C++ interface.

A factory is used to create all kinds of kernel objects:

- Task
- Thread
- Factory
- IPC-Gate API
- IRQs
- Virtual Machines

To create a new kernel object the caller has to specify the factory to use for creation. The caller has to allocate a capability slot where the kernel stores the new object's capability.

The factory is equipped with a limit that limits the amount of kernel memory available for that factory.

Note

The limit does not give any guarantee for the amount of available kernel memory.

Include File

```
#include <l4/sys/factory.h>
```

For the C++ interface refer to L4::Factory.

13.33.2 Function Documentation

13.33.2.1 l4_factory_create()

```
l4_msgtag_t l4_factory_create (
            l4_cap_idx_t factory,
            long obj,
            l4_cap_idx_t target )  [inline]
```

Create a new kernel object.

Parameters

	factory	Factory to use for creation.
	obj	Protocol ID to describe the type of the object to create.
out	*target*	The kernel stores the new objects's capability into this slot.

Returns

Syscall return tag

Definition at line 560 of file factory.h.

13.33.2.2 l4_factory_create_factory()

```
l4_msgtag_t l4_factory_create_factory (
            l4_cap_idx_t factory,
            l4_cap_idx_t target_cap,
            unsigned long limit )  [inline]
```

Create a new factory.

Parameters

	factory	Capability selector for factory to use for creation.
out	*target_cap*	The kernel stores the new factory's capability into this slot.
	limit	Limit for the new factory in bytes.

Returns

Syscall return tag

Note

The limit of the new factory is subtracted from the available amount of the factory used for creation.

Definition at line 414 of file factory.h.

13.33.2.3 l4_factory_create_gate()

```
l4_msgtag_t l4_factory_create_gate (
            l4_cap_idx_t factory,
            l4_cap_idx_t target_cap,
            l4_cap_idx_t thread_cap,
            l4_umword_t label )  [inline]
```

Create a new IPC gate.

Parameters

	factory	Capability selector for factory to use for creation.
out	*target_cap*	The kernel stores the new IPC gate's capability into this slot.
	thread_cap	Optional capability selector of the thread to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate.
	label	Optional label of the gate (precisely used if `thread_cap` is valid). If `thread_cap` is valid, `label` must be present.

Returns

Syscall return tag containing one of the following return codes.

Return values

L4_EOK	No error occurred.
-L4_ENOMEM	Out-of-memory during allocation of the lpc_gate object.
-L4_ENOENT	`thread_cap` is void or points to something that is not a thread.
-L4_EPERM	No L4_CAP_FPAGE_S rights on `thread_cap`.

An unbound IPC gate can be bound to a thread using l4_ipc_gate_bind_thread.

See also

IPC-Gate API

Definition at line 422 of file factory.h.

13.33.2.4 l4_factory_create_irq()

```
l4_msgtag_t l4_factory_create_irq (
            l4_cap_idx_t factory,
            l4_cap_idx_t target_cap )  [inline]
```

Create a new IRQ sender.

Parameters

	factory	Factory to use for creation.
out	*target_cap*	The kernel stores the new IRQ's capability into this slot.

Returns

Syscall return tag

See also

IRQs

Examples

examples/sys/isr/main.c.

Definition at line 430 of file factory.h.

13.33.2.5 l4_factory_create_task()

```
l4_msgtag_t l4_factory_create_task (
            l4_cap_idx_t factory,
            l4_cap_idx_t target_cap,
            l4_fpage_t utcb_area )  [inline]
```

Create a new task.

Parameters

	factory	Capability selector for factory to use for creation.
out	*target_cap*	The kernel stores the new task's capability into this slot.
	utcb_area	Flexpage that describes an area of kernel-user memory that can be used for UTCBs and vCPU state-save-areas of the new task.

Returns

Syscall return tag.

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to L4::Task::add_ku_mem / l4_task_add_ku_mem() for adding more of this type of memory.

See also

Task

Definition at line 400 of file factory.h.

13.33.2.6 l4_factory_create_thread()

```
l4_msgtag_t l4_factory_create_thread (
            l4_cap_idx_t factory,
            l4_cap_idx_t target_cap ) [inline]
```

Create a new thread.

Parameters

	factory	Capability selector for factory to use for creation.
out	*target_cap*	The kernel stores the new thread's capability into this slot.

Returns

Syscall return tag

See also

Thread

Examples

examples/sys/aliens/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 407 of file factory.h.

13.33.2.7 l4_factory_create_vm()

```
l4_msgtag_t l4_factory_create_vm (
            l4_cap_idx_t factory,
            l4_cap_idx_t target_cap ) [inline]
```

Create a new virtual machine.

Parameters

	factory	Capability selector for factory to use for creation.
out	*target_cap*	The kernel stores the new VM's capability into this slot.

Returns

Syscall return tag

See also

Virtual Machines

Definition at line 437 of file factory.h.

13.34 Fiasco extensions

Kernel debugger extensions of the Fiasco L4 implementation.

Collaboration diagram for Fiasco extensions:

Modules

- Fiasco real time scheduling extensions

  Real time scheduling extension for the Fiasco L4 implementation.
- Kernel Debugger

  Kernel debugger related functionality.

Files

- file segment.h

  l4f specific fs/gs manipulation
- file segment.h

  l4f specific segment manipulation

Functions

- l4_umword_t fiasco_tbuf_log (const char *text)

  Create new trace-buffer entry with describing <text>.
- l4_umword_t fiasco_tbuf_log_3val (const char *text, l4_umword_t v1, l4_umword_t v2, l4_umword_t v3)

  Create new trace-buffer entry with describing <text> and three additional values.
- l4_umword_t fiasco_tbuf_log_binary (const unsigned char *data)

  Create new trace-buffer entry with binary data.
- void fiasco_tbuf_clear (void)

  Clear trace-buffer.
- void fiasco_tbuf_dump (void)

  Dump trace-buffer to kernel console.
- long fiasco_ldt_set (l4_cap_idx_t task, void *ldt, unsigned int num_desc, unsigned int entry_number_start, l4_utcb_t *utcb)

  Set LDT segments descriptors.
- long fiasco_gdt_set (l4_cap_idx_t thread, void *desc, unsigned int size, unsigned int entry_number_start, l4_utcb_t *utcb)

  Set GDT segment descriptors.
- unsigned fiasco_gdt_get_entry_offset (l4_cap_idx_t thread, l4_utcb_t *utcb)

  Return the offset of the entry in the GDT.

13.34.1 Detailed Description

Kernel debugger extensions of the Fiasco L4 implementation.

13.34.2 Function Documentation

13.34.2.1 fiasco_gdt_get_entry_offset()

```
unsigned fiasco_gdt_get_entry_offset (
            l4_cap_idx_t thread,
            l4_utcb_t * utcb )  [inline]
```

Return the offset of the entry in the GDT.

Parameters

thread	Thread to get info from.
utcb	UTCB of the caller.

Definition at line 176 of file segment.h.

13.34.2.2 fiasco_gdt_set()

```
long fiasco_gdt_set (
            l4_cap_idx_t thread,
            void * desc,
            unsigned int size,
            unsigned int entry_number_start,
            l4_utcb_t * utcb )  [inline]
```

Set GDT segment descriptors.

Fiasco supports 3 consecutive entries, starting at the value returned by fiasco_gdt_get_entry_offset()

Parameters

thread	Thread to set the GDT entry for.
desc	Pointer to GDT descriptors.
size	Size of the descriptors in bytes (multiple of 8).
entry_number_start	Entry number to start (valid values: 0-2).
utcb	UTCB of the caller.

Return values

<0	Error.
L4_EOK	Success.

Definition at line 52 of file segment.h.

13.34.2.3 fiasco_ldt_set()

```
long fiasco_ldt_set (
            l4_cap_idx_t task,
            void * ldt,
            unsigned int num_desc,
            unsigned int entry_number_start,
            l4_utcb_t * utcb )  [inline]
```

Set LDT segments descriptors.

Parameters

task	Task to set the segment for.
ldt	Pointer to LDT hardware descriptors.
num_desc	Number of descriptors.
entry_number_start	Entry number to start.
utcb	UTCB of the caller.

Return values

-L4_ENOSYS	The kernel configuration doesn't support this feature.
-L4_EINVAL	Invalid descriptor or invalid entry number.
L4_EOK	Success.

Note

This feature is not available if the kernel is configured with page table isolation.

Definition at line 163 of file segment.h.

References L4_EINVAL, and L4_TASK_LDT_X86_MAX_ENTRIES.

13.34.2.4 fiasco_tbuf_log()

```
l4_umword_t fiasco_tbuf_log (
            const char * text )  [inline]
```

Create new trace-buffer entry with describing <text>.

Parameters

text	Logging text

Returns

Pointer to trace-buffer entry

Definition at line 35 of file __ktrace-impl.h.

References l4_error().

Here is the call graph for this function:

13.34.2.5 fiasco_tbuf_log_3val()

```
l4_umword_t fiasco_tbuf_log_3val (
            const char * text,
            l4_umword_t v1,
            l4_umword_t v2,
            l4_umword_t v3 )  [inline]
```

Create new trace-buffer entry with describing <text> and three additional values.

Parameters

text	Logging text
v1	first value
v2	second value
v3	third value

Returns

Pointer to trace-buffer entry

Definition at line 42 of file __ktrace-impl.h.

References l4_error().

Here is the call graph for this function:

13.34.2.6 fiasco_tbuf_log_binary()

```
l4_umword_t fiasco_tbuf_log_binary (
            const unsigned char * data )  [inline]
```

Create new trace-buffer entry with binary data.

Parameters

data	binary data

Returns

Pointer to trace-buffer entry

Definition at line 65 of file __ktrace-impl.h.

References l4_error().

Here is the call graph for this function:

13.35 Fiasco real time scheduling extensions

Real time scheduling extension for the Fiasco L4 implementation.

Collaboration diagram for Fiasco real time scheduling extensions:

Real time scheduling extension for the Fiasco L4 implementation.

13.36 Fiasco-UX Virtual devices

Virtual hardware devices, provided by Fiasco-UX.

Collaboration diagram for Fiasco-UX Virtual devices:

Data Structures

- struct l4_vhw_entry

  Description of a device.
- struct l4_vhw_descriptor

  Virtual hardware devices description.

Enumerations

- enum l4_vhw_entry_type { L4_TYPE_VHW_NONE , L4_TYPE_VHW_FRAMEBUFFER , L4_TYPE_VHW_INPUT , L4_TYPE_VHW_NET }

  Type of device.

13.36.1 Detailed Description

Virtual hardware devices, provided by Fiasco-UX.

Include File

```
#include <l4/sys/vhw.h>
```

13.36.2 Enumeration Type Documentation

13.36.2.1 l4_vhw_entry_type

```
enum l4_vhw_entry_type
```

Type of device.

Enumerator

L4_TYPE_VHW_NONE	None entry.
L4_TYPE_VHW_FRAMEBUFFER	Framebuffer device.
L4_TYPE_VHW_INPUT	Input device.
L4_TYPE_VHW_NET	Network device.

Definition at line 44 of file vhw.h.

13.37 Flex pages

Flex-page related API.

Collaboration diagram for Flex pages:

Data Structures

- union l4_fpage_t

  L4 flexpage type.
- struct l4_snd_fpage_t

  Send-flex-page types.

Enumerations

- enum l4_fpage_consts {
  L4_FPAGE_RIGHTS_SHIFT = 0 , L4_FPAGE_TYPE_SHIFT = 4 , L4_FPAGE_SIZE_SHIFT = 6 , L4_FPAGE_ADDR_SHIFT = 12 ,
  L4_FPAGE_RIGHTS_BITS = 4 , L4_FPAGE_TYPE_BITS = 2 , L4_FPAGE_SIZE_BITS = 6 , L4_FPAGE_ADDR_BITS = L4_MWORD_BITS - L4_FPAGE_ADDR_SHIFT ,
  L4_FPAGE_RIGHTS_MASK , **L4_FPAGE_TYPE_MASK** , **L4_FPAGE_SIZE_MASK** , **L4_FPAGE_ADDR_MASK** = ~0UL << L4_FPAGE_ADDR_SHIFT }

  L4 flexpage structure.
- enum { L4_WHOLE_ADDRESS_SPACE = 63 }

  Constants for flexpages.
- enum L4_fpage_rights {
  L4_FPAGE_X = 1 , L4_FPAGE_W = 2 , L4_FPAGE_RO = 4 , L4_FPAGE_RW = L4_FPAGE_RO | L4_FPAGE_W ,
  L4_FPAGE_RX = L4_FPAGE_RO | L4_FPAGE_X , L4_FPAGE_RWX = L4_FPAGE_RW | L4_FPAGE_X }

  Memory and IO port flex-page rights.
- enum L4_cap_fpage_rights {
  L4_CAP_FPAGE_W = 0x1 , L4_CAP_FPAGE_S = 0x2 , L4_CAP_FPAGE_R = 0x4 , L4_CAP_FPAGE_RO = 0x4 ,
  L4_CAP_FPAGE_D = 0x8 , L4_CAP_FPAGE_RW = L4_CAP_FPAGE_R | L4_CAP_FPAGE_W ,
  L4_CAP_FPAGE_RS = L4_CAP_FPAGE_R | L4_CAP_FPAGE_S , L4_CAP_FPAGE_RWS = L4_CAP_FPAGE_RW | L4_CAP_FPAGE_S ,
  L4_CAP_FPAGE_RWSD = L4_CAP_FPAGE_RWS | L4_CAP_FPAGE_D , L4_CAP_FPAGE_RWD = L4_CAP_FPAGE_RW | L4_CAP_FPAGE_D , L4_CAP_FPAGE_RSD = L4_CAP_FPAGE_RS | L4_CAP_FPAGE_D }

  Cap-flex-page rights.
- enum L4_fpage_type { L4_FPAGE_SPECIAL = 0 , L4_FPAGE_MEMORY = 1 , L4_FPAGE_IO = 2 , L4_FPAGE_OBJ = 3 }

  Flex-page type.
- enum L4_fpage_control { L4_FPAGE_CONTROL_OFFSET_SHIFT = 12 , L4_FPAGE_CONTROL_MASK = ~0UL << L4_FPAGE_CONTROL_OFFSET_SHIFT }

  Flex-page map control flags.

- enum L4_obj_fpage_ctl {
  L4_FPAGE_C_REF_CNT = 0x00 , L4_FPAGE_C_NO_REF_CNT = 0x10 , L4_FPAGE_C_OBJ_RIGHT1 = 0x20 , L4_FPAGE_C_OBJ_RIGHT2 = 0x40 ,
  L4_FPAGE_C_OBJ_RIGHT3 = 0x80 , L4_FPAGE_C_OBJ_RIGHTS = 0xe0 , L4_FPAGE_C_IPCGATE_SVR = L4_FPAGE_C_OBJ_RIGHT1 }

  Flex-page map control for capabilities (snd_base)
- enum l4_fpage_cacheability_opt_t { L4_FPAGE_CACHE_OPT = 0x1 , L4_FPAGE_CACHEABLE = 0x3 , L4_FPAGE_BUFFERABLE = 0x5 , L4_FPAGE_UNCACHEABLE = 0x1 }

  Flex-page cacheability option.
- enum { L4_WHOLE_IOADDRESS_SPACE = 16 , L4_IOPORT_MAX = (1L << L4_WHOLE_IOADDRESS↩_SPACE) }

  Special constants for IO flex pages.

Functions

- l4_fpage_t l4_fpage (l4_addr_t address, unsigned int size, unsigned char rights) L4_NOTHROW

  Create a memory flex page.
- l4_fpage_t l4_fpage_all (void) L4_NOTHROW

  Get a flex page, describing all address spaces at once.
- l4_fpage_t l4_fpage_invalid (void) L4_NOTHROW

  Get an invalid flex page.
- l4_fpage_t l4_iofpage (unsigned long port, unsigned int size) L4_NOTHROW

  Create an IO-port flex page.
- l4_fpage_t l4_obj_fpage (l4_cap_idx_t obj, unsigned int order, unsigned char rights) L4_NOTHROW

  Create a kernel-object flex page.
- int l4_is_fpage_writable (l4_fpage_t fp) L4_NOTHROW

  Test if the flex page is writable.
- unsigned l4_fpage_rights (l4_fpage_t f) L4_NOTHROW

  Return rights from a flex page.
- unsigned l4_fpage_type (l4_fpage_t f) L4_NOTHROW

  Return type from a flex page.
- unsigned l4_fpage_size (l4_fpage_t f) L4_NOTHROW

  Return size from a flex page.
- unsigned long l4_fpage_page (l4_fpage_t f) L4_NOTHROW

  Return the page part from a flex page.
- l4_addr_t l4_fpage_memaddr (l4_fpage_t f) L4_NOTHROW

  Return the memory address from the memory flex page.
- l4_cap_idx_t l4_fpage_obj (l4_fpage_t f) L4_NOTHROW

  Return the capability index from the object flex page.
- unsigned long l4_fpage_ioport (l4_fpage_t f) L4_NOTHROW

  Return the IO port number from the IO flex page.
- l4_fpage_t l4_fpage_set_rights (l4_fpage_t src, unsigned char new_rights) L4_NOTHROW

  Set new right in a flex page.
- int l4_fpage_contains (l4_fpage_t fpage, l4_addr_t addr, unsigned size) L4_NOTHROW

  Test whether a given range is completely within an fpage.
- unsigned char l4_fpage_max_order (unsigned char order, l4_addr_t addr, l4_addr_t min_addr, l4_addr_t max_addr, l4_addr_t hotspot=0)

  Determine maximum flex page size of a region.

13.37.1 Detailed Description

Flex-page related API.

A flex page is a page with a variable size, that can describe memory, IO-Ports (IA32 only), and sets of kernel objects.

A flex page describes an always size aligned region of an address space. The size is given in a log2 scale. This means the size in elements (bytes for memory, ports for IO-Ports, and capabilities for kernel objects) is always a power of two.

A flex page also carries type and access right information for the described region. The type information selects the address space in which the flex page is valid. Access rights have a meaning depending on the specific address space (type).

There exists a special type for defining *receive windows* or for the l4_task_unmap() method, that can be used to describe all address spaces (all types) with a single flex page.

13.37.2 Enumeration Type Documentation

13.37.2.1 anonymous enum

`anonymous enum`

Constants for flexpages.

Enumerator

L4_WHOLE_ADDRESS_SPACE	Whole address space size. This value does not only specify the log2 size of the biggest possible memory flex page. It can be also used as size for a special flex page to define a flex page which completely covers all spaces.

Definition at line 91 of file __l4_fpage.h.

13.37.2.2 anonymous enum

`anonymous enum`

Special constants for IO flex pages.

Enumerator

L4_WHOLE_IOADDRESS_SPACE	Whole I/O address space size. In contrast to L4_WHOLE_ADDRESS_SPACE, this value forms the log2 size of the biggest possible I/O flex page.
L4_IOPORT_MAX	Maximum I/O port address plus 1.

Definition at line 300 of file __l4_fpage.h.

13.37.2.3 L4_cap_fpage_rights

`enum L4_cap_fpage_rights`

Cap-flex-page rights.

Capabilities are modified or transfered with map and unmap operations. For that capabilities are wrapped into flex-page objects. The flex-page carries a set of rights the sender wants to hand over to the receiver along with the capability.

For the user only the 'S' and the 'W' right are visible. Other rights such as the 'D' right are internal to the corresponding kernel object and cannot be evaluated by the receiver.

Note

A thread can also map a capability from its task's capability table with a reduced set of rights into another slot of its own capability table.

Enumerator

L4_CAP_FPAGE_W	Interface specific 'W' right for capability flex-pages. The semantics of the 'W' right is defined by the protocol. For example in case of a dataspace cap, the 'W' right is needed to get a writable dataspace.
L4_CAP_FPAGE_S	Interface specific 'S' right for capability flex-pages. The semantics of the 'S' right is defined by the interface. The kernel masks this right with the 'S' right of the IPC gate over which the capability is mapped. That means that the receiver capability will only have the 'S' right set if both the flex-page and the IPC gate have the 'S' bit set.
L4_CAP_FPAGE_R	Read right for capability flex-pages. This is always required, otherwise no capability is mapped.
L4_CAP_FPAGE_RO	Read right for capability flex-pages. This is always required, otherwise no capability is mapped.
L4_CAP_FPAGE_D	Delete right for capability flex-pages. This allows the receiver to delete the corresponding kernel object using unmap() regardless of other tasks still holding a capability to the kernel object. Such capabilities are set to an empty capability if the object is deleted.
L4_CAP_FPAGE_RW	Read and interface specific 'W' right for capability flex-pages. The semantics of the 'W' right is defined by the interface. **See also** L4_CAP_FPAGE_W
L4_CAP_FPAGE_RS	Read and interface specific 'S' right for capability flex-pages. The semantics of the 'S' right is defined by the interface. **See also** L4_CAP_FPAGE_S

Enumerator

L4_CAP_FPAGE_RWS	Read, interface specific 'W', and 'S' rights for capability flex-pages. The semantics of the 'W' and 'S' right are defined by the interface. **See also** L4_CAP_FPAGE_R, L4_CAP_FPAGE_W, and L4_CAP_FPAGE_S
L4_CAP_FPAGE_RWSD	Full rights for capability flex-pages. **See also** L4_CAP_FPAGE_R, L4_CAP_FPAGE_W, L4_CAP_FPAGE_S, and L4_CAP_FPAGE_D
L4_CAP_FPAGE_RWD	Read, write, and delete right for capability flex-pages. **See also** L4_CAP_FPAGE_R, L4_CAP_FPAGE_W, and L4_CAP_FPAGE_D
L4_CAP_FPAGE_RSD	Read, 'S', and delete right for capability flex-pages. **See also** L4_CAP_FPAGE_R, L4_CAP_FPAGE_S, and L4_CAP_FPAGE_D

Definition at line 149 of file __l4_fpage.h.

13.37.2.4 l4_fpage_cacheability_opt_t

`enum l4_fpage_cacheability_opt_t`

Flex-page cacheability option.

Enumerator

L4_FPAGE_CACHE_OPT	Enable the cacheability option in a send flex page.
L4_FPAGE_CACHEABLE	Cacheability option to enable caches for the mapping.
L4_FPAGE_BUFFERABLE	Cacheability option to enable buffered writes for the mapping.
L4_FPAGE_UNCACHEABLE	Cacheability option to disable caching for the mapping.

Definition at line 281 of file __l4_fpage.h.

13.37.2.5 l4_fpage_consts

`enum l4_fpage_consts`

L4 flexpage structure.

Enumerator

L4_FPAGE_RIGHTS_SHIFT	Access permissions shift.
L4_FPAGE_TYPE_SHIFT	Flexpage type shift (memory, IO port, obj...)
L4_FPAGE_SIZE_SHIFT	Flexpage size shift (log2-based)
L4_FPAGE_ADDR_SHIFT	Page address shift.
L4_FPAGE_RIGHTS_BITS	Access permissions size.
L4_FPAGE_TYPE_BITS	Flexpage type size (memory, IO port, obj...)
L4_FPAGE_SIZE_BITS	Flexpage size size (log2-based)
L4_FPAGE_ADDR_BITS	Page address size.
L4_FPAGE_RIGHTS_MASK	Mask to get the flexpage rights.

Definition at line 57 of file __l4_fpage.h.

13.37.2.6 L4_fpage_control

`enum L4_fpage_control`

Flex-page map control flags.

Enumerator

L4_FPAGE_CONTROL_OFFSET_SHIFT	Number of bits an index must be shifted or an address must be aligned to in the control word.
L4_FPAGE_CONTROL_MASK	Mask for truncating the lower bits of the send base or the index of the control word.

Definition at line 240 of file __l4_fpage.h.

13.37.2.7 L4_fpage_rights

`enum L4_fpage_rights`

Memory and IO port flex-page rights.

For IO flexpages, bit 1 and bit 2 are a combined read/write right. In a map operation, the receiver receives the IO port capability when the sender possesses it and at least one of these bits is present. For an unmap operation, the absence of one of those bits is sufficient to unmap the IO port capability.

Enumerator

L4_FPAGE_X	Executable flex page.
L4_FPAGE_W	Writable flex page.
L4_FPAGE_RO	Read-only flex page
L4_FPAGE_RW	Read-write flex page.
L4_FPAGE_RX	Read-execute flex page.
L4_FPAGE_RWX	Read-write-execute flex page.

Definition at line 122 of file __l4_fpage.h.

13.37.2.8 L4_fpage_type

`enum L4_fpage_type`

Flex-page type.

Enumerator

L4_FPAGE_SPECIAL	Special flex page, either invalid or all spaces.
L4_FPAGE_MEMORY	Memory flex page.
L4_FPAGE_IO	IO-port flex page.
L4_FPAGE_OBJ	Object flex page (capabilities).

Definition at line 229 of file __l4_fpage.h.

13.37.2.9 L4_obj_fpage_ctl

`enum L4_obj_fpage_ctl`

Flex-page map control for capabilities (snd_base)

These rights need to be added to the snd_base when mapping and control internal behavior. The exact meaning depends on the type of capability (currently used only with IPC gates).

Enumerator

L4_FPAGE_C_REF_CNT	Mapping is reference-counted (default).
L4_FPAGE_C_NO_REF_CNT	Don't increase the reference counter.
L4_FPAGE_C_OBJ_RIGHT1	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHT2	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHT3	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHTS	All Object-type specific right bits.
L4_FPAGE_C_IPCGATE_SVR	The receiver may invoke IPC-gate-specific functions on the capability, e.g. bind a thread to the gate and modify the label. Needed if the receiver implements the server side of an IPC gate.

Definition at line 259 of file __l4_fpage.h.

13.37.3 Function Documentation

13.37.3.1 l4_fpage()

```
l4_fpage_t l4_fpage (
            l4_addr_t address,
            unsigned int size,
            unsigned char rights )  [inline]
```

Create a memory flex page.

Parameters

address	Flex-page start address
size	Flex-page size (log2), L4_WHOLE_ADDRESS_SPACE to specify the whole address space (with `address` 0). The minimum log2 size of a memory flex page is defined by L4_LOG2_PAGESIZE according to the size of the smallest virtual page supported by the MMU.
rights	Access rights, see L4_fpage_rights

Returns

Memory flex page

Definition at line 664 of file __l4_fpage.h.

References L4_FPAGE_MEMORY.

13.37.3.2 l4_fpage_all()

```
l4_fpage_t l4_fpage_all (
            void )  [inline]
```

Get a flex page, describing all address spaces at once.

Returns

Special *all-spaces* flex page.

Note

This flex page can be used to define a receive window where the sender can send objects of any type, or for an unmap item completely covering all spaces of the target task. It does not make sense to use this flex page as send item.

Definition at line 684 of file __l4_fpage.h.

References L4_FPAGE_SPECIAL, and L4_WHOLE_ADDRESS_SPACE.

13.37.3.3 l4_fpage_contains()

```
int l4_fpage_contains (
            l4_fpage_t fpage,
            l4_addr_t addr,
            unsigned size )  [inline]
```

Test whether a given range is completely within an fpage.

Parameters

fpage	Flex page
addr	Address
size	Size of range in log2.

Return values

==0	The range is not completely in the fpage.
!=0	The range is within the fpage.

Definition at line 716 of file __l4_fpage.h.

References l4_fpage_memaddr(), and l4_fpage_size().

Here is the call graph for this function:

13.37.3.4 l4_fpage_invalid()

```
l4_fpage_t l4_fpage_invalid (
            void ) [inline]
```

Get an invalid flex page.

Returns

Special *invalid* flex page.

Definition at line 690 of file __l4_fpage.h.

References L4_FPAGE_SPECIAL.

13.37.3.5 l4_fpage_ioport()

```
unsigned long l4_fpage_ioport (
            l4_fpage_t f ) [inline]
```

Return the IO port number from the IO flex page.

Parameters

f	Flex page

Returns

IO port number from the given IO flex page.

Precondition

`f` must be an IO flex page (`l4_fpage_type(f) == L4_FPAGE_IO`) and

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 620 of file __l4_fpage.h.

13.37.3.6 l4_fpage_max_order()

```
unsigned char l4_fpage_max_order (
            unsigned char order,
            l4_addr_t addr,
            l4_addr_t min_addr,
            l4_addr_t max_addr,
            l4_addr_t hotspot = 0 )  [inline]
```

Determine maximum flex page size of a region.

Parameters

order	Order value to start with (e.g. for memory L4_LOG2_PAGESIZE would be used)
addr	Address to be covered by the flex page.
min_addr	Start of region / minimal address (including).
max_addr	End of region / maximal address (excluding).
hotspot	(Optional) hot spot.

Returns

Maximum order (log2-size) possible.

Note

The start address of the flex-page can be determined with l4_trunc_size(addr, returnvalue)

Definition at line 724 of file __l4_fpage.h.

References l4_trunc_size().

Here is the call graph for this function:

13.37.3.7 l4_fpage_memaddr()

```
l4_addr_t l4_fpage_memaddr (
            l4_fpage_t f ) [inline]
```

Return the memory address from the memory flex page.

Parameters

f	Flex page

Returns

Page address from the given memory flex page.

Precondition

`f` must be a memory flex page (`l4_fpage_type(f) == L4_FPAGE_MEMORY`).

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 626 of file __l4_fpage.h.

Referenced by l4_fpage_contains().

Here is the caller graph for this function:

13.37.3.8 l4_fpage_obj()

```
l4_cap_idx_t l4_fpage_obj (
            l4_fpage_t f ) [inline]
```

Return the capability index from the object flex page.

Parameters

f	Flex page

Returns

Capability index from the given object flex page.

Precondition

`f` must be an object flex page (`l4_fpage_type(f) == L4_FPAGE_OBJ`)

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 632 of file __l4_fpage.h.

13.37.3.9 l4_fpage_page()

```
unsigned long l4_fpage_page (
            l4_fpage_t f ) [inline]
```

Return the page part from a flex page.

Parameters

f	Flex page

Returns

Page part of the given flex page.

Note

The meaning of the page part depends on the flex-page type.

Definition at line 614 of file __l4_fpage.h.

13.37.3.10 l4_fpage_rights()

```
unsigned l4_fpage_rights (
            l4_fpage_t f )  [inline]
```

Return rights from a flex page.

Parameters

f	Flex page

Returns

Size part of the given flex page.

Definition at line 596 of file __l4_fpage.h.

References L4_FPAGE_RIGHTS_MASK, and L4_FPAGE_RIGHTS_SHIFT.

Referenced by l4_is_fpage_writable().

Here is the caller graph for this function:

13.37.3.11 l4_fpage_set_rights()

```
l4_fpage_t l4_fpage_set_rights (
            l4_fpage_t src,
            unsigned char new_rights )  [inline]
```

Set new right in a flex page.

Parameters

src	Flex page
new_rights	New rights

Returns

Modified flex page with new rights.

Definition at line 655 of file __l4_fpage.h.

References l4_fpage_t::raw.

13.37.3.12 l4_fpage_size()

```
unsigned l4_fpage_size (
            l4_fpage_t f ) [inline]
```

Return size from a flex page.

Parameters

f	Flex page

Returns

Size part of the given flex page.

See also

l4_fpage_memaddr(), l4_fpage_obj(), l4_fpage_ioport()

Definition at line 608 of file __l4_fpage.h.

Referenced by l4_fpage_contains().

Here is the caller graph for this function:

13.37.3.13 l4_fpage_type()

```
unsigned l4_fpage_type (
            l4_fpage_t f ) [inline]
```

Return type from a flex page.

Parameters

f	Flex page

Returns

Type part of the given flex page.

Definition at line 602 of file __l4_fpage.h.

13.37.3.14 l4_iofpage()

```
l4_fpage_t l4_iofpage (
            unsigned long port,
            unsigned int size )  [inline]
```

Create an IO-port flex page.

Parameters

port	I/O-flex-page port base
size	I/O-flex-page size (log2), L4_WHOLE_IOADDRESS_SPACE to specify the whole I/O address space (with `port` 0)

Returns

I/O flex page

Definition at line 670 of file __l4_fpage.h.

References L4_FPAGE_ADDR_SHIFT, L4_FPAGE_IO, and L4_FPAGE_RW.

Referenced by l4util_ioport_map().

Here is the caller graph for this function:

13.37.3.15 l4_is_fpage_writable()

```
int l4_is_fpage_writable (
            l4_fpage_t fp )  [inline]
```

Test if the flex page is writable.

Parameters

fp	Flex page.

Return values

!=0	if flex page is writable.
==0	if flex pags is not writable.

Definition at line 697 of file __l4_fpage.h.

References l4_fpage_rights(), and L4_FPAGE_W.

Here is the call graph for this function:

13.37.3.16 l4_obj_fpage()

```
l4_fpage_t l4_obj_fpage (
            l4_cap_idx_t obj,
            unsigned int order,
            unsigned char rights )  [inline]
```

Create a kernel-object flex page.

Parameters

obj	Base capability selector.
order	Log2 size (number of capabilities).
rights	Access rights, see L4_cap_fpage_rights

Returns

Flex page for a set of kernel objects.

Note

L4_CAP_FPAGE_R is always required, otherwise no capability is mapped.

Examples

examples/sys/utcb-ipc/main.c.

Definition at line 676 of file __l4_fpage.h.

References L4_CAP_SHIFT, L4_FPAGE_ADDR_SHIFT, and L4_FPAGE_OBJ.

Referenced by L4::Cap_base::fpage().

Here is the caller graph for this function:

13.38 Functions for rendering bitmap data in frame buffers

Collaboration diagram for Functions for rendering bitmap data in frame buffers:

13.39 Functions for rendering bitmap fonts to frame buffers

Collaboration diagram for Functions for rendering bitmap fonts to frame buffers:

13.40 Functions to manipulate the local IDT

Collaboration diagram for Functions to manipulate the local IDT:

Data Structures

- struct l4util_idt_desc_t

  IDT entry.
- struct l4util_idt_header_t

  Header of an IDT table.

13.40.1 Detailed Description

13.41 IA32 Port I/O API

Collaboration diagram for IA32 Port I/O API:

Functions

- l4_uint8_t l4util_in8 (l4_uint16_t port)

  Read byte from I/O port.
- l4_uint16_t l4util_in16 (l4_uint16_t port)

  Read 16-bit-value from I/O port.
- l4_uint32_t l4util_in32 (l4_uint16_t port)

  Read 32-bit-value from I/O port.
- void l4util_ins8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)

  Read a block of 8-bit-values from I/O ports.
- void l4util_ins16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)

Read a block of 16-bit-values from I/O ports.

- void l4util_ins32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)

  Read a block of 32-bit-values from I/O ports.
- void l4util_out8 (l4_uint8_t value, l4_uint16_t port)

  Write byte to I/O port.
- void l4util_out16 (l4_uint16_t value, l4_uint16_t port)

  Write 16-bit-value to I/O port.
- void l4util_out32 (l4_uint32_t value, l4_uint16_t port)

  Write 32-bit-value to I/O port.
- void l4util_outs8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)

  Write a block of bytes to I/O port.
- void l4util_outs16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)

  Write a block of 16-bit-values to I/O port.
- void l4util_outs32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)

  Write block of 32-bit-values to I/O port.
- void l4util_iodelay (void)

  delay I/O port access by writing to port 0x80

13.41.1 Detailed Description

13.41.2 Function Documentation

13.41.2.1 l4util_in16()

```
l4_uint16_t l4util_in16 (
            l4_uint16_t port )  [inline]
```

Read 16-bit-value from I/O port.

Parameters

port	I/O port address

Returns

value

Definition at line 180 of file port_io.h.

13.41.2.2 l4util_in32()

```
l4_uint32_t l4util_in32 (
            l4_uint16_t port )  [inline]
```

Read 32-bit-value from I/O port.

Parameters

port	I/O port address

Returns

value

Definition at line 188 of file port_io.h.

13.41.2.3 l4util_in8()

```
l4_uint8_t l4util_in8 (
            l4_uint16_t port )  [inline]
```

Read byte from I/O port.

Parameters

port	I/O port address

Returns

value

Definition at line 172 of file port_io.h.

Referenced by l4util_irq_acknowledge().

Here is the caller graph for this function:

13.41.2.4 l4util_ins16()

```
void l4util_ins16 (
            l4_uint16_t port,
            l4_umword_t addr,
            l4_umword_t count )  [inline]
```

Read a block of 16-bit-values from I/O ports.

Parameters

port	I/O port address
addr	address of buffer
count	number of I/O operations

Definition at line 205 of file port_io.h.

13.41.2.5 l4util_ins32()

```
void l4util_ins32 (
            l4_uint16_t port,
            l4_umword_t addr,
            l4_umword_t count )  [inline]
```

Read a block of 32-bit-values from I/O ports.

Parameters

port	I/O port address
addr	address of buffer
count	number of I/O operations

Definition at line 214 of file port_io.h.

13.41.2.6 l4util_ins8()

```
void l4util_ins8 (
            l4_uint16_t port,
            l4_umword_t addr,
            l4_umword_t count )  [inline]
```

Read a block of 8-bit-values from I/O ports.

Parameters

port	I/O port address
addr	address of buffer
count	number of I/O operations

Definition at line 196 of file port_io.h.

13.41.2.7 l4util_out16()

```
void l4util_out16 (
            l4_uint16_t value,
            l4_uint16_t port )  [inline]
```

Write 16-bit-value to I/O port.

Parameters

port	I/O port address
value	value to write

Definition at line 229 of file port_io.h.

13.41.2.8 l4util_out32()

```
void l4util_out32 (
            l4_uint32_t value,
            l4_uint16_t port )  [inline]
```

Write 32-bit-value to I/O port.

Parameters

port	I/O port address
value	value to write

Definition at line 235 of file port_io.h.

13.41.2.9 l4util_out8()

```
void l4util_out8 (
            l4_uint8_t value,
            l4_uint16_t port )  [inline]
```

Write byte to I/O port.

Parameters

port	I/O port address
value	value to write

Definition at line 223 of file port_io.h.

Referenced by l4util_irq_acknowledge().

Here is the caller graph for this function:

13.41.2.10 l4util_outs16()

```
void l4util_outs16 (
            l4_uint16_t port,
            l4_umword_t addr,
            l4_umword_t count )  [inline]
```

Write a block of 16-bit-values to I/O port.

Parameters

port	I/O port address
addr	address of buffer
count	number of I/O operations

Definition at line 250 of file port_io.h.

13.41.2.11 l4util_outs32()

```
void l4util_outs32 (
            l4_uint16_t port,
            l4_umword_t addr,
            l4_umword_t count )  [inline]
```

Write block of 32-bit-values to I/O port.

Parameters

port	I/O port address
addr	address of buffer
count	number of I/O operations

Definition at line 259 of file port_io.h.

13.41.2.12 l4util_outs8()

```
void l4util_outs8 (
            l4_uint16_t port,
            l4_umword_t addr,
            l4_umword_t count )  [inline]
```

Write a block of bytes to I/O port.

Parameters

port	I/O port address
addr	address of buffer
count	number of I/O operations

Definition at line 241 of file port_io.h.

13.42 IO interface

Typedefs

- typedef l4vbus_resource_t l4io_resource_t

  Resource descriptor.
- typedef l4vbus_device_t l4io_device_t

  Device descriptor.

Enumerations

- enum l4io_iomem_flags_t {
  L4IO_MEM_NONCACHED = 0 , L4IO_MEM_CACHED = 1 , L4IO_MEM_USE_MTRR = 2 , **L4IO_MEM_↩ ATTR_MASK** = 0xf ,
  L4IO_MEM_WRITE_COMBINED = L4IO_MEM_USE_MTRR | L4IO_MEM_CACHED , L4IO_MEM_USE_RESERVED_AREA = 0x40 << 8 , L4IO_MEM_EAGER_MAP = 0x80 << 8 }

  Flags for IO memory.
- enum l4io_device_types_t {
  L4IO_DEVICE_INVALID = 0 , L4IO_DEVICE_PCI , L4IO_DEVICE_USB , L4IO_DEVICE_OTHER ,
  L4IO_DEVICE_ANY = ∼0 }

  Device types.
- enum l4io_resource_types_t {
  L4IO_RESOURCE_INVALID = L4VBUS_RESOURCE_INVALID , L4IO_RESOURCE_IRQ = L4VBUS_↩ RESOURCE_IRQ , L4IO_RESOURCE_MEM = L4VBUS_RESOURCE_MEM , L4IO_RESOURCE_PORT = L4VBUS_RESOURCE_PORT ,
  L4IO_RESOURCE_ANY = ∼0 }

  Resource types.

Functions

- long l4io_request_iomem (l4_addr_t phys, unsigned long size, int flags, l4_addr_t *virt)

  Request an IO memory region.
- long l4io_request_iomem_region (l4_addr_t phys, l4_addr_t virt, unsigned long size, int flags)

  Request an IO memory region and map it to a specified region.
- long l4io_release_iomem (l4_addr_t virt, unsigned long size)

  Release an IO memory region.
- long l4io_request_ioport (unsigned portnum, unsigned len)

  Request an IO port region.
- long l4io_release_ioport (unsigned portnum, unsigned len)

  Release an IO port region.
- int l4io_lookup_device (const char *devname, l4io_device_handle_t *dev_handle, l4io_device_t *dev, l4io↩_resource_handle_t *res_handle)

  Find a device by name.
- int l4io_lookup_resource (l4io_device_handle_t devhandle, enum l4io_resource_types_t type, l4io_↩resource_handle_t *reshandle, l4io_resource_t *res)

  Request a specific resource from a device description.
- l4_addr_t l4io_request_resource_iomem (l4io_device_handle_t devhandle, l4io_resource_handle_↩t *reshandle)

  Request IO memory.
- int l4io_has_resource (enum l4io_resource_types_t type, l4vbus_paddr_t start, l4vbus_paddr_t end)

  Check if a resource is available.

13.42.1 Detailed Description

13.42.2 Typedef Documentation

13.42.2.1 l4io_resource_t

```
typedef l4vbus_resource_t l4io_resource_t
```

Resource descriptor.

For IRQ types, the end field is not used, i.e. only a single interrupt can be described with a l4io_resource_t

Definition at line 69 of file types.h.

13.42.3 Enumeration Type Documentation

13.42.3.1 l4io_device_types_t

```
enum l4io_device_types_t
```

Device types.

Enumerator

L4IO_DEVICE_INVALID	Invalid type.
L4IO_DEVICE_PCI	PCI device.
L4IO_DEVICE_USB	USB device.
L4IO_DEVICE_OTHER	Any other device without unique IDs.
L4IO_DEVICE_ANY	any type

Definition at line 38 of file types.h.

13.42.3.2 l4io_iomem_flags_t

`enum l4io_iomem_flags_t`

Flags for IO memory.

Enumerator

L4IO_MEM_NONCACHED	Non-cache memory.
L4IO_MEM_CACHED	Cache memory.
L4IO_MEM_USE_MTRR	Use MTRR.
L4IO_MEM_USE_RESERVED_AREA	Use reserved area for mapping I/O memory. Flag only valid for l4io_request_iomem_region()
L4IO_MEM_EAGER_MAP	Eagerly map the I/O memory. Passthrough to the l4re-rm.

Definition at line 16 of file types.h.

13.42.3.3 l4io_resource_types_t

`enum l4io_resource_types_t`

Resource types.

Enumerator

L4IO_RESOURCE_INVALID	Invalid type.
L4IO_RESOURCE_IRQ	Interrupt resource.
L4IO_RESOURCE_MEM	I/O memory resource.
L4IO_RESOURCE_PORT	I/O port resource (x86 only)
L4IO_RESOURCE_ANY	any type

Definition at line 50 of file types.h.

13.42.4 Function Documentation

13.42.4.1 l4io_has_resource()

```
int l4io_has_resource (
            enum l4io_resource_types_t type,
            l4vbus_paddr_t start,
            l4vbus_paddr_t end )
```

Check if a resource is available.

Parameters

type	Type of resource
start	Minimal value.
end	Maximum value.

13.42.4.2 l4io_lookup_device()

```
int l4io_lookup_device (
            const char * devname,
            l4io_device_handle_t * dev_handle,
            l4io_device_t * dev,
            l4io_resource_handle_t * res_handle )
```

Find a device by name.

Parameters

devname	Name of device

Return values

dev_handle	Device handle for found device, can be NULL.
dev	Device information, filled by the function, can be NULL.
res_handle	Resource handle, can be NULL.

Returns

0 on success, error code otherwise

13.42.4.3 l4io_lookup_resource()

```
int l4io_lookup_resource (
            l4io_device_handle_t devhandle,
            enum l4io_resource_types_t type,
            l4io_resource_handle_t * reshandle,
            l4io_resource_t * res )
```

Request a specific resource from a device description.

Parameters

devhandle	Device handle.
type	Type of resource to request (see #l4io_resource_types_t)
reshandle	Resource handle, start with handle returned by device functions.

Return values

reshandle	Next resource handle.
res	Device descriptor

Returns

0 on success, error code otherwise, esp. -L4_ENOENT if no more resources found

13.42.4.4 l4io_release_iomem()

```
long l4io_release_iomem (
            l4_addr_t virt,
            unsigned long size )
```

Release an IO memory region.

Parameters

virt	Virtual address of region to free, see l4io_request_iomem
size	Size of the region to release.

Returns

0 on success, <0 on error

13.42.4.5 l4io_release_ioport()

```
long l4io_release_ioport (
            unsigned portnum,
            unsigned len )
```

Release an IO port region.

Parameters

portnum	Start of port range to release
len	Length of range to request

Returns

0 on success, <0 on error

Note

X86 architecture only

13.42.4.6 l4io_request_iomem()

```
long l4io_request_iomem (
            l4_addr_t phys,
            unsigned long size,
            int flags,
            l4_addr_t * virt )
```

Request an IO memory region.

Parameters

	phys	Physical address of the I/O memory region
	size	Size of the region in Bytes, granularity pages.
	flags	See l4io_iomem_flags_t
in,out	*virt*	Virtual address where the IO memory region should be mapped to. If the caller passes '0' a region in the caller's address space is searched and the virtual address is returned.

Return values

0	Success.
-L4_ENOENT	No area in the caller's address space could be found to map the IO memory region.
-L4_EPERM	Operation not allowed.
-L4_EINVAL	Invalid value.
-L4_EADDRNOTAVAIL	The requested virtual address is not available.
-L4_ENOMEM	The requested IO memory region could not be allocated.
<0	IPC errors.

Note

This function uses L4Re functionality to reserve a part of the virtual address space of the caller.

13.42.4.7 l4io_request_iomem_region()

```
long l4io_request_iomem_region (
            l4_addr_t phys,
            l4_addr_t virt,
            unsigned long size,
            int flags )
```

Request an IO memory region and map it to a specified region.

Parameters

phys	Physical address of the I/O memory region
virt	Virtual address.
size	Size of the region in Bytes, granularity pages.
flags	See l4io_iomem_flags_t

Return values

0	Success.
-L4_ENOENT	No area could be found to map the IO memory region.
-L4_EPERM	Operation not allowed.
-L4_EINVAL	Invalid value.
-L4_EADDRNOTAVAIL	The requested virtual address is not available.
-L4_ENOMEM	The requested IO memory region could not be allocated.
<0	IPC errors.

Note

This function uses L4Re functionality to reserve a part of the virtual address space of the caller.

13.42.4.8 l4io_request_ioport()

```
long l4io_request_ioport (
            unsigned portnum,
            unsigned len )
```

Request an IO port region.

Parameters

portnum	Start of port range to request
len	Length of range to request

Returns

0 on success, <0 on error

Note

X86 architecture only

13.42.4.9 l4io_request_resource_iomem()

```
l4_addr_t l4io_request_resource_iomem (
            l4io_device_handle_t devhandle,
            l4io_resource_handle_t * reshandle )
```

Request IO memory.

Parameters

	devhandle	Device handle.
in,out	*reshandle*	Resource handle from which IO memory should be requested. Upon successfull completion 'reshandle' points to the device's next resource.

Return values

0	An error occured. The value of 'reshandle' is undefined.
>0	The virtual address of the IO memory mapping.

13.43 IPC-Gate API

The C IPC gate interface, see L4::Ipc_gate for the C++ interface.

Collaboration diagram for IPC-Gate API:

Functions

- l4_msgtag_t l4_ipc_gate_get_infos (l4_cap_idx_t gate, l4_umword_t *label)

  Get information about the IPC-gate.
- l4_msgtag_t l4_rcv_ep_bind_thread (l4_cap_idx_t ep, l4_cap_idx_t thread, l4_umword_t label)

  Bind the IPC receive endpoint to a thread.

13.43.1 Detailed Description

The C IPC gate interface, see L4::Ipc_gate for the C++ interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the Factory interface.

Depending on the permissions of the capability used, an IPC gate forwards IPC to the Thread that is *bound* to the IPC gate (cf. l4_ipc_gate_bind_thread()). If the capability has the L4_FPAGE_C_IPCGATE_SVR permission, only IPC using a protocol different from the L4_PROTO_KOBJECT protocol is forwarded. Without the L4_FPAGE_C_IPCGATE_SVR permission, all IPC is forwarded. The latter is the usual case for a client in a client/server scenario. When no thread is bound yet, the forwarded IPC blocks until a thread is bound or the IPC times out.

Forwarded IPC is always forwarded to the userland of the bound thread. That means, the Thread interface of the bound thread is not accessible via an IPC gate. The IPC-Gate API of an IPC gate is only accessible if the capability used has the L4_FPAGE_C_IPCGATE_SVR permission (cf. previous paragraph). Conversely that means, if the capability used lacks the L4_FPAGE_C_IPCGATE_SVR permission, IPC-Gate API calls are forwarded to the bound thread instead of being processed by the IPC gate itself. In a client/server scenario, a client should only get IPC gate capabilities without L4_FPAGE_C_IPCGATE_SVR permission so the client cannot tamper with the IPC gate.

When binding a thread to an IPC gate, a user-defined, kernel protected, machine-word sized payload called the IPC gate's *label* is assigned to the IPC gate (cf. l4_ipc_gate_bind_thread()). When a send-only IPC or call IPC is forwarded via an IPC gate, the label provided by the sender is ignored and replaced by the IPC gate's label where the two least significant bits are the result of bitwise disjunction of the corresponding label bits with the L4_CAP_FPAGE_S and L4_CAP_FPAGE_W permissions of the capability used. Hence, the label provided via l4_ipc_gate_bind_thread() should usually have its two least significant bits set to zero. The replaced label is only visible to the bound thread upon receive. However, the configured label of an IPC gate can also be queried via l4_ipc_gate_get_infos() if the capability used has the L4_FPAGE_C_IPCGATE_SVR permission.

When deleting an IPC gate or when unbinding it from a thread, the label of IPC already in flight won't be changed. To ensure that no IPC from this IPC gate is received by a thread with an unexpected label, l4_thread_modify_sender_start() shall be used to change the labels of every pending IPC to that gate. This is also required if the label of an already bound IPC gate is changed. It is not necessary after binding the IPC gate to a thread for the first time.

Include File

```
#include <l4/sys/ipc_gate.h>
```

For the C++ interface refer to the L4::Ipc_gate documentation.

See also

Object Invocation

13.43.2 Function Documentation

13.43.2.1 l4_ipc_gate_get_infos()

```
l4_msgtag_t l4_ipc_gate_get_infos (
            l4_cap_idx_t gate,
            l4_umword_t * label )  [inline]
```

Get information about the IPC-gate.

Parameters

	gate	The IPC gate object to get information about.
out	*label*	The label of the IPC gate is returned here.

Returns

System call return tag.

Definition at line 146 of file ipc_gate.h.

13.43.2.2 l4_rcv_ep_bind_thread()

```
l4_msgtag_t l4_rcv_ep_bind_thread (
            l4_cap_idx_t ep,
            l4_cap_idx_t thread,
            l4_umword_t label )  [inline]
```

Bind the IPC receive endpoint to a thread.

Parameters

ep	The IPC receive endpoint object.
thread	The thread object that shall be bound to `ep`.
label	Label to assign to `ep`. The two least significant bits should usually be set to zero.

Returns

Syscall return tag containing one of the following return codes.

Return values

L4_EOK	Operation successful.
-L4_EINVAL	`thread` is not a thread object or other arguments were malformed.
-L4_EPERM	No L4_CAP_FPAGE_S rights on `ep` or `thread`.

The specified `label` is passed to the receiver of the incoming IPC. It is possible to re-bind a receive endpoint to the same or a different thread. In this case, IPC already in flight will be delivered with the old label to the previously bound thread unless l4_thread_modify_sender_start() is used to change these labels.

Examples

examples/sys/isr/main.c.

Definition at line 86 of file rcv_endpoint.h.

13.44 IRQ handling library

Collaboration diagram for IRQ handling library:

Modules

- Interface for asynchronous ISR handlers.

  This interface has just two (main) functions.
- Interface using direct functionality.

13.44.1 Detailed Description

13.45 IRQs

C IRQ interface, see L4::Irq for the C++ interface.

Collaboration diagram for IRQs:

Enumerations

- enum L4_irq_mode {
  L4_IRQ_F_NONE = 0 , L4_IRQ_F_LEVEL = 0x2 , L4_IRQ_F_EDGE = 0x0 , L4_IRQ_F_POS = 0x0 , L4_IRQ_F_NEG = 0x4 , L4_IRQ_F_BOTH = 0x8 , L4_IRQ_F_LEVEL_HIGH = 0x3 , L4_IRQ_F_LEVEL_LOW = 0x7 ,
  L4_IRQ_F_POS_EDGE = 0x1 , L4_IRQ_F_NEG_EDGE = 0x5 , L4_IRQ_F_BOTH_EDGE = 0x9 , L4_IRQ_F_MASK = 0xf ,
  L4_IRQ_F_SET_WAKEUP = 0x10 , L4_IRQ_F_CLEAR_WAKEUP = 0x20 }

  Interrupt attributes.

Functions

- l4_msgtag_t l4_irq_mux_chain (l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW

  Chain an IRQ to another master IRQ source.
- l4_msgtag_t l4_irq_mux_chain_u (l4_cap_idx_t irq, l4_cap_idx_t slave, l4_utcb_t *utcb) L4_NOTHROW

  Attach an IRQ to this multiplexer.
- l4_msgtag_t l4_irq_detach (l4_cap_idx_t irq) L4_NOTHROW

  Detach from an interrupt source.
- l4_msgtag_t l4_irq_detach_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW

  Detach from this interrupt.
- l4_msgtag_t l4_irq_trigger (l4_cap_idx_t irq) L4_NOTHROW

  Trigger an IRQ.
- l4_msgtag_t l4_irq_trigger_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW

  Trigger the object.
- l4_msgtag_t l4_irq_receive (l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW

  Unmask and wait for specified IRQ.
- l4_msgtag_t l4_irq_receive_u (l4_cap_idx_t irq, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW

  Unmask and wait for this IRQ.
- l4_msgtag_t l4_irq_wait (l4_cap_idx_t irq, l4_umword_t *label, l4_timeout_t to) L4_NOTHROW

  Unmask IRQ and wait for any message.
- l4_msgtag_t l4_irq_wait_u (l4_cap_idx_t irq, l4_umword_t *label, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW

  Unmask IRQ and (open) wait for any message.
- l4_msgtag_t l4_irq_unmask (l4_cap_idx_t irq) L4_NOTHROW

  Unmask IRQ.
- l4_msgtag_t l4_irq_unmask_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW

  Unmask IRQ.

13.45.1 Detailed Description

C IRQ interface, see L4::Irq for the C++ interface.

The IRQ interface provides access to abstract interrupts provided by the microkernel. Interrupts may be

- hardware interrupts provided by the platform interrupt controller,
- virtual device interrupts provided by the microkernel's virtual devices (virtual serial or trace buffer) or
- virtual interrupts that can be triggered by user programs (IRQs) via l4_irq_trigger().

For hardware and virtual device interrupts the Irq object must be bound to an interrupt source, see Interrupt controller. To receive interrupts, the Irq object must be bound to a thread, see l4_rcv_ep_bind_thread().

IRQ objects can be created using a factory, see the Factory API (use l4_factory_create_irq()).

Include File

```
#include <l4/sys/irq.h>
```

For the C++ interface refer to the L4::Irq API for an overview.

13.45.2 Enumeration Type Documentation

13.45.2.1 L4_irq_mode

`enum L4_irq_mode`

Interrupt attributes.

Enumerator

L4_IRQ_F_NONE	Flow types. None
L4_IRQ_F_LEVEL	Level triggered.
L4_IRQ_F_EDGE	Edge triggered.
L4_IRQ_F_POS	Positive trigger.
L4_IRQ_F_NEG	Negative trigger.
L4_IRQ_F_BOTH	Both edges trigger.
L4_IRQ_F_LEVEL_HIGH	Level high trigger.
L4_IRQ_F_LEVEL_LOW	Level low trigger.
L4_IRQ_F_POS_EDGE	Positive edge trigger.
L4_IRQ_F_NEG_EDGE	Negative edge trigger.
L4_IRQ_F_BOTH_EDGE	Both edges trigger.
L4_IRQ_F_MASK	Mask.
L4_IRQ_F_SET_WAKEUP	Wakeup source? Use irq as wakeup source
L4_IRQ_F_CLEAR_WAKEUP	Do not use irq as wakeup source.

Definition at line 80 of file icu.h.

13.45.3 Function Documentation

13.45.3.1 l4_irq_detach()

```
l4_msgtag_t l4_irq_detach (
            l4_cap_idx_t irq ) [inline]
```

Detach from an interrupt source.

Parameters

irq	The IRQ object that shall be detached.

Returns

Syscall return tag

Examples

examples/sys/isr/main.c.

Definition at line 294 of file irq.h.

References l4_irq_detach_u(), and l4_utcb().

Here is the call graph for this function:

13.45.3.2 l4_irq_detach_u()

```
l4_msgtag_t l4_irq_detach_u (
            l4_cap_idx_t irq,
            l4_utcb_t * utcb )  [inline]
```

Detach from this interrupt.

Parameters

irq	The IRQ object that shall be detached.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 249 of file irq.h.

Referenced by L4::Irq::detach(), and l4_irq_detach().

Here is the caller graph for this function:

13.45.3.3 l4_irq_mux_chain()

```
l4_msgtag_t l4_irq_mux_chain (
            l4_cap_idx_t irq,
            l4_cap_idx_t slave )  [inline]
```

Chain an IRQ to another master IRQ source.

Deprecated IRQ muxer objects are no longer supported by the kernel.

Parameters

irq	The master IRQ object.
slave	The slave that shall be attached to the master.

Returns

Syscall return tag

The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be a master IRQ object, bound to the real IRQ source. Note, the master IRQ must not have a thread bound to it.

This function allows to add a limited number of slave IRQs to this master IRQ, with the semantics that each of the slave IRQs is triggered whenever the master IRQ is triggered. The master IRQ will be masked automatically when an IRQ is delivered and shall be unmasked when all attached slave IRQs are unmasked.

Definition at line 288 of file irq.h.

References l4_irq_mux_chain_u(), and l4_utcb().

Here is the call graph for this function:

13.45.3.4 l4_irq_mux_chain_u()

```
l4_msgtag_t l4_irq_mux_chain_u (
            l4_cap_idx_t irq,
            l4_cap_idx_t slave,
            l4_utcb_t * utcb )  [inline]
```

Attach an IRQ to this multiplexer.

Parameters

irq	The master IRQ object.
slave	The slave that shall be attached to the master.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be an IRQ multiplexer (Irq_mux) bound to the real IRQ source. This function allows to add slave IRQs to this multiplexer.

Definition at line 237 of file irq.h.

Referenced by L4::Irq_mux::chain(), and l4_irq_mux_chain().

Here is the caller graph for this function:

13.45.3.5 l4_irq_receive()

```
l4_msgtag_t l4_irq_receive (
            l4_cap_idx_t irq,
            l4_timeout_t to )  [inline]
```

Unmask and wait for specified IRQ.

Parameters

irq	The IRQ object that shall be unmasked.
to	Timeout.

Returns

Syscall return tag

Examples

examples/sys/isr/main.c.

Definition at line 306 of file irq.h.

References l4_irq_receive_u(), and l4_utcb().

Referenced by L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process().

Here is the call graph for this function:

Here is the caller graph for this function:

13.45.3.6 l4_irq_receive_u()

```
l4_msgtag_t l4_irq_receive_u (
            l4_cap_idx_t irq,
            l4_timeout_t timeout,
            l4_utcb_t * utcb )  [inline]
```

Unmask and wait for this IRQ.

Parameters

irq	The IRQ object that shall be unmasked.
timeout	Timeout.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Note

If this is the function normally used for your IRQs consider using L4::Semaphore instead of L4::Irq.

Definition at line 264 of file irq.h.

Referenced by l4_irq_receive(), and L4::Irq::receive().

Here is the caller graph for this function:

13.45.3.7 l4_irq_trigger()

```
l4_msgtag_t l4_irq_trigger (
            l4_cap_idx_t irq ) [inline]
```

Trigger an IRQ.

Parameters

irq	The IRQ object that shall be triggered.

Returns

Syscall return tag.

Note that this function is a send only operation, i.e. there is no return value except for a failed send operation. Especially l4_error() will return an error value from the message tag which still contains the IRQ protocol used for the send operation.

Use l4_ipc_error() to check for (send) errors.

Definition at line 300 of file irq.h.

References l4_irq_trigger_u(), and l4_utcb().

Referenced by l4_semaphore_up().

Here is the call graph for this function:

Here is the caller graph for this function:

13.45.3.8 l4_irq_trigger_u()

```
l4_msgtag_t l4_irq_trigger_u (
            l4_cap_idx_t irq,
            l4_utcb_t * utcb )  [inline]
```

Trigger the object.

Parameters

irq	The IRQ object that shall be triggered.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag for a send-only operation, use l4_ipc_error() to check for errors (**do not** use l4_error()).

Note

This function is a send-only operation, this means there is no return value except for a failed send operation. Use l4_ipc_error() to check for errors, **do not** use l4_error(), because l4_error() will always return an error.

Definition at line 257 of file irq.h.

References L4_IPC_BOTH_TIMEOUT_0, l4_ipc_send(), l4_msgtag(), and L4_PROTO_IRQ.

Referenced by l4_irq_trigger(), and L4::Triggerable::trigger().

Here is the call graph for this function:

Here is the caller graph for this function:

13.45.3.9 l4_irq_unmask()

```
l4_msgtag_t l4_irq_unmask (
            l4_cap_idx_t irq ) [inline]
```

Unmask IRQ.

Parameters

irq	The IRQ object that shall be unmasked.

Returns

Syscall return tag

Note

l4_irq_wait() and l4_irq_receive() are doing the unmask themselves.

Definition at line 319 of file irq.h.

References l4_irq_unmask_u(), and l4_utcb().

Here is the call graph for this function:

13.45.3.10 l4_irq_unmask_u()

```
l4_msgtag_t l4_irq_unmask_u (
            l4_cap_idx_t irq,
            l4_utcb_t * utcb )  [inline]
```

Unmask IRQ.

Parameters

irq	The IRQ object that shall be unmasked.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag for a send-only operation, use l4_ipc_error() to check for errors (**do not** use l4_error()).

Note

This function is a send-only operation, this means there is no return value except for a failed send operation. Use l4_ipc_error() to check for errors, **do not** use l4_error(), because l4_error() will always return an error.

Irq::wait() and Irq::receive() operations already include an unmask(), do not use an extra unmask() in these cases.

Deprecated Use L4::Irq_eoi::unmask()

Definition at line 280 of file irq.h.

Referenced by l4_irq_unmask().

Here is the caller graph for this function:

13.45.3.11 l4_irq_wait()

```
l4_msgtag_t l4_irq_wait (
            l4_cap_idx_t irq,
            l4_umword_t * label,
            l4_timeout_t to )  [inline]
```

Unmask IRQ and wait for any message.

Parameters

irq	The IRQ object that shall be unmasked.
label	Receive label.
to	Timeout.

Returns

Syscall return tag

Definition at line 312 of file irq.h.

References l4_irq_wait_u(), and l4_utcb().

Here is the call graph for this function:

13.45.3.12 l4_irq_wait_u()

```
l4_msgtag_t l4_irq_wait_u (
            l4_cap_idx_t irq,
            l4_umword_t * label,
            l4_timeout_t timeout,
            l4_utcb_t * utcb )  [inline]
```

Unmask IRQ and (open) wait for any message.

Parameters

irq	The IRQ object that shall be unmasked.
label	The *protected label* shall be received here.
timeout	Timeout.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 271 of file irq.h.

Referenced by l4_irq_wait().

Here is the caller graph for this function:

13.46 Initial Environment

C interface of the initial environment that is provided to an L4 task.

Collaboration diagram for Initial Environment:

Data Structures

- struct l4re_env_cap_entry_t

  Entry in the L4Re environment array for the named inital objects.

Typedefs

- typedef struct l4re_env_cap_entry_t l4re_env_cap_entry_t

  Entry in the L4Re environment array for the named inital objects.

Functions

- l4re_env_t ∗ l4re_env (void) L4_NOTHROW

  Get L4Re initial environment.
- l4_kernel_info_t ∗ l4re_kip (void) L4_NOTHROW

  Get Kernel Info Page.
- l4_cap_idx_t l4re_env_get_cap (char const ∗name) L4_NOTHROW

  Get the capability selector for the object named name.
- l4_cap_idx_t l4re_env_get_cap_e (char const ∗name, l4re_env_t const ∗e) L4_NOTHROW

  Get the capability selector for the object named name.
- l4re_env_cap_entry_t const ∗ l4re_env_get_cap_l (char const ∗name, unsigned l, l4re_env_t const ∗e) L4_NOTHROW

  Get the full l4re_env_cap_entry_t for the object named name.

13.46.1 Detailed Description

C interface of the initial environment that is provided to an L4 task.

Include File

```
#include <l4/re/env.h>
```

For an explanation of the default task capabilites see l4_default_caps_t.

For the C++ interface refer to L4Re::Env.

13.46.2 Function Documentation

13.46.2.1 l4re_env()

```
l4re_env_t * l4re_env (
            void  )  [inline]
```

Get L4Re initial environment.

Returns

Pointer to L4Re initial environment.

Examples

examples/sys/aliens/main.c, examples/sys/isr/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 185 of file env.h.

Referenced by l4re_env_get_cap().

Here is the caller graph for this function:

13.46.2.2 l4re_env_get_cap()

```
l4_cap_idx_t l4re_env_get_cap (
            char const * name )  [inline]
```

Get the capability selector for the object named *name*.

Parameters

name	is the name of the object to lookup in the initial objects.

Returns

A valid capability selector if the object exists or an invalid capability selector if not (l4_is_invalid_cap()).

Examples

examples/sys/isr/main.c.

Definition at line 227 of file env.h.

References l4re_env(), and l4re_env_get_cap_e().

Here is the call graph for this function:

13.46.2.3 l4re_env_get_cap_e()

```
l4_cap_idx_t l4re_env_get_cap_e (
            char const * name,
            l4re_env_t const * e )  [inline]
```

Get the capability selector for the object named *name*.

Parameters

name	is the name of the object to lookup in the initial objects.
e	is the environment structure to use for the operation.

Returns

A valid capability selector if the object exists or an invalid capability selector if not (l4_is_invalid_cap()).

Definition at line 214 of file env.h.

References l4re_env_cap_entry_t::cap, L4_INVALID_CAP, and l4re_env_get_cap_l().

Referenced by l4re_env_get_cap().

Here is the call graph for this function:

Here is the caller graph for this function:

13.46.2.4 l4re_env_get_cap_l()

```
l4re_env_cap_entry_t const * l4re_env_get_cap_l (
            char const * name,
            unsigned l,
            l4re_env_t const * e )  [inline]
```

Get the full l4re_env_cap_entry_t for the object named *name*.

Parameters

name	is the name of the object to lookup in the initial objects.
l	is the length of the name string, thus *name* might not be zero terminated.
e	is the environment structure to use for the operation.

Returns

A pointer to an l4re_env_cap_entry_t if the object exists or NULL if not.

Definition at line 196 of file env.h.

References l4re_env_cap_entry_t::flags, and l4re_env_cap_entry_t::name.

Referenced by L4Re::Env::get(), and l4re_env_get_cap_e().

Here is the caller graph for this function:

13.46.2.5 l4re_kip()

```
l4_kernel_info_t * l4re_kip (
            void  )  [inline]
```

Get Kernel Info Page.

Returns

Pointer to Kernel Info Page (KIP) structure.

Examples

examples/sys/aliens/main.c, and examples/sys/ux-vhw/main.c.

Definition at line 189 of file env.h.

Referenced by L4::Poll_timeout_kipclock::set(), and L4::Poll_timeout_kipclock::test().

Here is the caller graph for this function:

13.47 Integer Types

Collaboration diagram for Integer Types:

Files

- file l4int.h

  Fixed sized integer types, generic version.
- file l4int.h

  Fixed sized integer types, arm version.
- file l4int.h

  Fixed sized integer types, amd64 version.
- file l4int.h

  Fixed sized integer types, x86 version.

Macros

- #define L4_MWORD_BITS 32

  Size of machine words in bits.
- #define L4_MWORD_BITS 64

  Size of machine words in bits.
- #define L4_MWORD_BITS 32

  Size of machine words in bits.

Typedefs

- typedef signed char l4_int8_t

  Signed 8bit value.
- typedef unsigned char l4_uint8_t

  Unsigned 8bit value.
- typedef signed short int l4_int16_t

  Signed 16bit value.
- typedef unsigned short int l4_uint16_t

  Unsigned 16bit value.
- typedef signed int l4_int32_t

  Signed 32bit value.
- typedef unsigned int l4_uint32_t

  Unsigned 32bit value.
- typedef signed long long l4_int64_t

  Signed 64bit value.
- typedef unsigned long long l4_uint64_t

  Unsigned 64bit value.
- typedef unsigned long l4_addr_t

  Address type.
- typedef signed long l4_mword_t

  Signed machine word.
- typedef unsigned long l4_umword_t

  Unsigned machine word.
- typedef l4_uint64_t l4_cpu_time_t

  CPU clock type.
- typedef l4_uint64_t l4_kernel_clock_t

  Kernel clock type.
- typedef unsigned int l4_size_t

  Unsigned size type.
- typedef signed int l4_ssize_t

  Signed size type.
- typedef unsigned long l4_size_t

  Unsigned size type.
- typedef signed long l4_ssize_t

  Signed size type.
- typedef unsigned int l4_size_t

  Unsigned size type.
- typedef signed int l4_ssize_t

  Signed size type.

13.47.1 Detailed Description

Include File

```
#include <l4/sys/l4int.h>
```

13.48 Interface for asynchronous ISR handlers with a given IRQ capability.

This group is just an enhanced version to l4irq_request() which takes a capability object instead of a plain number.

Collaboration diagram for Interface for asynchronous ISR handlers with a given IRQ capability.:

Functions

- l4irq_t ∗ l4irq_request_cap (l4_cap_idx_t irqcap, void(∗isr_handler)(void ∗), void ∗isr_data, int irq_thread_↩ prio, unsigned mode)

  Attach asychronous ISR handler to IRQ.

13.48.1 Detailed Description

This group is just an enhanced version to l4irq_request() which takes a capability object instead of a plain number.

13.48.2 Function Documentation

13.48.2.1 l4irq_request_cap()

```
l4irq_t* l4irq_request_cap (
            l4_cap_idx_t irqcap,
            void(*)(void *) isr_handler,
            void * isr_data,
            int irq_thread_prio,
            unsigned mode )
```

Attach asychronous ISR handler to IRQ.

Parameters

irqcap	IRQ capability
isr_handler	Handler routine that is called when an interrupt triggers
isr_data	Pointer given as argument to isr_handler
irq_thread_prio	L4 thread priority of the ISR handler. Give -1 for same priority as creator.
mode	Interrupt type,

See also

L4_irq_mode

Returns

Pointer to l4irq_t structure, 0 on error

13.49 Interface for asynchronous ISR handlers.

This interface has just two (main) functions.

Collaboration diagram for Interface for asynchronous ISR handlers.:

Modules

- Interface for asynchronous ISR handlers with a given IRQ capability.

  This group is just an enhanced version to l4irq_request() which takes a capability object instead of a plain number.

Functions

- l4irq_t ∗ l4irq_request (int irqnum, void(∗isr_handler)(void ∗), void ∗isr_data, int irq_thread_prio, unsigned mode)

  Attach asychronous ISR handler to IRQ.
- long l4irq_release (l4irq_t ∗irq)

  Release asynchronous ISR handler and free resources.

13.49.1 Detailed Description

This interface has just two (main) functions.

l4irq_request to install a handler for an interrupt and l4irq_release to uninstall the handler again and release all resources associated with it.

13.49.2 Function Documentation

13.49.2.1 l4irq_release()

```
long l4irq_release (
            l4irq_t * irq )
```

Release asynchronous ISR handler and free resources.

Parameters

irq	IRQ data structure

Returns

0 sucess, != 0 failure

Examples

examples/libs/libirq/async_isr.c.

13.49.2.2 l4irq_request()

```
l4irq_t* l4irq_request (
            int irqnum,
            void(*)(void *) isr_handler,
            void * isr_data,
            int irq_thread_prio,
            unsigned mode )
```

Attach asychronous ISR handler to IRQ.

Parameters

irqnum	IRQ number to request
isr_handler	Handler routine that is called when an interrupt triggers
isr_data	Pointer given as argument to isr_handler
irq_thread_prio	L4 thread priority of the ISR handler. Give -1 for same priority as creator.
mode	Interrupt type,

See also

L4_irq_mode

Returns

Pointer to l4irq_t structure, 0 on error

Examples

examples/libs/libirq/async_isr.c.

13.50 Interface using direct functionality.

Collaboration diagram for Interface using direct functionality.:

Functions

- l4irq_t * l4irq_attach_cap (l4_cap_idx_t irqcap)

  Attach/connect to IRQ.
- l4irq_t * l4irq_attach_cap_ft (l4_cap_idx_t irqcap, unsigned mode)

  Attach/connect to IRQ using given type.
- l4irq_t * l4irq_attach_thread_cap (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread)

  Attach/connect to IRQ.
- l4irq_t * l4irq_attach_thread_cap_ft (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread, unsigned mode)

  Attach/connect to IRQ using given type.

13.50.1 Detailed Description

13.50.2 Function Documentation

13.50.2.1 l4irq_attach_cap()

```
l4irq_t* l4irq_attach_cap (
            l4_cap_idx_t irqcap )
```

Attach/connect to IRQ.

Parameters

irqcap	IRQ capability

Returns

Pointer to l4irq_t structure, 0 on error

This l4irq_attach has to be called in the same thread as l4irq_wait and caller has to be a pthread thread.

13.50.2.2 l4irq_attach_cap_ft()

```
l4irq_t* l4irq_attach_cap_ft (
            l4_cap_idx_t irqcap,
            unsigned mode )
```

Attach/connect to IRQ using given type.

Parameters

irqcap	IRQ capability
mode	Interrupt type,

See also

L4_irq_mode

Returns

Pointer to l4irq_t structure, 0 on error

This l4irq_attach has to be called in the same thread as l4irq_wait and caller has to be a pthread thread.

13.50.2.3 l4irq_attach_thread_cap()

```
l4irq_t* l4irq_attach_thread_cap (
            l4_cap_idx_t irqcap,
            l4_cap_idx_t to_thread )
```

Attach/connect to IRQ.

Parameters

irqcap	IRQ capability
to_thread	Attach IRQ to this thread.

Returns

Pointer to l4irq_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

13.50.2.4 l4irq_attach_thread_cap_ft()

```
l4irq_t* l4irq_attach_thread_cap_ft (
            l4_cap_idx_t irqcap,
            l4_cap_idx_t to_thread,
            unsigned mode )
```

Attach/connect to IRQ using given type.

Parameters

irqcap	IRQ capability
to_thread	Attach IRQ to this thread.
mode	Interrupt type,

See also

L4_irq_mode

Returns

Pointer to l4irq_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

13.51 Interface using direct functionality.

Collaboration diagram for Interface using direct functionality.:

Modules

- Interface using direct functionality.

Functions

- l4irq_t ∗ l4irq_attach (int irqnum)

  Attach/connect to IRQ.
- l4irq_t ∗ l4irq_attach_ft (int irqnum, unsigned mode)

  Attach/connect to IRQ using given type.
- l4irq_t ∗ l4irq_attach_thread (int irqnum, l4_cap_idx_t to_thread)

  Attach/connect to IRQ.
- l4irq_t ∗ l4irq_attach_thread_ft (int irqnum, l4_cap_idx_t to_thread, unsigned mode)

  Attach/connect to IRQ using given type.
- long l4irq_wait (l4irq_t ∗irq)

  Wait for specified IRQ.
- long l4irq_unmask_and_wait_any (l4irq_t ∗unmask_irq, l4irq_t ∗∗ret_irq)

  Unmask a specific IRQ and wait for any attached IRQ.
- long l4irq_wait_any (l4irq_t ∗∗irq)

  Wait for any attached IRQ.
- long l4irq_unmask (l4irq_t ∗irq)

  Unmask a specific IRQ.
- long l4irq_detach (l4irq_t ∗irq)

  Detach from IRQ.

13.51.1 Detailed Description

13.51.2 Function Documentation

13.51.2.1 l4irq_attach()

```
l4irq_t* l4irq_attach (
            int irqnum )
```

Attach/connect to IRQ.

Parameters

irqnum	IRQ number to request

Returns

Pointer to l4irq_t structure, 0 on error

This l4irq_attach has to be called in the same thread as l4irq_wait and caller has to be a pthread thread.

Examples

examples/libs/libirq/loop.c.

13.51.2.2 l4irq_attach_ft()

```
l4irq_t* l4irq_attach_ft (
            int irqnum,
            unsigned mode )
```

Attach/connect to IRQ using given type.

Parameters

irqnum	IRQ number to request
mode	Interrupt type,

See also

L4_irq_mode

Returns

Pointer to l4irq_t structure, 0 on error

This l4irq_attach has to be called in the same thread as l4irq_wait and caller has to be a pthread thread.

13.51.2.3 l4irq_attach_thread()

```
l4irq_t* l4irq_attach_thread (
            int irqnum,
            l4_cap_idx_t to_thread )
```

Attach/connect to IRQ.

Parameters

irqnum	IRQ number to request
to_thread	Attach IRQ to this specified thread.

Returns

Pointer to l4irq_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

13.51.2.4 l4irq_attach_thread_ft()

```
l4irq_t* l4irq_attach_thread_ft (
            int irqnum,
            l4_cap_idx_t to_thread,
            unsigned mode )
```

Attach/connect to IRQ using given type.

Parameters

irqnum	IRQ number to request
to_thread	Attach IRQ to this specified thread.
mode	Interrupt type,

See also

L4_irq_mode

Returns

Pointer to l4irq_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

13.51.2.5 l4irq_detach()

```
long l4irq_detach (
            l4irq_t * irq )
```

Detach from IRQ.

Parameters

irq	IRQ data structure

Returns

0 on success, != 0 on error

13.51.2.6 l4irq_unmask()

```
long l4irq_unmask (
            l4irq_t * irq )
```

Unmask a specific IRQ.

Parameters

irq	IRQ data structure

Returns

0 on success, != 0 on error

This function is useful if a thread wants to wait for multiple IRQs using l4_ipc_wait.

13.51.2.7 l4irq_unmask_and_wait_any()

```
long l4irq_unmask_and_wait_any (
            l4irq_t * unmask_irq,
            l4irq_t ** ret_irq )
```

Unmask a specific IRQ and wait for any attached IRQ.

Parameters

unmask_irq	IRQ data structure for unmask.

Return values

ret_irq	Received interrupt.

Returns

0 on success, != 0 on error

13.51.2.8 l4irq_wait()

```
long l4irq_wait (
            l4irq_t * irq )
```

Wait for specified IRQ.

Parameters

irq	IRQ data structure

Returns

0 on success, != 0 on error

Examples

examples/libs/libirq/loop.c.

13.51.2.9 l4irq_wait_any()

```
long l4irq_wait_any (
            l4irq_t ** irq )
```

Wait for any attached IRQ.

Return values

irq	Received interrupt.

Returns

0 on success, != 0 on error

13.52 Internal Helpers

Collaboration diagram for Internal Helpers:

Data Structures

- struct L4::Types::Bool< V >

  Boolean meta type.
- struct L4::Types::False

  False meta value.
- struct L4::Types::True

  True meta value.
- struct L4::Types::Same< A, B >

  Compare two data types for equality.

13.52.1 Detailed Description

13.53 Internal constants

Internal sigma0 definitions.

Collaboration diagram for Internal constants:

Macros

- #define SIGMA0_REQ_MAGIC ~0xFFUL

  Request magic.
- #define SIGMA0_REQ_MASK ~0xFFUL

  Request mask.
- #define SIGMA0_REQ_ID_MASK 0xF0

  ID mask.
- #define SIGMA0_REQ_ID_FPAGE_RAM 0x60

  RAM.
- #define SIGMA0_REQ_ID_FPAGE_IOMEM 0x70

  I/O memory.
- #define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED 0x80

  Cached I/O memory.
- #define SIGMA0_REQ_ID_FPAGE_ANY 0x90

  Any.
- #define SIGMA0_REQ_ID_KIP 0xA0

  KIP.
- #define SIGMA0_REQ_ID_DEBUG_DUMP 0xC0

  Debug dump.
- #define SIGMA0_REQ_ID_NEW_CLIENT 0xD0

  New client.
- #define SIGMA0_IS_MAGIC_REQ(d1) ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)

  Check if magic.
- #define SIGMA0_REQ(x) (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## x)

  Construct.
- #define SIGMA0_REQ_FPAGE_RAM (SIGMA0_REQ(FPAGE_RAM))

  RAM.
- #define SIGMA0_REQ_FPAGE_IOMEM (SIGMA0_REQ(FPAGE_IOMEM))

  I/O memory.
- #define SIGMA0_REQ_FPAGE_IOMEM_CACHED (SIGMA0_REQ(FPAGE_IOMEM_CACHED))

  Cache I/O memory.
- #define SIGMA0_REQ_FPAGE_ANY (SIGMA0_REQ(FPAGE_ANY))

  Any.
- #define SIGMA0_REQ_KIP (SIGMA0_REQ(KIP))

  KIP.
- #define SIGMA0_REQ_DEBUG_DUMP (SIGMA0_REQ(DEBUG_DUMP))

  Debug dump.
- #define SIGMA0_REQ_NEW_CLIENT (SIGMA0_REQ(NEW_CLIENT))

  New client.

13.53.1 Detailed Description

Internal sigma0 definitions.

13.54 Internal functions

Collaboration diagram for Internal functions:

Functions

- void base64_encode (const char ∗infile, unsigned int in_size, char ∗∗outfile)

  base-64-encode string infile
- void base64_decode (const char ∗infile, unsigned int in_size, char ∗∗outfile)

  decode base-64-encoded string infile

13.54.1 Detailed Description

13.55 Interrupt controller

The C Icu interface, see L4::Icu for the C++ interface.

Collaboration diagram for Interrupt controller:

Data Structures

- struct l4_icu_info_t

  Info structure for an ICU.

Typedefs

- typedef struct l4_icu_info_t l4_icu_info_t

  Info structure for an ICU.

Enumerations

- enum L4_icu_flags { L4_ICU_FLAG_MSI }

  Flags for IRQ numbers used for the ICU.

Functions

- l4_msgtag_t l4_icu_bind (l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW

  Bind an interrupt line of an interrupt controller to an interrupt object.
- l4_msgtag_t l4_icu_bind_u (l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW

  Bind an interrupt line of an interrupt controller to an interrupt object.
- l4_msgtag_t l4_icu_unbind (l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW

  Remove binding of an interrupt line from the interrupt controller object.
- l4_msgtag_t l4_icu_unbind_u (l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW

  Remove binding of an interrupt line from the interrupt controller object.
- l4_msgtag_t l4_icu_set_mode (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW

  Set interrupt mode.
- l4_msgtag_t l4_icu_set_mode_u (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode, l4_utcb_t *utcb) L4_NOTHROW

  Set interrupt mode.
- l4_msgtag_t l4_icu_info (l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW

  Get information about the ICU features.
- l4_msgtag_t l4_icu_info_u (l4_cap_idx_t icu, l4_icu_info_t *info, l4_utcb_t *utcb) L4_NOTHROW

  Get information about the ICU features.
- l4_msgtag_t l4_icu_msi_info (l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source, l4_icu_msi_info_t *msi_info) L4_NOTHROW

  Get MSI info about IRQ.
- l4_msgtag_t l4_icu_msi_info_u (l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source, l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW

  Get MSI info about IRQ.
- l4_msgtag_t l4_icu_unmask (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to) L4_NOTHROW

  Unmask an IRQ line.
- l4_msgtag_t l4_icu_unmask_u (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW

  Unmask the given interrupt line.
- l4_msgtag_t l4_icu_mask (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to) L4_NOTHROW

  Mask an IRQ line.
- l4_msgtag_t l4_icu_mask_u (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW

  Mask an IRQ line.

13.55.1 Detailed Description

The C Icu interface, see L4::Icu for the C++ interface.

Note

"ICU" is short for "interrupt control unit".

These functions define the interface for interrupt controllers, for binding IRQ objects to interrupt lines and other interrupt sources, as well as functions for masking and unmasking of interrupts.

To setup an IRQ line the following steps are required:

1. l4_icu_set_mode() (optional if IRQ has a default mode)
2. l4_rcv_ep_bind_thread() to attach the IRQ object to a thread
3. l4_icu_bind()
4. l4_icu_unmask() to receive the first IRQ

For certain interrupt sources only some of these steps are necessary and supported, see Scheduler and Virtual Console.

At most one IRQs object can be bound to a certain interrupt source and a certain IRQs object can be bound to at most one interrupt source.

Include File

```
#include <l4/sys/icu.h>
```

13.55.2 Typedef Documentation

13.55.2.1 l4_icu_info_t

```
typedef struct l4_icu_info_t l4_icu_info_t
```

Info structure for an ICU.

This structure contains information about the features of an ICU.

See also

l4_icu_info().

13.55.3 Enumeration Type Documentation

13.55.3.1 L4_icu_flags

```
enum L4_icu_flags
```

Flags for IRQ numbers used for the ICU.

Enumerator

L4_ICU_FLAG_MSI	Flag to denote that the IRQ is actually an MSI. This flag may be used for l4_icu_bind() and l4_icu_unbind() functions to denote that the IRQ number is meant to be an MSI.

Definition at line 63 of file icu.h.

13.55.4 Function Documentation

13.55.4.1 l4_icu_bind()

```
l4_msgtag_t l4_icu_bind (
            l4_cap_idx_t icu,
            unsigned irqnum,
            l4_cap_idx_t irq )  [inline]
```

Bind an interrupt line of an interrupt controller to an interrupt object.

Parameters

icu	ICU object to bind `irq` to.
irqnum	IRQ line at the ICU.
irq	IRQ object to bind to this ICU.

Returns

Syscall return tag. The caller should check the return value using l4_error() to check for errors and to identify the correct method for unmasking the interrupt. Return values $<$ `0` indicate an error. A return value of `0` means a direct unmask via the IRQ object using l4_irq_unmask(). A return value of `1` means that the interrupt has to be unmasked via the ICU using l4_icu_unmask().

Return values

L4_EINVAL	`irq` is bound to an interrupt source.
L4_EPERM	The ICU instance requires L4_CAP_FPAGE_W on `irq` and L4_CAP_FPAGE_W was not present.

In case the `irq` is already bound to an interrupt source, it is unbound first. In case the `irq` is bound and the interrupt source is bound to a different IRQ object, only the unbinding happens. An IRQ object that is bound to an interrupt source will get unbound if the IRQ object is deleted.

Examples

examples/sys/isr/main.c.

Definition at line 493 of file icu.h.

References l4_icu_bind_u(), and l4_utcb().

Here is the call graph for this function:

13.55.4.2 l4_icu_bind_u()

```
l4_msgtag_t l4_icu_bind_u (
            l4_cap_idx_t icu,
            unsigned irqnum,
            l4_cap_idx_t irq,
            l4_utcb_t * utcb )  [inline]
```

Bind an interrupt line of an interrupt controller to an interrupt object.

Parameters

icu	The ICU object to bind `irq` to.
irqnum	IRQ line at the ICU.
irq	IRQ object for the given IRQ line to bind to this ICU.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag. The caller should check the return value using l4_error() to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of 0 means a direct unmask via the IRQ object using L4::Irq::unmask. A return value of 1 means that the interrupt has to be unmasked via the ICU using L4::Icu::unmask.

Return values

L4_EINVAL	`irq` is bound to an interrupt source.
L4_EPERM	The ICU instance requires L4_CAP_FPAGE_W on `irq` and L4_CAP_FPAGE_W was not present.

In case the `irq` is already bound to an interrupt source, it is unbound first. In case the `irq` is bound and the interrupt source is bound to a different L4::Irq object, only the unbinding happens. An L4::Irq object that is bound to an interrupt source will get unbound if the L4::Irq object is deleted.

Definition at line 393 of file icu.h.

Referenced by L4::Icu::bind(), and l4_icu_bind().

Here is the caller graph for this function:

13.55.4.3 l4_icu_info()

```
l4_msgtag_t l4_icu_info (
            l4_cap_idx_t icu,
            l4_icu_info_t * info )  [inline]
```

Get information about the ICU features.

Parameters

	icu	The ICU object from which information shall be retrieved.
out	*info*	Info structure to be filled with information.

Returns

Syscall return tag

Definition at line 501 of file icu.h.

References l4_icu_info_u(), and l4_utcb().

Here is the call graph for this function:

13.55.4.4 l4_icu_info_u()

```
l4_msgtag_t l4_icu_info_u (
            l4_cap_idx_t icu,
            l4_icu_info_t * info,
            l4_utcb_t * utcb )  [inline]
```

Get information about the ICU features.

Parameters

	icu	The ICU object from which MSI information shall be retrieved.
out	*info*	Info structure to be filled with information.
	utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 417 of file icu.h.

Referenced by L4::Icu::info(), and l4_icu_info().

Here is the caller graph for this function:

13.55.4.5 l4_icu_mask()

```
l4_msgtag_t l4_icu_mask (
            l4_cap_idx_t icu,
            unsigned irqnum,
            l4_umword_t * label,
            l4_timeout_t to )  [inline]
```

Mask an IRQ line.

Parameters

icu	The ICU object where the IRQ line 'irqnum' shall be masked.
irqnum	IRQ line at the ICU.
label	If non-NULL the function also waits for the next message.
to	Timeout for message to ICU, if unsure use L4_IPC_NEVER.

Returns

Syscall return tag

Definition at line 515 of file icu.h.

13.55.4.6 l4_icu_mask_u()

```
l4_msgtag_t l4_icu_mask_u (
            l4_cap_idx_t icu,
            unsigned irqnum,
            l4_umword_t * label,
            l4_timeout_t to,
            l4_utcb_t * utcb )  [inline]
```

Mask an IRQ line.

Parameters

icu	The ICU object where the IRQ line 'irqnum' shall be masked.
irqnum	IRQ line at the ICU.
label	If NULL this function is a send-only message to the ICU. If not NULL this function will enter an open wait after sending the mask message.
to	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL `label` only.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 480 of file icu.h.

Referenced by L4::Icu::mask().

Here is the caller graph for this function:

13.55.4.7 l4_icu_msi_info()

```
l4_msgtag_t l4_icu_msi_info (
            l4_cap_idx_t icu,
            unsigned irqnum,
            l4_uint64_t source,
            l4_icu_msi_info_t * msi_info )  [inline]
```

Get MSI info about IRQ.

Parameters

	icu	The ICU object from which MSI information shall be retrieved.
	irqnum	IRQ line at the ICU.
	source	Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification.
out	*msi_info*	A l4_icu_msi_info_t structure receiving the address and the data value to trigger this MSI.

Returns

Syscall return tag

Definition at line 505 of file icu.h.

References l4_icu_msi_info_u(), and l4_utcb().

Here is the call graph for this function:

13.55.4.8 l4_icu_msi_info_u()

```
l4_msgtag_t l4_icu_msi_info_u (
            l4_cap_idx_t icu,
            unsigned irqnum,
            l4_uint64_t source,
            l4_icu_msi_info_t * msi_info,
            l4_utcb_t * utcb )  [inline]
```

Get MSI info about IRQ.

Parameters

	utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.
	icu	The ICU object from which MSI information shall be retrieved.
	irqnum	IRQ line at the ICU.
	source	Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification.
out	*msi_info*	A l4_icu_msi_info_t structure receiving the address and the data value to trigger this MSI.

Returns

Syscall return tag

Definition at line 431 of file icu.h.

Referenced by l4_icu_msi_info().

Here is the caller graph for this function:

13.55.4.9 l4_icu_set_mode()

```
l4_msgtag_t l4_icu_set_mode (
            l4_cap_idx_t icu,
            unsigned irqnum,
            l4_umword_t mode )  [inline]
```

Set interrupt mode.

Parameters

icu	The ICU object.
irqnum	IRQ line at the ICU.
mode	Mode, see L4_irq_mode.

Returns

Syscall return tag

Definition at line 520 of file icu.h.

References l4_icu_set_mode_u(), and l4_utcb().

Here is the call graph for this function:

13.55.4.10 l4_icu_set_mode_u()

```
l4_msgtag_t l4_icu_set_mode_u (
            l4_cap_idx_t icu,
            unsigned irqnum,
            l4_umword_t mode,
            l4_utcb_t * utcb )  [inline]
```

Set interrupt mode.

Parameters

icu	The ICU object.
irqnum	IRQ line at the ICU.
mode	Mode, see L4_irq_mode.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 454 of file icu.h.

Referenced by l4_icu_set_mode(), and L4::Icu::set_mode().

Here is the caller graph for this function:

13.55.4.11 l4_icu_unbind()

```
l4_msgtag_t l4_icu_unbind (
            l4_cap_idx_t icu,
            unsigned irqnum,
            l4_cap_idx_t irq )  [inline]
```

Remove binding of an interrupt line from the interrupt controller object.

Parameters

icu	The ICU object from where the binding shall be removed.
irqnum	IRQ line at the ICU.
irq	IRQ object to remove from the ICU.

Returns

Syscall return tag

Definition at line 497 of file icu.h.

References l4_icu_unbind_u(), and l4_utcb().

Here is the call graph for this function:

13.55.4.12 l4_icu_unbind_u()

```
l4_msgtag_t l4_icu_unbind_u (
            l4_cap_idx_t icu,
            unsigned irqnum,
            l4_cap_idx_t irq,
            l4_utcb_t * utcb )  [inline]
```

Remove binding of an interrupt line from the interrupt controller object.

Parameters

icu	The ICU object from where the binding shall be removed.
irqnum	IRQ line at the ICU.
irq	IRQ object to remove from the ICU.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 405 of file icu.h.

Referenced by l4_icu_unbind(), and L4::Icu::unbind().

Here is the caller graph for this function:

13.55.4.13 l4_icu_unmask()

```
l4_msgtag_t l4_icu_unmask (
            l4_cap_idx_t icu,
            unsigned irqnum,
            l4_umword_t * label,
            l4_timeout_t to )  [inline]
```

Unmask an IRQ line.

Parameters

icu	The ICU object where the IRQ line shall be unmasked.
irqnum	IRQ line at the ICU.
label	If non-NULL the function also waits for the next message.
to	Timeout for message to ICU, if unsure use L4_IPC_NEVER.

Returns

Syscall return tag, the error values therein are undefined because l4_icu_unmask() is a sender-only IPC.

Definition at line 510 of file icu.h.

13.55.4.14 l4_icu_unmask_u()

```
l4_msgtag_t l4_icu_unmask_u (
            l4_cap_idx_t icu,
            unsigned irqnum,
            l4_umword_t * label,
            l4_timeout_t to,
            l4_utcb_t * utcb )  [inline]
```

Unmask the given interrupt line.

Parameters

icu	The ICU object where the IRQ line shall be unmasked. When the object is an IRQ, the given interrupt line is ignored and instead the line which the IRQ is bound to (if any) is unmasked.

Its counterpart for explicitly masking an interrupt line is L4::Icu::mask().

Parameters

	irqnum	The interrupt line that shall be unmasked. Ignored if the object is an IRQ.
`out`	*label*	If NULL this is a send-only unmask, if not NULL then this operation enters an open wait and the *protected label* shall be received here.
	to	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL `label` only.
	utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag.

Note

If `label` is NULL this function is a send-only operation and there is no return value except for a failed send operation. In this case use l4_ipc_error() to check for errors, **do not** use l4_error(), because l4_error() will always return an error.

Definition at line 485 of file icu.h.

13.56 Kernel Debugger

Kernel debugger related functionality.

Collaboration diagram for Kernel Debugger:

Functions

- l4_msgtag_t l4_debugger_set_object_name (l4_cap_idx_t cap, const char *name) L4_NOTHROW

  Set the name of a kernel object.
- unsigned long l4_debugger_global_id (l4_cap_idx_t cap) L4_NOTHROW

  Get the globally unique ID of the object behind a capability.
- unsigned long l4_debugger_kobj_to_id (l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW

  Get the globally unique ID of the object behind the kobject pointer.

13.56.1 Detailed Description

Kernel debugger related functionality.

Attention

This API is subject to change!

This is a debugging facility, any call to any function might be invalid. Do not rely on it in any real code.

Include File

```
#include <l4/sys/debugger.h>
```

13.56.2 Function Documentation

13.56.2.1 l4_debugger_global_id()

```
unsigned long l4_debugger_global_id (
            l4_cap_idx_t cap )  [inline]
```

Get the globally unique ID of the object behind a capability.

Parameters

cap	Capability

Return values

~0UL	Capability is not valid.
>=0	Global debugger id.

This is a debugging facility, the call might be invalid.

Definition at line 346 of file debugger.h.

13.56.2.2 l4_debugger_kobj_to_id()

```
unsigned long l4_debugger_kobj_to_id (
            l4_cap_idx_t cap,
            l4_addr_t kobjp )  [inline]
```

Get the globally unique ID of the object behind the kobject pointer.

Parameters

cap	Capability
kobjp	Kobject pointer

Return values

~0UL	The capability or the kobject pointer are invalid.
>=0	The globally unique id.

This is a debugging facility, the call might be invalid.

Definition at line 352 of file debugger.h.

13.56.2.3 l4_debugger_set_object_name()

```
l4_msgtag_t l4_debugger_set_object_name (
            l4_cap_idx_t cap,
            const char * name )  [inline]
```

Set the name of a kernel object.

Parameters

cap	Capability which refers to the kernel object.
name	Name of the kernel object that is e.g. displayed in the kernel debugger.

This is a debugging facility, the call might be invalid.

Examples

examples/sys/aliens/main.c.

Definition at line 339 of file debugger.h.

13.57 Kernel Interface Page

Kernel Interface Page.

Collaboration diagram for Kernel Interface Page:

Modules

- Fiasco-UX Virtual devices

  Virtual hardware devices, provided by Fiasco-UX.
- Memory descriptors (C version)

  C Interface for KIP memory descriptors.

Data Structures

- struct l4_kernel_info_t

  L4 Kernel Interface Page.
- class L4::Kip::Mem_desc

  Memory descriptors stored in the kernel interface page.

Macros

- #define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /∗ "L4µK" ∗/

  Kernel Info Page identifier ("L4µK").

Typedefs

- typedef struct l4_kernel_info_t l4_kernel_info_t

  L4 Kernel Interface Page.
- typedef struct l4_kernel_info_t l4_kernel_info_t

  L4 Kernel Interface Page.

Enumerations

- enum { L4_KIP_OFFS_READ_US = 0x900 , L4_KIP_OFFS_READ_NS = 0x980 }

Functions

- l4_umword_t l4_kip_version (l4_kernel_info_t ∗kip) L4_NOTHROW

  Get the kernel version.
- const char ∗ l4_kip_version_string (l4_kernel_info_t ∗kip) L4_NOTHROW

  Get the kernel version string.
- int l4_kernel_info_version_offset (l4_kernel_info_t ∗kip) L4_NOTHROW

  Return offset in bytes of version_strings relative to the KIP base.
- l4_cpu_time_t l4_kip_clock (l4_kernel_info_t ∗kip) L4_NOTHROW

  Return clock value from the KIP.
- l4_umword_t l4_kip_clock_lw (l4_kernel_info_t ∗kip) L4_NOTHROW

  Return least significant machine word of clock value from the KIP.
- l4_uint64_t l4_kip_clock_ns (l4_kernel_info_t ∗kip) L4_NOTHROW

  Return current clock using the KIP in nanoseconds.

13.57.1 Detailed Description

Kernel Interface Page.

C interface for the Kernel Interface Page:

C++ interface for the Kernel Interface Page:

Include File

```
#include <l4/sys/kip>
```

Include File

```
#include <l4/sys/kip.h>
```

13.57.2 Enumeration Type Documentation

13.57.2.1 anonymous enum

```
anonymous enum
```

Enumerator

L4_KIP_OFFS_READ_US	Offset of KIP code (provided by the kernel) for reading the KIP clock in microseconds. If the kernel is configured for a fine-grained KIP clock (CONFIG_SYNC_TSC enabled for IA32, ARM_SYNC_CLOCK for ARM), this code provides the KIP clock with microseconds granularity and accuracy by reading the hardware clock used by the kernel and transforming this value into microseconds. Otherwise this code just reads the KIP clock value.
L4_KIP_OFFS_READ_NS	Offset of KIP code (provided by the kernel) for reading the time stamp counter and transforming this value into nanoseconds. If the kernel is configured for fine-grained KIP clock (CONFIG_SYNC enabled for IA32, ARM_SYNC_CLOCK for ARM), this code provides the KIP clock with nanoseconds granularity and accuracy by reading the hardware clock used by the kernel and transforming this value into nanoseconds. Otherwise this code just reads the KIP clock value and multiplies it by 1000.

Definition at line 64 of file kip.h.

13.57.3 Function Documentation

13.57.3.1 l4_kernel_info_version_offset()

```
int l4_kernel_info_version_offset (
            l4_kernel_info_t * kip )  [inline]
```

Return offset in bytes of version_strings relative to the KIP base.

Parameters

kip	Pointer to the kernel info page (KIP).

Returns

offset of version_strings relative to the KIP base address, in bytes.

Definition at line 187 of file kip.h.

Referenced by l4_kip_version_string().

Here is the caller graph for this function:

13.57.3.2 l4_kip_clock()

```
l4_cpu_time_t l4_kip_clock (
            l4_kernel_info_t * kip )  [inline]
```

Return clock value from the KIP.

Parameters

kip	Pointer to the kernel info page (KIP).

Returns

Value of the clock field in the KIP.

The KIP clock always contains the current (relative) time in micro seconds independently of the CPU frequency. The clock is only guaranteed to be accurate within the scheduling granularity announced in the KIP.

This function basically calls the KIP code for reading the KIP clock with microseconds resolution. The accuracy depends on the platform and the kernel configuration.

See also

L4_KIP_OFFS_READ_US.

Definition at line 191 of file kip.h.

References L4_KIP_OFFS_READ_US.

Referenced by L4::Poll_timeout_kipclock::set(), and L4::Poll_timeout_kipclock::test().

Here is the caller graph for this function:

13.57.3.3 l4_kip_clock_lw()

```
l4_umword_t l4_kip_clock_lw (
            l4_kernel_info_t * kip )  [inline]
```

Return least significant machine word of clock value from the KIP.

Parameters

kip	Pointer to the kernel info page (KIP).

Returns

Lower machine word of clock value from the KIP.

This function will always provide the least significant machine word of the clock value from the KIP, regardless of the kernel configuration.

Definition at line 210 of file kip.h.

References l4_mb().

Here is the call graph for this function:

13.57.3.4 l4_kip_clock_ns()

```
l4_cpu_time_t l4_kip_clock_ns (
            l4_kernel_info_t * kip )  [inline]
```

Return current clock using the KIP in nanoseconds.

Parameters

kip	Pointer to the kernel info page (KIP).

Returns

Value of the current clock in nanoseconds.

This function basically calls the KIP code for reading the KIP clock with nanoseconds resolution. The accuracy depends on the platform and the kernel configuration.

See also

L4_KIP_OFFS_READ_NS.

Definition at line 201 of file kip.h.

References L4_KIP_OFFS_READ_NS.

13.57.3.5 l4_kip_version()

```
l4_umword_t l4_kip_version (
            l4_kernel_info_t * kip )  [inline]
```

Get the kernel version.

Parameters

kip	Kernel Info Page.

Returns

Kernel version string. 0 if KIP could not be mapped.

Definition at line 179 of file kip.h.

13.57.3.6 l4_kip_version_string()

```
const char * l4_kip_version_string (
            l4_kernel_info_t * kip )  [inline]
```

Get the kernel version string.

Parameters

kip	Kernel Info Page.

Returns

Kernel version string.

Definition at line 183 of file kip.h.

References l4_kernel_info_version_offset().

Here is the call graph for this function:

13.58 Kernel Interface Page API

Collaboration diagram for Kernel Interface Page API:

Files

- file kip.h

Macros

- #define l4util_kip_for_each_feature(s) for (s += strlen(s) + 1; ∗s; s += strlen(s) + 1)

  Cycle through kernel features given in the KIP.

Functions

- int l4util_kip_kernel_is_ux (l4_kernel_info_t ∗)

  Return whether the kernel is running natively or under UX.
- int l4util_kip_kernel_has_feature (l4_kernel_info_t ∗, const char ∗str)

  Check if kernel supports a feature.
- unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t ∗)

  Return kernel ABI version.

13.58.1 Detailed Description

13.58.2 Macro Definition Documentation

13.58.2.1 l4util_kip_for_each_feature

```
#define l4util_kip_for_each_feature(
              s )  for (s += strlen(s) + 1; *s; s += strlen(s) + 1)
```

Cycle through kernel features given in the KIP.

Cycles through all KIP kernel feature strings. s must be a character pointer (char const ∗) initialized with l4_kip_version_string().

Definition at line 61 of file kip.h.

13.58.3 Function Documentation

13.58.3.1 l4util_kip_kernel_abi_version()

```
unsigned long l4util_kip_kernel_abi_version (
            l4_kernel_info_t *  )
```

Return kernel ABI version.

Returns

Kernel ABI version.

13.58.3.2 l4util_kip_kernel_has_feature()

```
int l4util_kip_kernel_has_feature (
            l4_kernel_info_t * ,
            const char * str )
```

Check if kernel supports a feature.

Parameters

str	Feature name to check.

Returns

1 if the kernel supports the feature, 0 if not.

Checks the feature field in the KIP for the given string.

13.58.3.3 l4util_kip_kernel_is_ux()

```
int l4util_kip_kernel_is_ux (
            l4_kernel_info_t *  )
```

Return whether the kernel is running natively or under UX.

Returns

1 when running under UX, 0 if not running under UX.

Examples

examples/sys/ux-vhw/main.c.

13.59 Kernel Objects

API of kernel objects.

Collaboration diagram for Kernel Objects:

Modules

- DMA space

  A DMA space represents a device memory address space managed by an IOMMU.
- Factory

  C factory interface to create kernel objects, see L4::Factory for the C++ interface.
- IPC-Gate API

  The C IPC gate interface, see L4::Ipc_gate for the C++ interface.
- IRQs

  C IRQ interface, see L4::Irq for the C++ interface.
- Interrupt controller

The C Icu interface, see L4::Icu for the C++ interface.

- Kernel-provided semaphore

  C semaphore interface, see L4::Semaphore for the C++ interface.
- L4 kernel object type information

  Type information for L4 server objects that can be called via IPC.
- Platform Control C API

  C interface for controlling platform-wide properties, see L4::Platform_control for the C++ interface.
- Scheduler

  C interface of the Scheduler kernel object, see L4::Scheduler for the C interface.
- Task

  C interface of the Task kernel object, see L4::Task for the C++ interface.
- Thread

  C Thread object interface, see L4::Thread for the C++ interface.
- Virtual Console

  C Virtual console interface for simple character based input and output, see L4::Vcon for the C++ interface.
- Virtual Machines

  Virtual Machine API.

Data Structures

- class L4::Kobject

  Base class for all kinds of kernel objects and remote objects, referenced by capabilities.
- class L4::Vm

  Virtual machine host address space.

13.59.1 Detailed Description

API of kernel objects.

Include File

```
#include <l4/sys/kernel_object.h>
```

13.60 Kernel-provided semaphore

C semaphore interface, see L4::Semaphore for the C++ interface.

Collaboration diagram for Kernel-provided semaphore:

Functions

- l4_msgtag_t l4_semaphore_up (l4_cap_idx_t sem) L4_NOTHROW

  Semaphore up operation (wrapper for trigger()).
- l4_msgtag_t l4_semaphore_down (l4_cap_idx_t sem, l4_timeout_t timeout) L4_NOTHROW

  Semaphore down operation.

13.60.1 Detailed Description

C semaphore interface, see L4::Semaphore for the C++ interface.

Include File

```
#include <l4/sys/semaphore.h>
```

13.60.2 Function Documentation

13.60.2.1 l4_semaphore_down()

```
l4_msgtag_t l4_semaphore_down (
            l4_cap_idx_t sem,
            l4_timeout_t timeout )  [inline]
```

Semaphore down operation.

Parameters

sem	Semaphore object.
timeout	Timeout for blocking the semaphore down operation. Note: The receive timeout of this timeout-pair is significant for blocking, the send part is usually non-blocking.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag. Use l4_error() to check for errors.

Return values

-L4_EPERM	No L4_CAP_FPAGE_S right on invoked semaphore capability.

This method decrements the semaphore counter by one, or blocks if the counter is already zero, until either a timeout or cancel condition hits or the counter is increased by an `up()` operation.

Definition at line 89 of file semaphore.h.

13.60.2.2 l4_semaphore_up()

```
l4_msgtag_t l4_semaphore_up (
            l4_cap_idx_t sem ) [inline]
```

Semaphore up operation (wrapper for trigger()).

Parameters

sem	Semaphore object.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Send-only IPC message return tag. Use l4_ipc_error() to check for errors, do **not** use l4_error().

Increases the semaphore counter by one if it is smaller than an unspecified limit. The unspecified limit is guaranteed to be at least 2^31-1.

Definition at line 48 of file semaphore.h.

References l4_irq_trigger().

Here is the call graph for this function:

13.61 Kumem allocator utility

Kumem allocator utility C interface.

Collaboration diagram for Kumem allocator utility:

Kumem allocator utility C interface.

13.62 Kumem utiltities

Collaboration diagram for Kumem utiltities:

Functions

- int L4Re::Util::kumem_alloc (l4_addr_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) noexcept

  Allocate state area.

13.62.1 Detailed Description

13.62.2 Function Documentation

13.62.2.1 kumem_alloc()

```
int L4Re::Util::kumem_alloc (
            l4_addr_t * mem,
            unsigned pages_order,
            L4::Cap< L4::Task > task = L4Re::Env::env() ->task(),
            L4::Cap< L4Re::Rm > rm = L4Re::Env::env() ->rm() )  [noexcept]
```

Allocate state area.

Parameters

out	*mem*	Pointer to memory that has been allocated.
	pages_order	Size to allocate, in log2 pages.
	task	Task to use for allocation.
	rm	Region manager to use for allocation.

Return values

0	for success
<0	error code on failure

Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page. A portable implementation should not depend on allocations greater than 16KiB to succeed.

13.63 L4 IPC Opcodes

List of protocol specific opcodes used for communication with L4Re and Kernel objects.

Enumerations

- enum L4_icu_opcode {
  L4_ICU_OP_BIND , L4_ICU_OP_UNBIND , L4_ICU_OP_INFO , L4_ICU_OP_MSI_INFO ,
  L4_ICU_OP_UNMASK , L4_ICU_OP_MASK , L4_ICU_OP_SET_MODE }

  Opcodes to the ICU interface.
- enum L4_ipc_gate_ops { L4_IPC_GATE_BIND_OP = 0x10 , L4_IPC_GATE_GET_INFO_OP = 0x11 }

  Operations on the IPC-gate.
- enum L4_platform_ctl_ops {
  L4_PLATFORM_CTL_SYS_SUSPEND_OP = 0UL , L4_PLATFORM_CTL_SYS_SHUTDOWN_OP = 1UL ,
  L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP = 2UL , L4_PLATFORM_CTL_CPU_ENABLE_OP = 3UL ,
  L4_PLATFORM_CTL_CPU_DISABLE_OP = 4UL }

  Operations on platform-control objects.
- enum L4_task_ops {
  L4_TASK_MAP_OP = 0UL , L4_TASK_UNMAP_OP = 1UL , L4_TASK_CAP_INFO_OP = 2UL ,
  L4_TASK_ADD_KU_MEM_OP = 3UL ,
  L4_TASK_LDT_SET_X86_OP = 0x11UL , L4_TASK_MAP_VGICC_ARM_OP = 0x12UL }

  Operations on task objects.
- enum L4_thread_ops {
  L4_THREAD_CONTROL_OP = 0UL , L4_THREAD_EX_REGS_OP = 1UL , L4_THREAD_SWITCH_OP = 2UL , L4_THREAD_STATS_OP = 3UL ,
  L4_THREAD_VCPU_RESUME_OP = 4UL , L4_THREAD_REGISTER_DELETE_IRQ_OP = 5UL ,
  L4_THREAD_MODIFY_SENDER_OP = 6UL , L4_THREAD_VCPU_CONTROL_OP = 7UL ,
  L4_THREAD_VCPU_CONTROL_EXT_OP = L4_THREAD_VCPU_CONTROL_OP | 0x10000 , L4_THREAD_X86_GDT_OP = 0x10UL , L4_THREAD_ARM_TPIDRURO_OP = 0x10UL , L4_THREAD_AMD64_SET_SEGMENT_BASE_OP = 0x12UL ,
  L4_THREAD_AMD64_GET_SEGMENT_INFO_OP = 0x13UL , L4_THREAD_OPCODE_MASK = 0xffff }

  Operations on thread objects.
- enum L4_vcon_ops { L4_VCON_WRITE_OP = 0UL , L4_VCON_READ_OP = 1UL , L4_VCON_SET_ATTR_OP = 2UL , L4_VCON_GET_ATTR_OP = 3UL }

  Operations on vcon objects.

13.63.1 Detailed Description

List of protocol specific opcodes used for communication with L4Re and Kernel objects.

13.63.2 Enumeration Type Documentation

13.63.2.1 L4_icu_opcode

`enum L4_icu_opcode`

Opcodes to the ICU interface.

Enumerator

L4_ICU_OP_BIND	Bind opcode. **See also** l4_icu_bind()
L4_ICU_OP_UNBIND	Unbind opcode. **See also** l4_icu_unbind()
L4_ICU_OP_INFO	Info opcode. **See also** l4_icu_info()
L4_ICU_OP_MSI_INFO	Msi-info opcode. **See also** l4_icu_msi_info()
L4_ICU_OP_UNMASK	Unmask opcode. **See also** l4_icu_unmask()
L4_ICU_OP_MASK	Mask opcode. **See also** l4_icu_mask()
L4_ICU_OP_SET_MODE	Set-mode opcode. **See also** l4_icu_set_mode()

Definition at line 106 of file icu.h.

13.63.2.2 L4_ipc_gate_ops

`enum L4_ipc_gate_ops`

Operations on the IPC-gate.

Enumerator

L4_IPC_GATE_BIND_OP	Bind operation.
L4_IPC_GATE_GET_INFO_OP	Info operation.

Definition at line 103 of file ipc_gate.h.

13.63.2.3 L4_platform_ctl_ops

`enum L4_platform_ctl_ops`

Operations on platform-control objects.

See L4_PROTO_PLATFORM_CTL for the protocol type to use for messages to platform-control objects.

Enumerator

L4_PLATFORM_CTL_SYS_SUSPEND_OP	Suspend.
L4_PLATFORM_CTL_SYS_SHUTDOWN_OP	shutdown/reboot
L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP	allow CPU shutdown
L4_PLATFORM_CTL_CPU_ENABLE_OP	enable an offline CPU
L4_PLATFORM_CTL_CPU_DISABLE_OP	disable an online CPU

Definition at line 166 of file platform_control.h.

13.63.2.4 L4_task_ops

`enum L4_task_ops`

Operations on task objects.

Enumerator

L4_TASK_MAP_OP	Map.
L4_TASK_UNMAP_OP	Unmap.
L4_TASK_CAP_INFO_OP	Cap info.
L4_TASK_ADD_KU_MEM_OP	Add kernel-user memory.
L4_TASK_LDT_SET_X86_OP	x86: LDT set
L4_TASK_MAP_VGICC_ARM_OP	Arm: Map virtual GICC area.

Definition at line 269 of file task.h.

13.63.2.5 L4_thread_ops

```
enum L4_thread_ops
```

Operations on thread objects.

Enumerator

L4_THREAD_CONTROL_OP	Control operation.
L4_THREAD_EX_REGS_OP	Exchange registers operation.
L4_THREAD_SWITCH_OP	Do a thread switch.
L4_THREAD_STATS_OP	Thread statistics.
L4_THREAD_VCPU_RESUME_OP	VCPU resume.
L4_THREAD_REGISTER_DELETE_IRQ_OP	Register an IPC-gate deletion IRQ.
L4_THREAD_MODIFY_SENDER_OP	Modify all senders IDs that match the given pattern.
L4_THREAD_VCPU_CONTROL_OP	Enable / disable VCPU feature.
L4_THREAD_X86_GDT_OP	Gdt.
L4_THREAD_ARM_TPIDRURO_OP	Set TPIDRURO register.
L4_THREAD_AMD64_SET_SEGMENT_BASE_OP	Set segment base.
L4_THREAD_AMD64_GET_SEGMENT_INFO_OP	Get segment information.
L4_THREAD_OPCODE_MASK	Mask for opcodes.

Definition at line 673 of file thread.h.

13.63.2.6 L4_vcon_ops

```
enum L4_vcon_ops
```

Operations on vcon objects.

Enumerator

L4_VCON_WRITE_OP	Write.
L4_VCON_READ_OP	Read.
L4_VCON_SET_ATTR_OP	Get console attributes.
L4_VCON_GET_ATTR_OP	Set console attributes.

Definition at line 293 of file vcon.h.

13.64 L4 VIRTIO Block Device

Collaboration diagram for L4 VIRTIO Block Device:

Data Structures

- struct l4virtio_block_header_t

  Header structure of a request for a block device.
- struct l4virtio_block_discard_t

  Structure used for the write zeroes and discard commands.
- struct l4virtio_block_config_t

  Device configuration for block devices.

Typedefs

- typedef struct l4virtio_block_header_t l4virtio_block_header_t

  Header structure of a request for a block device.
- typedef struct l4virtio_block_discard_t l4virtio_block_discard_t

  Structure used for the write zeroes and discard commands.
- typedef struct l4virtio_block_config_t l4virtio_block_config_t

  Device configuration for block devices.

Enumerations

- enum L4virtio_block_operations {
  L4VIRTIO_BLOCK_T_IN = 0 , L4VIRTIO_BLOCK_T_OUT = 1 , L4VIRTIO_BLOCK_T_FLUSH = 4 , L4VIRTIO_BLOCK_T_GET_ID = 8 ,
  L4VIRTIO_BLOCK_T_DISCARD = 11 , L4VIRTIO_BLOCK_T_WRITE_ZEROES = 13 }

  Kinds of operation over a block device.
- enum L4virtio_block_status { L4VIRTIO_BLOCK_S_OK = 0 , L4VIRTIO_BLOCK_S_IOERR = 1 , L4VIRTIO_BLOCK_S_UNSUPP = 2 }

  Status of a finished block request.

13.64.1 Detailed Description

13.64.2 Enumeration Type Documentation

13.64.2.1 L4virtio_block_operations

`enum L4virtio_block_operations`

Kinds of operation over a block device.

Enumerator

L4VIRTIO_BLOCK_T_IN	Read from device.
L4VIRTIO_BLOCK_T_OUT	Write to device.
L4VIRTIO_BLOCK_T_FLUSH	Flush data to disk.
L4VIRTIO_BLOCK_T_GET_ID	Get device ID.
L4VIRTIO_BLOCK_T_DISCARD	Discard a range of sectors.
L4VIRTIO_BLOCK_T_WRITE_ZEROES	Write zeroes to a range of sectors.

Definition at line 31 of file virtio_block.h.

13.64.2.2 L4virtio_block_status

```
enum L4virtio_block_status
```

Status of a finished block request.

Enumerator

L4VIRTIO_BLOCK_S_OK	Request finished successfully.
L4VIRTIO_BLOCK_S_IOERR	IO error on device.
L4VIRTIO_BLOCK_S_UNSUPP	Operation is not supported.

Definition at line 44 of file virtio_block.h.

13.65 L4 VIRTIO Input Device

Collaboration diagram for L4 VIRTIO Input Device:

Data Structures

- struct l4virtio_input_absinfo_t

  Information about the absolute axis in the underlying evdev implementation.
- struct l4virtio_input_devids_t

  Device ID information for the device.
- struct l4virtio_input_config_t

  Device configuration for input devices.
- struct l4virtio_input_event_t

  Single event in event or status queue.

Typedefs

- typedef struct l4virtio_input_absinfo_t l4virtio_absinfo_t

  Information about the absolute axis in the underlying evdev implementation.
- typedef struct l4virtio_input_devids_t l4virtio_input_devids_t

  Device ID information for the device.
- typedef struct l4virtio_input_config_t l4virtio_input_config_t

  Device configuration for input devices.
- typedef struct l4virtio_input_event_t l4virtio_input_event_t

  Single event in event or status queue.

Enumerations

- enum L4virtio_input_config_select

  Device information selectors.

13.65.1 Detailed Description

13.66 L4 VIRTIO Interface

Collaboration diagram for L4 VIRTIO Interface:

Modules

- L4 VIRTIO Block Device
- L4 VIRTIO Input Device
- L4 VIRTIO Transport Layer

  L4 specific VIRTIO Transport layer.

13.66.1 Detailed Description

13.67 L4 VIRTIO Transport Layer

L4 specific VIRTIO Transport layer.

Collaboration diagram for L4 VIRTIO Transport Layer:

Data Structures

- struct l4virtio_config_hdr_t

  L4-VIRTIO config header, provided in shared data space.
- struct l4virtio_config_queue_t

  Queue configuration entry.

Typedefs

- typedef struct l4virtio_config_hdr_t l4virtio_config_hdr_t

  L4-VIRTIO config header, provided in shared data space.
- typedef struct l4virtio_config_queue_t l4virtio_config_queue_t

  Queue configuration entry.

Enumerations

- enum L4_virtio_protocol

  L4-VIRTIO protocol number.
- enum L4_virtio_opcodes { L4VIRTIO_OP_SET_STATUS = 0 , L4VIRTIO_OP_CONFIG_QUEUE , L4VIRTIO_OP_REGISTER_IFACE , L4VIRTIO_OP_REGISTER_DS }

  L4-VIRTIO opcodes.
- enum L4virtio_device_ids {
  L4VIRTIO_ID_NET = 1 , L4VIRTIO_ID_BLOCK = 2 , L4VIRTIO_ID_CONSOLE = 3 , L4VIRTIO_ID_RNG = 4 ,
  L4VIRTIO_ID_BALLOON = 5 , L4VIRTIO_ID_RPMSG = 7 , L4VIRTIO_ID_SCSI = 8 , L4VIRTIO_ID_9P = 9 , L4VIRTIO_ID_RPROC_SERIAL = 11 , L4VIRTIO_ID_CAIF = 12 , L4VIRTIO_ID_GPU = 16 , L4VIRTIO_ID_INPUT = 18 ,
  L4VIRTIO_ID_VSOCK = 19 , L4VIRTIO_ID_CRYPTO = 20 , L4VIRTIO_ID_SOCK = 0x9999 }

  Virtio device IDs as reported in the driver's config space.

- enum L4virtio_device_status { L4VIRTIO_STATUS_ACKNOWLEDGE = 1 , L4VIRTIO_STATUS_DRIVER = 2 , L4VIRTIO_STATUS_DRIVER_OK = 4 , L4VIRTIO_STATUS_FEATURES_OK = 8 , L4VIRTIO_STATUS_DEVICE_NEEDS_RESET = 0x40 , L4VIRTIO_STATUS_FAILED = 0x80 }

  Virtio device status bits.
- enum L4virtio_feature_bits { L4VIRTIO_FEATURE_VERSION_1 = 32 , L4VIRTIO_FEATURE_CMD_CONFIG = 224 }

  L4virtio-specific feature bits.
- enum L4_virtio_irq_status { L4VIRTIO_IRQ_STATUS_VRING = 1 , L4VIRTIO_IRQ_STATUS_CONFIG = 2 }

  VIRTIO IRQ status codes (l4virtio_config_hdr_t::irq_status).
- enum L4_virtio_cmd { L4VIRTIO_CMD_NONE = 0x00000000 , L4VIRTIO_CMD_SET_STATUS = 0x01000000 , L4VIRTIO_CMD_CFG_QUEUE = 0x02000000 , L4VIRTIO_CMD_MASK = 0xff000000 }

  Virtio commands for device configuration.

Functions

- l4virtio_config_queue_t ∗ l4virtio_config_queues (l4virtio_config_hdr_t const ∗cfg)

  Get the pointer to the first queue config.
- void ∗ l4virtio_device_config (l4virtio_config_hdr_t const ∗cfg)

  Get the pointer to the device configuration.
- void l4virtio_set_feature (l4_uint32_t ∗feature_map, unsigned feat)

  Set the given feature bit in a feature map.
- void l4virtio_clear_feature (l4_uint32_t ∗feature_map, unsigned feat)

  Clear the given feature bit in a feature map.
- unsigned l4virtio_get_feature (l4_uint32_t ∗feature_map, unsigned feat)

  Check if the given bit in a feature map is set.
- int l4virtio_set_status (l4_cap_idx_t cap, unsigned status) L4_NOTHROW

  Write the VIRTIO status register.
- int l4virtio_config_queue (l4_cap_idx_t cap, unsigned queue) L4_NOTHROW

  Trigger queue configuration of the given queue.
- int l4virtio_register_ds (l4_cap_idx_t cap, l4_cap_idx_t ds_cap, l4_uint64_t base, l4_umword_t offset, l4_umword_t size) L4_NOTHROW

  Register a shared data space with VIRTIO host.
- int l4virtio_register_iface (l4_cap_idx_t cap, l4_cap_idx_t guest_irq, l4_cap_idx_t host_irq, l4_cap_idx_t config_ds) L4_NOTHROW

  Register client to the L4-VIRTIO device.
- int l4virtio_device_config_ds (l4_cap_idx_t cap, l4_cap_idx_t config_ds, l4_addr_t ∗ds_offset) L4_NOTHROW

  Get the dataspace with the L4virtio configuration page.
- int l4virtio_device_notification_irq (l4_cap_idx_t cap, unsigned index, l4_cap_idx_t irq) L4_NOTHROW

  Get the notification interrupt corresponding to the given index.

13.67.1 Detailed Description

L4 specific VIRTIO Transport layer.

The L4 specific VIRTIO Transport layer is based on L4Re::Dataspace as shared memory and L4::Irq for signaling. The VIRTIO configuration space is mostly based on a shared memory implementation too and accompanied by two IPC functions to synchronize the configuration between device and driver.

13.67.2 Typedef Documentation

13.67.2.1 l4virtio_config_queue_t

```
typedef struct l4virtio_config_queue_t l4virtio_config_queue_t
```

Queue configuration entry.

An array of such entries is available at the l4virtio_config_hdr_t::queues_offset in the config data space.

Consistency rules for the queue config are:

- A driver might read `num_max` at any time.
- A driver must write to `num`, `desc_addr`, `avail_addr`, and `used_addr` only when `ready` is zero (0). Values in these fields are validated and used by the device only after successfully setting `ready` to one (1), either by the IPC or by L4VIRTIO_CMD_CFG_QUEUE.
- The value of `device_notify_index` is valid only when `ready` is one.
- The driver might write to `device_notify_index` at any time, however the change is guaranteed to take effect after a successful L4VIRTIO_CMD_CFG_QUEUE or after a config_queue IPC. Note, the change might also have immediate effect, depending on the device implementation.

13.67.3 Enumeration Type Documentation

13.67.3.1 L4_virtio_cmd

```
enum L4_virtio_cmd
```

Virtio commands for device configuration.

Enumerator

L4VIRTIO_CMD_NONE	No command pending.
L4VIRTIO_CMD_SET_STATUS	Set the status register.
L4VIRTIO_CMD_CFG_QUEUE	Configure a queue.
L4VIRTIO_CMD_MASK	Mask to get command bits.

Definition at line 111 of file virtio.h.

13.67.3.2 L4_virtio_irq_status

```
enum L4_virtio_irq_status
```

VIRTIO IRQ status codes (l4virtio_config_hdr_t::irq_status).

Note

l4virtio_config_hdr_t::irq_status is currently unused.

Enumerator

L4VIRTIO_IRQ_STATUS_VRING	VRING IRQ pending flag.
L4VIRTIO_IRQ_STATUS_CONFIG	CONFIG IRQ pending flag.

Definition at line 102 of file virtio.h.

13.67.3.3 L4_virtio_opcodes

`enum L4_virtio_opcodes`

L4-VIRTIO opcodes.

Enumerator

L4VIRTIO_OP_SET_STATUS	Set status register in device config.
L4VIRTIO_OP_CONFIG_QUEUE	Set queue config in device config.
L4VIRTIO_OP_REGISTER_IFACE	Register a transport driver to the device.
L4VIRTIO_OP_REGISTER_DS	Register a data space as transport memory.

Definition at line 49 of file virtio.h.

13.67.3.4 L4virtio_device_ids

`enum L4virtio_device_ids`

Virtio device IDs as reported in the driver's config space.

Enumerator

L4VIRTIO_ID_NET	Virtual ethernet card.
L4VIRTIO_ID_BLOCK	General block device.
L4VIRTIO_ID_CONSOLE	Simple device for data IO via ports.
L4VIRTIO_ID_RNG	Entropy source.
L4VIRTIO_ID_BALLOON	Memory ballooning device.
L4VIRTIO_ID_RPMSG	Device using rpmsg protocol.
L4VIRTIO_ID_SCSI	SCSI host device.
L4VIRTIO_ID_9P	Device using 9P transport protocol.

Enumerator

L4VIRTIO_ID_RPROC_SERIAL	Rproc serial device.
L4VIRTIO_ID_CAIF	Device using CAIF network protocol.
L4VIRTIO_ID_GPU	GPU.
L4VIRTIO_ID_INPUT	Input.
L4VIRTIO_ID_VSOCK	Vsock transport.
L4VIRTIO_ID_CRYPTO	Crypto.
L4VIRTIO_ID_SOCK	Unofficial socket device.

Definition at line 58 of file virtio.h.

13.67.3.5 L4virtio_device_status

`enum L4virtio_device_status`

Virtio device status bits.

Enumerator

L4VIRTIO_STATUS_ACKNOWLEDGE	Guest OS has found device.
L4VIRTIO_STATUS_DRIVER	Guest OS knows how to drive device.
L4VIRTIO_STATUS_DRIVER_OK	Driver is set up.
L4VIRTIO_STATUS_FEATURES_OK	Driver has acknowledged feature set.
L4VIRTIO_STATUS_DEVICE_NEEDS_RESET	Device detected fatal error.
L4VIRTIO_STATUS_FAILED	Driver detected fatal error.

Definition at line 79 of file virtio.h.

13.67.3.6 L4virtio_feature_bits

`enum L4virtio_feature_bits`

L4virtio-specific feature bits.

Enumerator

L4VIRTIO_FEATURE_VERSION_1	Virtio protocol version 1 supported. Must be 1 for L4virtio.
L4VIRTIO_FEATURE_CMD_CONFIG	Status and queue config are set via cmd field instead of via IPC.

Definition at line 90 of file virtio.h.

13.67.4 Function Documentation

13.67.4.1 l4virtio_config_queue()

```
int l4virtio_config_queue (
            l4_cap_idx_t cap,
            unsigned queue )
```

Trigger queue configuration of the given queue.

Parameters

cap	Capability to the VIRTIO host.

Usually all queues are configured when the status is written to running. However, in some cases queues shall be disabled or enabled dynamically, in this case this function triggers a reconfiguration from the shared memory register of the queue config.

Parameters

queue	Queue index for the queue to be configured.

Return values

0	on success.
-L4_EIO	The queue's status is invalid.
-L4_ERANGE	The queue index exceeds the number of queues.
-L4_EINVAL	Otherwise.

13.67.4.2 l4virtio_config_queues()

```
l4virtio_config_queue_t* l4virtio_config_queues (
            l4virtio_config_hdr_t const * cfg )  [inline]
```

Get the pointer to the first queue config.

Parameters

cfg	Pointer to the config header.

Returns

pointer to queue config of queue 0.

Definition at line 232 of file virtio.h.

References l4virtio_config_hdr_t::queues_offset.

13.67.4.3 l4virtio_device_config()

```
void* l4virtio_device_config (
            l4virtio_config_hdr_t const * cfg )  [inline]
```

Get the pointer to the device configuration.

Parameters

cfg	Pointer to the config header.

Returns

pointer to device configuration structure.

Definition at line 243 of file virtio.h.

13.67.4.4 l4virtio_device_config_ds()

```
int l4virtio_device_config_ds (
            l4_cap_idx_t cap,
            l4_cap_idx_t config_ds,
            l4_addr_t * ds_offset )
```

Get the dataspace with the L4virtio configuration page.

Parameters

cap	Capability to the L4-VIRTIO host
config_ds	Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space.
ds_offset	Offset into the dataspace where the device configuration structure starts.

13.67.4.5 l4virtio_device_notification_irq()

```
int l4virtio_device_notification_irq (
            l4_cap_idx_t cap,
            unsigned index,
            l4_cap_idx_t irq )
```

Get the notification interrupt corresponding to the given index.

Parameters

	cap	Capability to the L4-VIRTIO host
	index	Index of the interrupt.
out	*irq*	Triggerable for the given index.

Return values

L4_EOK	Success.
L4_ENOSYS	IRQ notification not supported by device.
<0	Other error.

An index is only guaranteed to return an IRQ object when the index is set in one of the device notify index fields. The device must return the same interrupt for a given index as long as the index is in use. If an index disappears as a result of a configuration change and then is reused later, the interrupt is not guaranteed to be the same.

Interrupts must always be rerequested after a device reset.

13.67.4.6 l4virtio_register_ds()

```
int l4virtio_register_ds (
            l4_cap_idx_t cap,
            l4_cap_idx_t ds_cap,
            l4_uint64_t base,
            l4_umword_t offset,
            l4_umword_t size )
```

Register a shared data space with VIRTIO host.

Parameters

cap	Capability to the VIRTIO host
ds_cap	Dataspace capability to register. The lower 8 bits determine the rights mask with which the guest's rights are masked during the registration of the dataspace at the VIRTIO host.
base	VIRTIO guest physical start address of shared memory region
offset	Offset within the data space that is attached to the given `base` in the guest physical memory.
size	Size of the memory region in the guest

Returns

0 on success, < 0 on error

13.67.4.7 l4virtio_register_iface()

```
int l4virtio_register_iface (
            l4_cap_idx_t cap,
```

```
            l4_cap_idx_t guest_irq,
            l4_cap_idx_t host_irq,
            l4_cap_idx_t config_ds )
```

Register client to the L4-VIRTIO device.

Parameters

cap	Capability to the L4-VIRTIO host
guest_irq	IRQ capability for valid IRQ object for device-to-driver notifications.
host_irq	Capability selector for receiving the driver-to-device notifications IRQ capability.
config_ds	Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space.

Return values

L4_EOK	Success.
-L4_ENOSYS	This interface is no longer supported by the server. Use get_device_config() etc. instead.

Deprecated Use device_config(), device_notification_irq() and Icu::bind() instead.

13.67.4.8 l4virtio_set_status()

```
int l4virtio_set_status (
            l4_cap_idx_t cap,
            unsigned status )
```

Write the VIRTIO status register.

Parameters

cap	Capability to the VIRTIO host
status	Status word to write to the VIRTIO status.

Return values

0	on success.

Note

All other registers are accessed via shared memory.

13.68 L4 Vbus functions

C interface of the Vbus API.

Collaboration diagram for L4 Vbus functions:

Modules

- L4vbus GPIO functions
- L4vbus PCI functions

Enumerations

- enum L4vbus_dma_domain_assign_flags { L4VBUS_DMAD_UNBIND = 0 , L4VBUS_DMAD_BIND = 1 , L4VBUS_DMAD_L4RE_DMA_SPACE = 0 , L4VBUS_DMAD_KERNEL_DMA_SPACE = 2 }

  Flags for l4vbus_assign_dma_domain().

Functions

- int l4vbus_get_device_by_hid (l4_cap_idx_t vbus, l4vbus_device_handle_t parent, l4vbus_device_handle_t ∗child, char const ∗hid, int depth, l4vbus_device_t ∗devinfo)

  Find a device by the hardware interface identifier (HID).
- int l4vbus_get_next_device (l4_cap_idx_t vbus, l4vbus_device_handle_t parent, l4vbus_device_handle_t ∗child, int depth, l4vbus_device_t ∗devinfo)

  Find next child following `child`.
- int l4vbus_get_device (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, l4vbus_device_t ∗devinfo)

  Obtain detailed information about a Vbus device.
- int l4vbus_get_resource (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, int res_idx, l4vbus_resource_t ∗res)

  Obtain the resource description of an individual device resource.
- int l4vbus_is_compatible (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char const ∗cid)

  Check if the given device has a compatibility ID (CID) or HID that matches cid.
- int l4vbus_get_hid (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char ∗hid, unsigned long max_len)

  Get the HID (hardware identifier) of a device.
- int l4vbus_request_ioport (l4_cap_idx_t vbus, l4vbus_resource_t const ∗res)

  Request an IO port resource.
- int l4vbus_request_resource (l4_cap_idx_t vbus, l4vbus_resource_t const ∗res, int flags)

  Request a resource of a specific type.
- int l4vbus_assign_dma_domain (l4_cap_idx_t vbus, unsigned domain_id, unsigned flags, l4_cap_idx_t dma_space)

  Bind or unbind a kernel DMA space (L4::Task) or a L4Re::Dma_space to a DMA domain.
- int l4vbus_release_ioport (l4_cap_idx_t vbus, l4vbus_resource_t const ∗res)

Release a previously requested IO port resource.

- int l4vbus_release_resource (l4_cap_idx_t vbus, l4vbus_resource_t const *res)

  Release a previously requested resource.

- int l4vbus_vicu_get_cap (l4_cap_idx_t vbus, l4vbus_device_handle_t icu, l4_cap_idx_t cap)

  Get capability of ICU.

13.68.1 Detailed Description

C interface of the Vbus API.

The virtual bus (Vbus) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an Icu (Interrupt controller) for interrupt handling.

The Vbus interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

Include File

```
#include <l4/vbus/vbus.h>
```

Refer to L4vbus for the C++ API.

13.68.2 Enumeration Type Documentation

13.68.2.1 L4vbus_dma_domain_assign_flags

```
enum L4vbus_dma_domain_assign_flags
```

Flags for l4vbus_assign_dma_domain().

Enumerator

L4VBUS_DMAD_UNBIND	Unbind the given DMA space from the DMA domain.
L4VBUS_DMAD_BIND	Bind the given DMA space to the DMA domain.
L4VBUS_DMAD_L4RE_DMA_SPACE	The given DMA space is an L4Re::Dma_space.
L4VBUS_DMAD_KERNEL_DMA_SPACE	The given DMA space is a kernel DMA space (L4::Task)

Definition at line 183 of file vbus.h.

13.68.3 Function Documentation

13.68.3.1 l4vbus_assign_dma_domain()

```
int l4vbus_assign_dma_domain (
            l4_cap_idx_t vbus,
            unsigned domain_id,
            unsigned flags,
            l4_cap_idx_t dma_space )
```

Bind or unbind a kernel DMA space (L4::Task) or a L4Re::Dma_space to a DMA domain.

Parameters

vbus	Capability of the system bus
domain_id	DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.
flags	A combination of L4vbus_dma_domain_assign_flags.
dma_space	The DMA space capability to bind or unbind, this must either be an L4Re::Dma_space or a kernel DMA space (L4::Task created with L4_PROTO_DMA_SPACE) and the type must be reflected in the `flags`.

Return values

0	Operation completed successfully.
-L4_ENOENT	The vbus does not support a global DMA domain or no DMA domain could be found.
-L4_EINVAL	Invalid argument used.
-L4_EBUSY	DMA domain is already active, this means another DMA space is already assigned.

Referenced by L4vbus::Vbus::assign_dma_domain().

Here is the caller graph for this function:

13.68.3.2 l4vbus_get_device()

```
int l4vbus_get_device (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t dev,
            l4vbus_device_t * devinfo )
```

Obtain detailed information about a Vbus device.

Parameters

	vbus	Capability of the vbus to which the device is connected.
	dev	Device handle of the device from which to retrieve the details.
`out`	*devinfo*	Information structure which contains details about the device. The pointer might be NULL.

Return values

0	Success.
-L4_ENODEV	No device with the given device handle `dev` could be found.

Referenced by L4vbus::Device::device().

Here is the caller graph for this function:

13.68.3.3 l4vbus_get_device_by_hid()

```
int l4vbus_get_device_by_hid (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t parent,
            l4vbus_device_handle_t * child,
            char const * hid,
            int depth,
            l4vbus_device_t * devinfo )
```

Find a device by the hardware interface identifier (HID).

Parameters

vbus	Capability of the system bus
parent	Handle to the parent to start the search

This function searches the vbus for a device with the given HID and returns a handle to the first matching device. The HID usually conforms to an ACPI HID or a Linux device tree compatible identifier.

It is possible to have multiple devices with the same HID on a vbus. In order to find all matching devices this function has to be called repeatedly with `child` pointing to the device found in the previous iteration. The iteration starts at `child` that might be any device node in the tree.

Parameters

in,out	*child*	Handle of the device from where in the device tree the search should start. To start searching from the beginning `child` must be initialized using the default (L4VBUS_NULL). If a matching device is found its handle is returned through this parameter.
	hid	HID of the device
	depth	Maximum depth for the recursive lookup
out	*devinfo*	Device information structure (might be NULL)

Return values

>=0	A device with the given HID was found.
-L4_ENOENT	No device with the given HID could be found on the vbus.
-L4_EINVAL	Invalid or no HID provided.
-L4_ENODEV	Function called on a non-existing device.

13.68.3.4 l4vbus_get_hid()

```
int l4vbus_get_hid (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t dev,
            char * hid,
            unsigned long max_len )
```

Get the HID (hardware identifier) of a device.

Parameters

vbus	Capability of the system bus
dev	Handle of the device
hid	Pointer to a buffer for the HID string
max_len	The size of the buffer (*hid*)

Returns

the length of the HID string on success, else failure

13.68.3.5 l4vbus_get_next_device()

```
int l4vbus_get_next_device (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t parent,
            l4vbus_device_handle_t * child,
```

```
int depth,
l4vbus_device_t * devinfo )
```

Find next child following `child`.

Parameters

	vbus	Capability of the system bus
	parent	Handle to the parent device (use L4VBUS_ROOT_BUS for the system bus)
in,out	*child*	Handle of the device that precedes the device that shall be returned. To start from the beginning, `child` must be initialized with L4VBUS_NULL. If a device is found, its handle is returned through this parameter.
	depth	Depth to look for
out	*devinfo*	Device information (might be NULL)

Returns

0 on success, else failure

13.68.3.6 l4vbus_get_resource()

```
int l4vbus_get_resource (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t dev,
            int res_idx,
            l4vbus_resource_t * res )
```

Obtain the resource description of an individual device resource.

Parameters

	vbus	Capability of the vbus to which the device is connected.
	dev	Device handle of the device on the vbus. The device handle can be obtained by using the l4vbus_get_device_by_hid() and l4vbus_get_next_device() functions.
	res_idx	Index of the resource for which the resource description should be returned. The total number of resources for a device is available in the l4vbus_device_t structure that is returned by L4vbus::Device::device_by_hid() and L4vbus::Device::next_device().
out	*res*	Descriptor of the resource.

This function returns the resource descriptor of an individual device resource selected by the `res_idx` parameter.

Return values

0	Success.
-L4_ENOENT	Invalid resource index `res_idx`.

Referenced by L4vbus::Device::get_resource().

Here is the caller graph for this function:

13.68.3.7 l4vbus_is_compatible()

```
int l4vbus_is_compatible (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t dev,
            char const * cid )
```

Check if the given device has a compatibility ID (CID) or HID that matches *cid*.

Parameters

vbus	Capability of the system bus
dev	device handle for which the CID shall be tested
cid	the compatibility ID to test

Returns

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

Referenced by L4vbus::Device::is_compatible().

Here is the caller graph for this function:

13.68.3.8 l4vbus_release_ioport()

```
int l4vbus_release_ioport (
            l4_cap_idx_t vbus,
            l4vbus_resource_t const * res )
```

Release a previously requested IO port resource.

Parameters

	vbus	Capability of the system bus.
in	*res*	The IO port resource to be released from the bus.

Returns

>=0 on success, <0 on error.

Referenced by L4vbus::Vbus::release_ioport(), and L4vbus::Vbus::release_resource().

Here is the caller graph for this function:

13.68.3.9 l4vbus_release_resource()

```
int l4vbus_release_resource (
            l4_cap_idx_t vbus,
            l4vbus_resource_t const * res )
```

Release a previously requested resource.

Parameters

	vbus	Capability of the system bus.
in	*res*	The resource to be released from the bus.

Returns

>=0 on success, <0 on error.

Deprecated This function is deprecated since Q3 2019. Use l4vbus_release_ioport() instead.

13.68.3.10 l4vbus_request_ioport()

```
int l4vbus_request_ioport (
            l4_cap_idx_t vbus,
            l4vbus_resource_t const * res )
```

Request an IO port resource.

Parameters

	vbus	Capability of the system bus.
in	*res*	The IO port resource to be requested from the bus.

Return values

0	Success.
-L4_EINVAL	Resource is not an IO port resource.
-L4_ENOENT	No matching IO port resource found.

If any IO port resource is found that contains the requested IO port range the IO ports are obtained.

Referenced by L4vbus::Vbus::request_ioport(), and L4vbus::Vbus::request_resource().

Here is the caller graph for this function:

13.68.3.11 l4vbus_request_resource()

```
int l4vbus_request_resource (
            l4_cap_idx_t vbus,
            l4vbus_resource_t const * res,
            int flags )
```

Request a resource of a specific type.

Parameters

	vbus	Capability of the system bus.
in	*res*	Descriptor of the resource.
	flags	Ignored.

Returns

0 on success, else failure.

If any resource is found that contains the requested type and resource address range the resource is obtained.

Deprecated This function is deprecated since Q3 2019. Use l4vbus_request_ioport() instead.

13.68.3.12 l4vbus_vicu_get_cap()

```
int l4vbus_vicu_get_cap (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t icu,
            l4_cap_idx_t cap )
```

Get capability of ICU.

Parameters

vbus	Capability of the system bus.
icu	ICU device handle.
cap	Capability slot for the capability.

Returns

0 on success, else failure

Referenced by L4vbus::Icu::vicu().

Here is the caller graph for this function:

13.69 L4 kernel object type information

Type information for L4 server objects that can be called via IPC.

Collaboration diagram for L4 kernel object type information:

Data Structures

- struct L4::Type_info

  Dynamic Type Information for L4Re Interfaces.
- struct L4::Kobject_typeid< T >

  Meta object for handling access to type information of Kobjects.
- class L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >

  Helper class to create an L4Re interface class that is derived from a single base class.
- class L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >

  Helper class to create an L4Re interface class that is derived from two base classes (see `L4::Kobject_t`).
- struct L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >

  Helper class to create an L4Re interface class that is derived from three base classes (see `L4::Kobject_t`).
- struct L4::Kobject_x< Derived, ARGS >

  Generic Kobject inheritance template.

Functions

- template<typename T >
  Type_info const * L4::kobject_typeid () noexcept

  Get the L4::Type_info for the L4Re interface given in `T`.

13.69.1 Detailed Description

Type information for L4 server objects that can be called via IPC.

This type information consists of inheritance information, the protocol number assigned to an interface as well as the demand on server-side resources.

13.69.2 Function Documentation

13.69.2.1 kobject_typeid()

```
template<typename T >
Type_info const* L4::kobject_typeid ( )  [inline], [noexcept]
```

Get the L4::Type_info for the L4Re interface given in `T`.

Template Parameters

T	The type (L4Re interface) for which the information shall be returned.

Returns

A pointer to the L4::Type_info structure for `T`.

Definition at line 692 of file __typeinfo.h.

References L4::Kobject_typeid< T >::id().

Here is the call graph for this function:

13.70 L4Re C Interface

Documentation for the L4Re C Interface.

Collaboration diagram for L4Re C Interface:

Modules

- Capability allocator

  Capability allocator C interface.
- DMA Space Interface

  DMA Space C interface.
- Dataspace interface

Dataspace C interface.

- Debug interface
- Event interface

  Event C interface.
- Initial Environment

  C interface of the initial environment that is provided to an L4 task.
- Kumem allocator utility

  Kumem allocator utility C interface.
- L4Re Util C Interface

  Documentation of the L4 Runtime Environment utility functionality in C.
- Log interface

  Log C interface.
- Memory allocator

  Memory allocator C interface.
- Namespace interface

  Namespace C interface.
- Region map interface

  Region map C interface.
- Video API

Functions

- long l4re_inhibitor_acquire (l4_cap_idx_t cap, l4_umword_t id, char const ∗reason)

  Inhibitor C interface.

13.70.1 Detailed Description

Documentation for the L4Re C Interface.

The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

13.70.2 Function Documentation

13.70.2.1 l4re_inhibitor_acquire()

```
long l4re_inhibitor_acquire (
            l4_cap_idx_t cap,
            l4_umword_t id,
            char const * reason )
```

Inhibitor C interface.

Acquire an inhibitor lock.

Parameters

cap	Capability for the Inhibitor object (

See also

L4Re::Inhibitor)

Parameters

id	ID of the inhibitor lock that shall be acquired.
reason	Reason why the inhibitor lock is acquired. (Used for informing the user or debugging.)

Returns

0 for success, <0 on error

See also

L4Re::Inhibitor::acquire()

13.71 L4Re C++ Interface

Documentation of the L4 Runtime Environment C++ API.

Collaboration diagram for L4Re C++ Interface:

Modules

- Auxiliary data
- C++ Exceptions
- Console API

  Console interface.
- Debugging API

  Debugging Interface.

- Event API

  Event API.
- L4Re ELF Auxiliary Information

  API for embedding auxiliary information into binary programs.
- L4Re Protocol identifiers

  Fix L4Re Protocol Constants.
- L4Re Util C++ Interface

  Documentation of the L4 Runtime Environment utility functionality in C++.
- Logging interface

  Interface for log output.
- Name-space API

  API for name spaces that store capabilities.
- Parent API

  Parent interface.
- Region map API

  Virtual address-space management.
- Vbus API

  C++ interface of the Vbus API.

13.71.1 Detailed Description

Documentation of the L4 Runtime Environment C++ API.

13.72 L4Re Capability API

Collaboration diagram for L4Re Capability API:

Data Structures

- class L4Re::Cap_alloc

  Capability allocator interface.
- class L4Re::Smart_cap_auto< Unmap_flags >

  Helper for Unique_cap and Unique_del_cap.
- class L4Re::Smart_count_cap< Unmap_flags >

  Helper for Ref_cap and Ref_del_cap.
- class L4Re::Util::Smart_cap_auto< Unmap_flags >

  Helper for Unique_cap and Unique_del_cap.
- class L4Re::Util::Smart_count_cap< Unmap_flags >

  Helper for Ref_cap and Ref_del_cap.
- struct L4Re::Util::Ref_cap< T >

  Automatic capability that implements automatic free and unmap of the capability selector.
- struct L4Re::Util::Ref_del_cap< T >

  Automatic capability that implements automatic free and unmap+delete of the capability selector.

Functions

- template<typename T >
  Ref_cap< T >::Cap L4Re::Util::make_ref_cap ()

  Allocate a capability slot and wrap it in a Ref_cap.
- template<typename T >
  Ref_del_cap< T >::Cap L4Re::Util::make_ref_del_cap ()

  Allocate a capability slot and wrap it in a Ref_del_cap.
- virtual L4Re::Cap_alloc::~Cap_alloc ()=0

  Destructor.

Variables

- _Cap_alloc & L4Re::Util::cap_alloc

  Capability allocator.

13.72.1 Detailed Description

13.72.2 Function Documentation

13.72.2.1 make_ref_cap()

```
template<typename T >
Ref_cap<T>::Cap L4Re::Util::make_ref_cap ( )
```

Allocate a capability slot and wrap it in a Ref_cap.

Template Parameters

T	Type of capability the slot is used for.

Definition at line 218 of file cap_alloc.

References L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc(), and L4Re::Util::cap_alloc.

Here is the call graph for this function:

13.72.2.2 make_ref_del_cap()

```
template<typename T >
Ref_del_cap<T>::Cap L4Re::Util::make_ref_del_cap ( )
```

Allocate a capability slot and wrap it in a Ref_del_cap.

Template Parameters

T	Type of capability the slot is used for.

Definition at line 227 of file cap_alloc.

References L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc(), and L4Re::Util::cap_alloc.

Here is the call graph for this function:

13.72.3 Variable Documentation

13.72.3.1 cap_alloc

```
_Cap_alloc& L4Re::Util::cap_alloc  [extern]
```

Capability allocator.

This is the instance of the capability allocator that is used by usual applications. The actual implementation of the allocator depends on the configuration of the system.

Per default we use Counting_cap_alloc, a reference-counting capability allocator, that keeps a reference counter for each managed capability selector.

Note

This capability allocator is not thread-safe.

Examples

examples/libs/l4re/c++/mem_alloc/ma+rm.cc, examples/libs/l4re/c++/shared_ds/ds_clnt.cc, examples/libs/l4re/c++/shared_ds/d and examples/libs/l4re/streammap/client.cc.

Referenced by L4Re::Util::Br_manager::alloc_buffer_demand(), L4Re::Util::Smart_count_cap< Unmap_flags >::copy(), L4Re::Util::Smart_cap_auto< Unmap_flags >::free(), L4Re::Util::Smart_count_cap< Unmap_flags >::free(), L4Re::Util::make_ref_cap(), L4Re::Util::make_ref_del_cap(), L4Re::Util::make_shared_cap(), L4Re::Util::make_shared_del_cap(), L4Re::Util::make_unique_cap(), L4Re::Util::make_unique_del_cap(), L4Re::Util::Br_manager::realloc_rcv_cap(), and L4Re::Util::Object_registry::unregister_obj().

13.73 L4Re ELF Auxiliary Information

API for embedding auxiliary information into binary programs.

Collaboration diagram for L4Re ELF Auxiliary Information:

Data Structures

- struct l4re_elf_aux_t

  Generic header for each auxiliary vector element.
- struct l4re_elf_aux_vma_t

  Auxiliary vector element for a reserved virtual memory area.
- struct l4re_elf_aux_mword_t

  Auxiliary vector element for a single unsigned data word.

Macros

- #define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".rol4re_elf_aux"), aligned(sizeof(l4_umword_t))))

  Define an auxiliary vector element.
- #define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...) static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}

  Define an auxiliary vector element.

Typedefs

- typedef struct l4re_elf_aux_t l4re_elf_aux_t

  Generic header for each auxiliary vector element.
- typedef struct l4re_elf_aux_vma_t l4re_elf_aux_vma_t

  Auxiliary vector element for a reserved virtual memory area.
- typedef struct l4re_elf_aux_mword_t l4re_elf_aux_mword_t

  Auxiliary vector element for a single unsigned data word.

Enumerations

- enum {
  L4RE_ELF_AUX_T_NONE = 0 , L4RE_ELF_AUX_T_VMA , L4RE_ELF_AUX_T_STACK_SIZE , L4RE_ELF_AUX_T_STACK_ADDR ,
  L4RE_ELF_AUX_T_KIP_ADDR }

13.73.1 Detailed Description

API for embedding auxiliary information into binary programs.

This API allows information for the binary loader to be embedded into a binary application. This information can be reserved areas in the virtual memory of an application and things such as the stack size to be allocated for the first application thread.

13.73.2 Macro Definition Documentation

13.73.2.1 L4RE_ELF_AUX_ELEM

```
#define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".rol4re_elf_aux"), aligned(sizeof(l4_umword_t)))
```

Define an auxiliary vector element.

This is the generic method for defining auxiliary vector elements. A more convenient way is to use L4RE_ELF_↩ AUX_ELEM_T.

Usage:

```
L4RE_ELF_AUX_ELEM l4re_elf_aux_vma_t decl_name =
  { L4RE_ELF_AUX_T_VMA, sizeof(l4re_elf_aux_vma_t), 0x2000, 0x4000 };
```

Definition at line 52 of file elf_aux.h.

13.73.2.2 L4RE_ELF_AUX_ELEM_T

```
#define L4RE_ELF_AUX_ELEM_T(
              type,
              id,
              tag,
              val...  )   static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}
```

Define an auxiliary vector element.

Parameters

type	is the data type for the element (e.g., l4re_elf_aux_vma_t)
id	is the identifier (variable name) for the declaration (the variable is defined with `static` storage class)
tag	is the tag value for the element e.g., L4RE_ELF_AUX_T_VMA
val	are the values to be set in the descriptor

Usage:

```
L4RE_ELF_AUX_ELEM_T(l4re_elf_aux_vma_t, decl_name, L4RE_ELF_AUX_T_VMA, 0x2000, 0x4000 };
```

Definition at line 67 of file elf_aux.h.

13.73.3 Enumeration Type Documentation

13.73.3.1 anonymous enum

```
anonymous enum
```

Enumerator

L4RE_ELF_AUX_T_NONE	Tag for an invalid element in the auxiliary vector.
L4RE_ELF_AUX_T_VMA	Tag for descriptor for a reserved virtual memory area.
L4RE_ELF_AUX_T_STACK_SIZE	Tag for descriptor that defines the stack size for the first application thread.
L4RE_ELF_AUX_T_STACK_ADDR	Tag for descriptor that defines the stack address for the first application thread.
L4RE_ELF_AUX_T_KIP_ADDR	Tag for descriptor that defines the KIP address for the binaries address space.

Definition at line 70 of file elf_aux.h.

13.74 L4Re Protocol identifiers

Fix L4Re Protocol Constants.

Collaboration diagram for L4Re Protocol identifiers:

Enumerations

- enum L4Re::Dataspace_::Opcodes

  Data-space communication-protocol opcodes.
- enum L4Re::Event_::Opcodes

  Event communication-protocol opcodes.
- enum L4Re::Inhibitor_::Opcodes

  Inhibitor communication-protocol opcodes.
- enum L4Re::Log_::Opcodes

  Logging-service communication-protocol opcodes.
- enum L4Re::Mem_alloc_::Opcodes

Memory-allocator communication-protocol opcodes.

- enum L4Re::Namespace_::Opcodes

  Name-space communication-protocol opcodes.
- enum L4Re::Parent_::Opcodes

  Parent communication-protocol opcodes.
- enum L4Re::Rm_::Opcodes

  Region-map communication-protocol opcodes.
- enum L4Re::Video::Goos_::Opcodes

  Frame buffer communication-protocol opcodes.

13.74.1 Detailed Description

Fix L4Re Protocol Constants.

13.75 L4Re Util C Interface

Documentation of the L4 Runtime Environment utility functionality in C.

Collaboration diagram for L4Re Util C Interface:

Documentation of the L4 Runtime Environment utility functionality in C.

The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

13.76 L4Re Util C++ Interface

Documentation of the L4 Runtime Environment utility functionality in C++.

Collaboration diagram for L4Re Util C++ Interface:

Modules

- Kumem utiltities
- L4Re Capability API

Data Structures

- class L4Re::Smart_cap_auto< Unmap_flags >

  Helper for Unique_cap and Unique_del_cap.
- class L4Re::Util::Cap_alloc_base

  Capability allocator.
- class L4Re::Util::Br_manager

  Buffer-register (BR) manager for L4::Server.
- class L4Re::Util::Counting_cap_alloc< COUNTERTYPE >

  Internal reference-counting cap allocator.
- class L4Re::Util::Event_buffer_t< PAYLOAD >

  Event_buffer utility class.
- class L4Re::Util::Event_buffer_consumer_t< PAYLOAD >

  An event buffer consumer.
- class L4Re::Util::Vcon_svr< SVR >

  Console server template class.
- class L4Re::Util::Video::Goos_svr

  Goos server class.

13.76.1 Detailed Description

Documentation of the L4 Runtime Environment utility functionality in C++.

13.77 L4vbus GPIO functions

Collaboration diagram for L4vbus GPIO functions:

Enumerations

- enum L4vbus_gpio_generic_func { L4VBUS_GPIO_SETUP_INPUT = 0x100 , L4VBUS_GPIO_SETUP_OUTPUT = 0x200 , L4VBUS_GPIO_SETUP_IRQ = 0x300 }

  Constants for generic GPIO functions.
- enum L4vbus_gpio_pull_modes { L4VBUS_GPIO_PIN_PULL_NONE = 0x100 , L4VBUS_GPIO_PIN_PULL_UP = 0x200 , L4VBUS_GPIO_PIN_PULL_DOWN = 0x300 }

  Constants for generic GPIO pull up/down resistor configuration.

Functions

- int l4vbus_gpio_setup (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned pin, unsigned mode, int value)

  Configure the function of a GPIO pin.
- int l4vbus_gpio_config_pull (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned pin, unsigned mode)

  Generic function to set pull up/down mode.
- int l4vbus_gpio_config_pad (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned pin, unsigned func, unsigned value)

  Hardware specific configuration function.
- int l4vbus_gpio_config_get (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned pin, unsigned func, unsigned *value)

  Read hardware specific configuration.
- int l4vbus_gpio_get (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned pin)

  Read value of GPIO input pin.
- int l4vbus_gpio_set (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned pin, int value)

  Set GPIO output pin.
- int l4vbus_gpio_multi_setup (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned offset, unsigned mask, unsigned mode, unsigned value)

  Configure function of multiple GPIO pins at once.
- int l4vbus_gpio_multi_config_pad (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned offset, unsigned mask, unsigned func, unsigned value)

  Hardware specific configuration function for multiple GPIO pins.
- int l4vbus_gpio_multi_get (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned offset, unsigned *data)

  Read values of multiple GPIO pins at once.
- int l4vbus_gpio_multi_set (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned offset, unsigned mask, unsigned data)

  Set multiple GPIO output pins at once.
- int l4vbus_gpio_to_irq (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned pin)

  Create IRQ for GPIO pin.

13.77.1 Detailed Description

13.77.2 Enumeration Type Documentation

13.77.2.1 L4vbus_gpio_generic_func

`enum L4vbus_gpio_generic_func`

Constants for generic GPIO functions.

Enumerator

L4VBUS_GPIO_SETUP_INPUT	Set GPIO pin to input.
L4VBUS_GPIO_SETUP_OUTPUT	Set GPIO pin to output.
L4VBUS_GPIO_SETUP_IRQ	Set GPIO pin to IRQ.

Definition at line 26 of file vbus_gpio.h.

13.77.2.2 L4vbus_gpio_pull_modes

```
enum L4vbus_gpio_pull_modes
```

Constants for generic GPIO pull up/down resistor configuration.

Enumerator

L4VBUS_GPIO_PIN_PULL_NONE	No pull up or pull down resistors.
L4VBUS_GPIO_PIN_PULL_UP	enable pull up resistor
L4VBUS_GPIO_PIN_PULL_DOWN	enable pull down resistor

Definition at line 36 of file vbus_gpio.h.

13.77.3 Function Documentation

13.77.3.1 l4vbus_gpio_config_get()

```
int l4vbus_gpio_config_get (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            unsigned pin,
            unsigned func,
            unsigned * value )
```

Read hardware specific configuration.

Parameters

	vbus	V-BUS capability
	handle	Device handle for the GPIO chip
	pin	GPIO pin number
	func	Hardware specific configuration register to read from. Usually this is an offset to the GPIO chip's base address.
out	*value*	The configuration value.

Returns

0 if OK, error code otherwise

Referenced by L4vbus::Gpio_pin::config_get().

Here is the caller graph for this function:

13.77.3.2 l4vbus_gpio_config_pad()

```
int l4vbus_gpio_config_pad (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            unsigned pin,
            unsigned func,
            unsigned value )
```

Hardware specific configuration function.

Parameters

vbus	V-BUS capability
handle	Device handle for the GPIO chip
pin	GPIO pin number
func	Hardware specific configuration register, usually offset to the GPIO chip's base address
value	Value which is written into the hardware specific configuration register for the specified pin

Returns

0 if OK, error code otherwise

Referenced by L4vbus::Gpio_pin::config_pad().

Here is the caller graph for this function:

13.77.3.3 l4vbus_gpio_config_pull()

```
int l4vbus_gpio_config_pull (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            unsigned pin,
            unsigned mode )
```

Generic function to set pull up/down mode.

Parameters

vbus	V-BUS capability
handle	Device handle for the GPIO chip
pin	GPIO pin number
mode	mode for pull up/down resistors, see L4vbus_gpio_pull_modes

Returns

0 if OK, error code otherwise

Referenced by L4vbus::Gpio_pin::config_pull().

Here is the caller graph for this function:

13.77.3.4 l4vbus_gpio_get()

```
int l4vbus_gpio_get (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            unsigned pin )
```

Read value of GPIO input pin.

Parameters

vbus	V-BUS capability
handle	Device handle for the GPIO chip
pin	GPIO pin number to read from

Returns

Value of GPIO pin (usually 0 or 1), negative error code otherwise.

Referenced by L4vbus::Gpio_pin::get().

Here is the caller graph for this function:

13.77.3.5 l4vbus_gpio_multi_config_pad()

```
int l4vbus_gpio_multi_config_pad (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            unsigned offset,
            unsigned mask,
            unsigned func,
            unsigned value )
```

Hardware specific configuration function for multiple GPIO pins.

Parameters

vbus	V-BUS capability
handle	Device handle for the GPIO chip
offset	Pin corresponding to the LSB in *mask*. Note: allowed may be hardware specific.
mask	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
func	Hardware specific configuration register, usually offset to the GPIO chip's base address.
value	Value which is written into the hardware specific configuration register for the specified pins

Returns

0 if OK, error code otherwise

Referenced by L4vbus::Gpio_module::config_pad().

Here is the caller graph for this function:

13.77.3.6 l4vbus_gpio_multi_get()

```
int l4vbus_gpio_multi_get (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            unsigned offset,
            unsigned * data )
```

Read values of multiple GPIO pins at once.

Parameters

	vbus	V-BUS capability
	handle	Device handle for the GPIO chip
	offset	Pin corresponding to the LSB in *data*. Note: allowed may be hardware specific.
out	*data*	Each bit returns the value (0 or 1) for the corresponding GPIO pin. The value of pins that are not accessible is undefined.

Returns

0 if OK, error code otherwise

Referenced by L4vbus::Gpio_module::get().

Here is the caller graph for this function:

13.77.3.7 l4vbus_gpio_multi_set()

```
int l4vbus_gpio_multi_set (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            unsigned offset,
            unsigned mask,
            unsigned data )
```

Set multiple GPIO output pins at once.

Parameters

vbus	V-BUS capability
handle	Device handle for the GPIO chip
offset	Pin corresponding to the LSB in *data*. Note: allowed may be hardware specific.
mask	Mask of GPIO pins to set. A bit set to 1 selects this pin. A maximum of 32 pins can be set at once. The real number depends on the hardware and the driver implementation.
data	Each bit corresponds to the GPIO pin in *mask*. The value of each bit is written to the GPIO pin if its bit in *mask* is set.

Returns

0 if OK, error code otherwise

Referenced by L4vbus::Gpio_module::set().

Here is the caller graph for this function:

13.77.3.8 l4vbus_gpio_multi_setup()

```
int l4vbus_gpio_multi_setup (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            unsigned offset,
            unsigned mask,
            unsigned mode,
            unsigned value )
```

Configure function of multiple GPIO pins at once.

Parameters

vbus	V-BUS capability
handle	Device handle for the GPIO chip
offset	Pin corresponding to the LSB in *mask*. Note: allowed may be hardware specific.
mask	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
mode	GPIO function, see L4vbus_gpio_generic_func for generic functions. Hardware specific functions must be provided in the lower 8 bits.
value	Optional value to set the GPIO pins to if they are configured as output pins

Returns

0 if OK, error code otherwise

Referenced by L4vbus::Gpio_module::setup().

Here is the caller graph for this function:

13.77.3.9 l4vbus_gpio_set()

```
int l4vbus_gpio_set (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            unsigned pin,
            int value )
```

Set GPIO output pin.

Parameters

vbus	V-BUS capability
handle	Device handle for the GPIO chip
pin	GPIO pin number to write to
value	Value to write to the GPIO pin (usually 0 or 1)

Returns

0 if OK, error code otherwise

Referenced by L4vbus::Gpio_pin::set().

Here is the caller graph for this function:

13.77.3.10 l4vbus_gpio_setup()

```
int l4vbus_gpio_setup (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            unsigned pin,
            unsigned mode,
            int value )
```

Configure the function of a GPIO pin.

Parameters

vbus	V-BUS capability
handle	Device handle for the GPIO chip
pin	GPIO pin number
mode	GPIO function, see L4vbus_gpio_generic_func for generic functions. Hardware specific functions must be provided in the lower 8 bits.
value	Optional value to set the GPIO pin to if it is configured as an output pin

Returns

0 if OK, error code otherwise

Referenced by L4vbus::Gpio_pin::setup().

Here is the caller graph for this function:

13.77.3.11 l4vbus_gpio_to_irq()

```
int l4vbus_gpio_to_irq (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            unsigned pin )
```

Create IRQ for GPIO pin.

Parameters

vbus	V-BUS capability
handle	Device handle for the GPIO chip
pin	GPIO pin to create an IRQ for.

Returns

IRQ number if OK, negative error code otherwise

Referenced by L4vbus::Gpio_pin::to_irq().

Here is the caller graph for this function:

13.78 L4vbus PCI functions

Collaboration diagram for L4vbus PCI functions:

Functions

- int l4vbus_pci_cfg_read (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width)

  Read from the vPCI configuration space using the PCI root bridge.
- int l4vbus_pci_cfg_write (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width)

  Write to the vPCI configuration space using the PCI root bridge.
- int l4vbus_pci_irq_enable (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, int pin, unsigned char *trigger, unsigned char *polarity)

  Enable PCI interrupt for a specific device using the PCI root bridge.
- int l4vbus_pcidev_cfg_read (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width)

  Read from the device's vPCI configuration space.
- int l4vbus_pcidev_cfg_write (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width)

  Write to the device's vPCI configuration space.
- int l4vbus_pcidev_irq_enable (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned char *trigger, unsigned char *polarity)

  Enable the device's PCI interrupt.

13.78.1 Detailed Description

13.78.2 Function Documentation

13.78.2.1 l4vbus_pci_cfg_read()

```
int l4vbus_pci_cfg_read (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            l4_uint32_t bus,
            l4_uint32_t devfn,
            l4_uint32_t reg,
            l4_uint32_t * value,
            l4_uint32_t width )
```

Read from the vPCI configuration space using the PCI root bridge.

Parameters

	vbus	Capability of the system bus
	handle	Device handle of the PCI root bridge
	bus	Bus number
	devfn	Device id (upper 16bit) and function (lower 16bit)
	reg	Register in configuration space to read
`out`	*value*	Value that has been read
	width	Width to read in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Referenced by L4vbus::Pci_host_bridge::cfg_read().

Here is the caller graph for this function:

13.78.2.2 l4vbus_pci_cfg_write()

```
int l4vbus_pci_cfg_write (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            l4_uint32_t bus,
            l4_uint32_t devfn,
            l4_uint32_t reg,
            l4_uint32_t value,
            l4_uint32_t width )
```

Write to the vPCI configuration space using the PCI root bridge.

Parameters

vbus	Capability of the system bus
handle	Device handle of the PCI root bridge
bus	Bus number
devfn	Device id (upper 16bit) and function (lower 16bit)
reg	Register in configuration space to write
value	Value to write
width	Width to write in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Referenced by L4vbus::Pci_host_bridge::cfg_write().

Here is the caller graph for this function:

13.78.2.3 l4vbus_pci_irq_enable()

```
int l4vbus_pci_irq_enable (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            l4_uint32_t bus,
            l4_uint32_t devfn,
            int pin,
            unsigned char * trigger,
            unsigned char * polarity )
```

Enable PCI interrupt for a specific device using the PCI root bridge.

Parameters

	vbus	Capability of the system bus
	handle	Device handle of the PCI root bridge
	bus	Bus number
	devfn	Device id (upper 16bit) and function (lower 16bit)
	pin	Interrupt pin (normally as reported in configuration register INTR)
out	*trigger*	False if interrupt is level-triggered
out	*polarity*	True if interrupt is of low polarity

Returns

On success: Interrupt line to be used, else failure

Referenced by L4vbus::Pci_host_bridge::irq_enable().

Here is the caller graph for this function:

13.78.2.4 l4vbus_pcidev_cfg_read()

```
int l4vbus_pcidev_cfg_read (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            l4_uint32_t reg,
            l4_uint32_t * value,
            l4_uint32_t width )
```

Read from the device's vPCI configuration space.

Parameters

	vbus	Capability of the system bus
	handle	Device handle of the PCI device
	reg	Register in configuration space to read
out	*value*	Value that has been read
	width	Width to read in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Referenced by L4vbus::Pci_dev::cfg_read().

Here is the caller graph for this function:

13.78.2.5 l4vbus_pcidev_cfg_write()

```
int l4vbus_pcidev_cfg_write (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            l4_uint32_t reg,
            l4_uint32_t value,
            l4_uint32_t width )
```

Write to the device's vPCI configuration space.

Parameters

vbus	Capability of the system bus
handle	Device handle of the PCI device
reg	Register in configuration space to write
value	Value to write
width	Width to write in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Referenced by L4vbus::Pci_dev::cfg_write().

Here is the caller graph for this function:

13.78.2.6 l4vbus_pcidev_irq_enable()

```
int l4vbus_pcidev_irq_enable (
            l4_cap_idx_t vbus,
            l4vbus_device_handle_t handle,
            unsigned char * trigger,
            unsigned char * polarity )
```

Enable the device's PCI interrupt.

Parameters

	vbus	Capability of the system bus
	handle	Device handle of the PCI device
`out`	*trigger*	False if interrupt is level-triggered
`out`	*polarity*	True if interrupt is of low polarity

Returns

On success: Interrupt line to be used, else failure

Referenced by L4vbus::Pci_dev::irq_enable().

Here is the caller graph for this function:

13.79 Log interface

Log C interface.

Collaboration diagram for Log interface:

Functions

- void l4re_log_print (char const ∗string) L4_NOTHROW

  Write a null terminated string to the default log.
- void l4re_log_printn (char const ∗string, int len) L4_NOTHROW

  Write a string of a given length to the default log.
- void l4re_log_print_srv (const l4_cap_idx_t logcap, char const ∗string) L4_NOTHROW

  Write a null terminated string to a log.
- void l4re_log_printn_srv (const l4_cap_idx_t logcap, char const ∗string, int len) L4_NOTHROW

  Write a string of a given length to a log.

13.79.1 Detailed Description

Log C interface.

13.79.2 Function Documentation

13.79.2.1 l4re_log_print()

```
void l4re_log_print (
            char const * string )  [inline]
```

Write a null terminated string to the default log.

Parameters

string	Text to print, null terminated.

Returns

0 for success, <0 on error

See also

L4Re::Log::print

Definition at line 99 of file log.h.

References l4re_log_print_srv().

Here is the call graph for this function:

13.79.2.2 l4re_log_print_srv()

```
void l4re_log_print_srv (
            const l4_cap_idx_t logcap,
            char const * string )
```

Write a null terminated string to a log.

Parameters

logcap	Log capability (service).
string	Text to print, null terminated.

Returns

0 for success, <0 on error

See also

L4Re::Log::print

Referenced by l4re_log_print().

Here is the caller graph for this function:

13.79.2.3 l4re_log_printn()

```
void l4re_log_printn (
            char const * string,
            int len )  [inline]
```

Write a string of a given length to the default log.

Parameters

string	Text to print, null terminated.
len	Length of string in bytes.

Returns

0 for success, <0 on error

See also

L4Re::Log::printn

Definition at line 105 of file log.h.

References l4re_log_printn_srv().

Here is the call graph for this function:

13.79.2.4 l4re_log_printn_srv()

```
void l4re_log_printn_srv (
            const l4_cap_idx_t logcap,
            char const * string,
            int len )
```

Write a string of a given length to a log.

Parameters

logcap	Log capability (service).
string	Text to print, null terminated.
len	Length of string in bytes.

Returns

0 for success, <0 on error

See also

L4Re::Log::printn

Referenced by l4re_log_printn().

Here is the caller graph for this function:

13.80 Logging interface

Interface for log output.

Collaboration diagram for Logging interface:

Data Structures

- class L4Re::Log

  Log interface class.

13.80.1 Detailed Description

Interface for log output.

The logging interface provides a facility sending log output. One purpose of the interface is to serialize the output and provide the possibility to tag output sent to a specific log object.

13.81 Low-Level Thread Functions

Collaboration diagram for Low-Level Thread Functions:

13.82 Memory allocator

Memory allocator C interface.

Collaboration diagram for Memory allocator:

Enumerations

- enum l4re_ma_flags

  Flags for requesting memory at the memory allocator.

Functions

- long l4re_ma_alloc (long size, l4re_ds_t const mem, unsigned long flags) L4_NOTHROW

  Allocate memory.
- long l4re_ma_alloc_align (long size, l4re_ds_t const mem, unsigned long flags, unsigned long align) L4_NOTHROW

  Allocate memory.
- long l4re_ma_alloc_align_srv (l4_cap_idx_t srv, long size, l4re_ds_t const mem, unsigned long flags, unsigned long align) L4_NOTHROW

  Allocate memory.

13.82.1 Detailed Description

Memory allocator C interface.

13.82.2 Enumeration Type Documentation

13.82.2.1 l4re_ma_flags

`enum l4re_ma_flags`

Flags for requesting memory at the memory allocator.

See also

L4Re::Mem_alloc::Mem_alloc_flags

Definition at line 42 of file mem_alloc.h.

13.82.3 Function Documentation

13.82.3.1 l4re_ma_alloc()

```
long l4re_ma_alloc (
            long size,
            l4re_ds_t const mem,
            unsigned long flags )  [inline]
```

Allocate memory.

Parameters

size	Size in bytes to be requested. Allocation granularity is (super)pages, however, the allocator will store the byte-granular given size as the size of the dataspace and consecutively will use this byte-granular size for servicing the dataspace. Allocators may optionally also implement a maximum allocation strategy: if `size` is a negative value and `flags` set the Mem_alloc_flags::Continuous bit, the allocator tries to allocate as much memory as possible leaving an amount of at least `-size` bytes within the associated quota.
mem	Capability slot where the capability to the dataspace is received.
flags	Special dataspace properties, see l4re_ma_flags

Return values

0	Success
-L4_ERANGE	Given size not supported.
-L4_ENOMEM	Not enough memory available.
<0	IPC error

See also

L4Re::Mem_alloc::alloc

The memory allocator returns a dataspace.

Note

This function is using the L4Re::Env::env()->mem_alloc() service.

Examples

examples/libs/l4re/c/ma+rm.c.

Definition at line 146 of file mem_alloc.h.

References l4re_ma_alloc_align_srv().

Here is the call graph for this function:

13.82.3.2 l4re_ma_alloc_align()

```
long l4re_ma_alloc_align (
            long size,
            l4re_ds_t const mem,
            unsigned long flags,
            unsigned long align )  [inline]
```

Allocate memory.

Parameters

size	Size in bytes to be requested. Allocation granularity is (super)pages, however, the allocator will store the byte-granular given size as the size of the dataspace and consecutively will use this byte-granular size for servicing the dataspace. Allocators may optionally also implement a maximum allocation strategy: if `size` is a negative value and `flags` set the Mem_alloc_flags::Continuous bit, the allocator tries to allocate as much memory as possible leaving an amount of at least `-size` bytes within the associated quota.
mem	Capability slot where the capability to the dataspace is received.
flags	Special dataspace properties, see l4re_ma_flags
align	Log2 alignment of dataspace if supported by allocator, will be at least L4_PAGESHIFT, with Super_pages flag set at least L4_SUPERPAGESHIFT

Return values

0	Success
-L4_ERANGE	Given size not supported.
-L4_ENOMEM	Not enough memory available.
<0	IPC error

See also

L4Re::Mem_alloc::alloc and

l4re_ma_alloc

The memory allocator returns a dataspace.

Note

This function is using the L4Re::Env::env()->mem_alloc() service.

Definition at line 154 of file mem_alloc.h.

References l4re_ma_alloc_align_srv().

Here is the call graph for this function:

13.82.3.3 l4re_ma_alloc_align_srv()

```
long l4re_ma_alloc_align_srv (
            l4_cap_idx_t srv,
            long size,
            l4re_ds_t const mem,
            unsigned long flags,
            unsigned long align )
```

Allocate memory.

Parameters

srv	Memory allocator service.
size	Size to be requested.
mem	Capability slot to put the requested dataspace in
flags	Flags, see l4re_ma_flags
align	Log2 alignment of dataspace if supported by allocator, will be at least L4_PAGESHIFT, with Super_pages flag set at least L4_SUPERPAGESHIFT, default 0

Returns

0 on success, <0 on error

See also

L4Re::Mem_alloc::alloc

The memory allocator returns a dataspace.

Referenced by l4re_ma_alloc(), and l4re_ma_alloc_align().

Here is the caller graph for this function:

13.83 Memory descriptors (C version)

C Interface for KIP memory descriptors.

Collaboration diagram for Memory descriptors (C version):

Data Structures

- struct l4_kernel_info_mem_desc_t

  Memory descriptor data structure.

Typedefs

- typedef struct l4_kernel_info_mem_desc_t l4_kernel_info_mem_desc_t

  Memory descriptor data structure.

Enumerations

- enum l4_mem_type_t { l4_mem_type_undefined = 0x0 , l4_mem_type_conventional = 0x1 , l4_mem_type_reserved = 0x2 , l4_mem_type_dedicated = 0x3 , l4_mem_type_shared = 0x4 , l4_mem_type_info = 0xd , l4_mem_type_bootloader = 0xe , l4_mem_type_archspecific = 0xf }

  Type of a memory descriptor.
- enum l4_mem_info_sub_type_t { l4_mem_info_acpi_rsdp = 0 }

  Memory sub types for l4_mem_type_info descriptors.

Functions

- l4_kernel_info_mem_desc_t * l4_kernel_info_get_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW

  Get pointer to memory descriptors from KIP.
- unsigned l4_kernel_info_get_num_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW

  Get number of memory descriptors in KIP.
- void l4_kernel_info_set_mem_desc (l4_kernel_info_mem_desc_t *md, l4_addr_t start, l4_addr_t end, unsigned type, unsigned virt, unsigned sub_type) L4_NOTHROW

  Populate a memory descriptor.
- l4_umword_t l4_kernel_info_get_mem_desc_start (l4_kernel_info_mem_desc_t *md) L4_NOTHROW

  Get start address of the region described by the memory descriptor.
- l4_umword_t l4_kernel_info_get_mem_desc_end (l4_kernel_info_mem_desc_t *md) L4_NOTHROW

  Get end address of the region described by the memory descriptor.
- l4_umword_t l4_kernel_info_get_mem_desc_type (l4_kernel_info_mem_desc_t *md) L4_NOTHROW

  Get type of the memory region.
- l4_umword_t l4_kernel_info_get_mem_desc_subtype (l4_kernel_info_mem_desc_t *md) L4_NOTHROW

  Get sub-type of memory region.
- l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (l4_kernel_info_mem_desc_t *md) L4_NOTHROW

  Get virtual flag of the memory descriptor.

13.83.1 Detailed Description

C Interface for KIP memory descriptors.

Include File

```
#include <l4/sys/memdesc.h>
```

This module contains the C functions to access the memory descriptor in the kernel interface page (KIP).

13.83.2 Typedef Documentation

13.83.2.1 l4_kernel_info_mem_desc_t

```
typedef struct l4_kernel_info_mem_desc_t l4_kernel_info_mem_desc_t
```

Memory descriptor data structure.

Note

This data type is opaque, and must be accessed by the accessor functions defined in this module.

13.83.3 Enumeration Type Documentation

13.83.3.1 l4_mem_info_sub_type_t

```
enum l4_mem_info_sub_type_t
```

Memory sub types for l4_mem_type_info descriptors.

Enumerator

l4_mem_info_acpi_rsdp	Physical address of the ACPI root pointer.

Definition at line 61 of file memdesc.h.

13.83.3.2 l4_mem_type_t

```
enum l4_mem_type_t
```

Type of a memory descriptor.

Enumerator

l4_mem_type_undefined	Undefined, unused descriptor.
l4_mem_type_conventional	Conventional memory.
l4_mem_type_reserved	Reserved memory for kernel etc.
l4_mem_type_dedicated	Dedicated memory (some device memory)
l4_mem_type_shared	Shared memory (not implemented)
l4_mem_type_info	Info from the boot loader.
l4_mem_type_bootloader	Memory owned by the boot loader.
l4_mem_type_archspecific	Architecture specific memory (e.g., ACPI memory)

Definition at line 44 of file memdesc.h.

13.83.4 Function Documentation

13.83.4.1 l4_kernel_info_get_mem_desc_end()

```
l4_umword_t l4_kernel_info_get_mem_desc_end (
            l4_kernel_info_mem_desc_t * md )  [inline]
```

Get end address of the region described by the memory descriptor.

Returns

End address.

Definition at line 217 of file memdesc.h.

13.83.4.2 l4_kernel_info_get_mem_desc_is_virtual()

```
l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (
            l4_kernel_info_mem_desc_t * md )  [inline]
```

Get virtual flag of the memory descriptor.

Returns

1 if region is virtual memory, 0 if region is physical memory

Definition at line 238 of file memdesc.h.

13.83.4.3 l4_kernel_info_get_mem_desc_start()

```
l4_umword_t l4_kernel_info_get_mem_desc_start (
            l4_kernel_info_mem_desc_t * md )  [inline]
```

Get start address of the region described by the memory descriptor.

Returns

Start address.

Definition at line 210 of file memdesc.h.

13.83.4.4 l4_kernel_info_get_mem_desc_subtype()

```
l4_umword_t l4_kernel_info_get_mem_desc_subtype (
            l4_kernel_info_mem_desc_t * md )  [inline]
```

Get sub-type of memory region.

Returns

Sub-type.

The sub type is defined for architecture specific memory descriptors (see l4_mem_type_archspecific) and has architecture specific meaning.

Definition at line 231 of file memdesc.h.

13.83.4.5 l4_kernel_info_get_mem_desc_type()

```
l4_umword_t l4_kernel_info_get_mem_desc_type (
            l4_kernel_info_mem_desc_t * md )  [inline]
```

Get type of the memory region.

Returns

Type of the region (see l4_mem_type_t).

Definition at line 224 of file memdesc.h.

13.83.4.6 l4_kernel_info_get_num_mem_descs()

```
unsigned l4_kernel_info_get_num_mem_descs (
            l4_kernel_info_t * kip )  [inline]
```

Get number of memory descriptors in KIP.

Returns

Number of memory descriptors.

Definition at line 188 of file memdesc.h.

13.83.4.7 l4_kernel_info_set_mem_desc()

```
void l4_kernel_info_set_mem_desc (
            l4_kernel_info_mem_desc_t * md,
            l4_addr_t start,
            l4_addr_t end,
            unsigned type,
            unsigned virt,
            unsigned sub_type )  [inline]
```

Populate a memory descriptor.

Parameters

md	Pointer to memory descriptor
start	Start of region
end	End of region
type	Type of region
virt	1 if virtual region, 0 if physical region
sub_type	Sub type.

Definition at line 195 of file memdesc.h.

13.84 Memory operations.

Operations for memory access.

Collaboration diagram for Memory operations.:

Enumerations

- enum L4_mem_op_widths { L4_MEM_WIDTH_1BYTE = 0 , L4_MEM_WIDTH_2BYTE = 1 , L4_MEM_WIDTH_4BYTE = 2 }

  Memory access width definitions.

Functions

- unsigned long l4_mem_read (unsigned long virtaddress, unsigned width)

  Read user task memory from kernel privilege level.
- void l4_mem_write (unsigned long virtaddress, unsigned width, unsigned long value)

  Write user task memory from kernel privilege level.

13.84.1 Detailed Description

Operations for memory access.

This module provides functionality to access user task memory from the kernel. This is needed for some devices that are only accessible from privileged processor mode. Only use this when absolutely required. This functionality is only available on the ARM architecture.

```
#include <l4/sys/mem_op.h>
```

13.84.2 Enumeration Type Documentation

13.84.2.1 L4_mem_op_widths

```
enum L4_mem_op_widths
```

Memory access width definitions.

Enumerator

L4_MEM_WIDTH_1BYTE	Access one byte (8-bit width)
L4_MEM_WIDTH_2BYTE	Access two bytes (16-bit width)
L4_MEM_WIDTH_4BYTE	Access four bytes (32-bit width)

Definition at line 51 of file mem_op.h.

13.84.3 Function Documentation

13.84.3.1 l4_mem_read()

```
unsigned long l4_mem_read (
            unsigned long virtaddress,
            unsigned width )  [inline]
```

Read user task memory from kernel privilege level.

Parameters

virtaddress	Virtual address in the calling task.
width	Width of access in bytes in log2,

See also

L4_mem_op_widths

Returns

Read value.

Upon an given invalid address or invalid width value the function does nothing.

Definition at line 141 of file mem_op.h.

References l4_mem_arm_op_call().

Here is the call graph for this function:

13.84.3.2 l4_mem_write()

```
void l4_mem_write (
            unsigned long virtaddress,
            unsigned width,
            unsigned long value )  [inline]
```

Write user task memory from kernel privilege level.

Parameters

virtaddress	Virtual address in the calling task.
width	Width of access in bytes in log2 (i.e. allowed values: 0, 1, 2)
value	Value to write.

Upon an given invalid address or invalid width value the function does nothing.

Definition at line 147 of file mem_op.h.

References l4_mem_arm_op_call().

Here is the call graph for this function:

13.85 Memory related

Memory related constants, data types and functions.

Collaboration diagram for Memory related:

Macros

- #define L4_PAGESIZE

  Minimal page size (in bytes).
- #define L4_PAGEMASK

  Mask for the page number.
- #define L4_LOG2_PAGESIZE

  Number of bits used for page offset.
- #define L4_SUPERPAGESIZE

  Size of a large page.
- #define L4_SUPERPAGEMASK

  Mask for the number of a large page.
- #define L4_LOG2_SUPERPAGESIZE

  Number of bits used as offset for a large page.
- #define L4_INVALID_PTR ((void ∗)L4_INVALID_ADDR)

  Invalid address as pointer type.
- #define L4_PAGESHIFT 12

  Size of a page, log2-based.
- #define L4_SUPERPAGESHIFT 21

  Size of a large page, log2-based.
- #define L4_PAGESHIFT 12

  Size of a page, log2-based.
- #define L4_SUPERPAGESHIFT 21

  Size of a large page, log2-based.
- #define L4_PAGESHIFT 12

  Size of a page log2-based.
- #define L4_SUPERPAGESHIFT 22

  Size of a large page log2-based.

Enumerations

- enum l4_addr_consts_t { L4_INVALID_ADDR = ∼0UL }

  Address related constants.

Functions

- l4_addr_t l4_trunc_page (l4_addr_t address) L4_NOTHROW

  Round an address down to the next lower page boundary.
- l4_addr_t l4_trunc_size (l4_addr_t address, unsigned char bits) L4_NOTHROW

  Round an address down to the next lower flex page with size bits.
- l4_addr_t l4_round_page (l4_addr_t address) L4_NOTHROW

  Round address up to the next page.
- l4_addr_t l4_round_size (l4_addr_t value, unsigned char bits) L4_NOTHROW

  Round value up to the next alignment with bits size.
- unsigned l4_bytes_to_mwords (unsigned size) L4_NOTHROW

  Determine how many machine words (l4_umword_t) are required to store a buffer of 'size' bytes.

13.85.1 Detailed Description

Memory related constants, data types and functions.

13.85.2 Macro Definition Documentation

13.85.2.1 L4_LOG2_PAGESIZE

```
#define L4_LOG2_PAGESIZE
```

Number of bits used for page offset.

Size of page in log2.

Definition at line 331 of file consts.h.

13.85.2.2 L4_LOG2_SUPERPAGESIZE

```
#define L4_LOG2_SUPERPAGESIZE
```

Number of bits used as offset for a large page.

Size of large page in log2

Definition at line 357 of file consts.h.

13.85.2.3 L4_PAGEMASK

```
#define L4_PAGEMASK
```

Mask for the page number.

Note

The most significant bits are set.

Definition at line 322 of file consts.h.

13.85.2.4 L4_SUPERPAGEMASK

```
#define L4_SUPERPAGEMASK
```

Mask for the number of a large page.

Note

The most significant bits are set.

Definition at line 349 of file consts.h.

13.85.2.5 L4_SUPERPAGESIZE

```
#define L4_SUPERPAGESIZE
```

Size of a large page.

A large page is a *super page* on IA32 or a *section* on ARM.

Definition at line 340 of file consts.h.

13.85.3 Enumeration Type Documentation

13.85.3.1 l4_addr_consts_t

```
enum l4_addr_consts_t
```

Address related constants.

Enumerator

L4_INVALID_ADDR	Invalid address.

Definition at line 425 of file consts.h.

13.85.4 Function Documentation

13.85.4.1 l4_bytes_to_mwords()

```
unsigned l4_bytes_to_mwords (
            unsigned size )  [inline]
```

Determine how many machine words (l4_umword_t) are required to store a buffer of 'size' bytes.

Parameters

size	The number of bytes to be translated into machine words.

Definition at line 418 of file consts.h.

13.85.4.2 l4_round_page()

```
l4_addr_t l4_round_page (
            l4_addr_t address )  [inline]
```

Round address up to the next page.

The address is rounded up to the next minimal page boundary. On most architectures this is a 4k page. Check L4_PAGESIZE for the minimal page size.

Parameters

address	The address to round up.

Definition at line 395 of file consts.h.

References L4_PAGEMASK, and L4_PAGESIZE.

Referenced by L4virtio::Driver::Block_device::setup_device().

Here is the caller graph for this function:

13.85.4.3 l4_round_size()

```
l4_addr_t l4_round_size (
            l4_addr_t value,
            unsigned char bits )  [inline]
```

Round value up to the next alignment with *bits* size.

Parameters

value	The value to round up to the next size-alignment.
bits	The size of the alignment (log2).

Definition at line 406 of file consts.h.

Referenced by cxx::List_alloc::alloc_max(), L4virtio::Driver::Block_device::setup_device(), L4virtio::Virtqueue::setup_simple(), and L4virtio::Virtqueue::total_size().

Here is the caller graph for this function:

13.85.4.4 l4_trunc_page()

```
l4_addr_t l4_trunc_page (
            l4_addr_t address ) [inline]
```

Round an address down to the next lower page boundary.

The address is rounded down to the next lower minimal page boundary. On most architectures this is a 4k page. Check L4_PAGESIZE for the minimal page size.

Parameters

address	The address to round.

Examples

examples/libs/l4re/c++/mem_alloc/ma+rm.cc, and examples/libs/l4re/c/ma+rm.c.

Definition at line 370 of file consts.h.

References L4_PAGEMASK.

Referenced by l4_touch_ro(), and l4_touch_rw().

Here is the caller graph for this function:

13.85.4.5 l4_trunc_size()

```
l4_addr_t l4_trunc_size (
            l4_addr_t address,
            unsigned char bits )  [inline]
```

Round an address down to the next lower flex page with size *bits*.

Parameters

address	The address to round.
bits	The size of the flex page (log2).

Definition at line 381 of file consts.h.

Referenced by cxx::List_alloc::alloc_max(), and l4_fpage_max_order().

Here is the caller graph for this function:

13.86 Message Items

Message item related functions.

Collaboration diagram for Message Items:

Enumerations

- enum l4_msg_item_consts_t {
  L4_ITEM_MAP = 8 , L4_ITEM_CONT = 1 , L4_MAP_ITEM_GRANT = 2 , L4_MAP_ITEM_MAP = 0 ,
  L4_RCV_ITEM_SINGLE_CAP = L4_ITEM_MAP | 2 , L4_RCV_ITEM_LOCAL_ID = 4 }

  Constants for message items.

Functions

- l4_umword_t l4_map_control (l4_umword_t spot, unsigned char cache, unsigned grant) L4_NOTHROW

  Create the first word for a map item for the memory space.
- l4_umword_t l4_map_obj_control (l4_umword_t spot, unsigned grant) L4_NOTHROW

  Create the first word for a map item for the object space.

13.86.1 Detailed Description

Message item related functions.

Message items are typed items that can be transferred via IPC operations. Message items are also used to specify receive windows for typed items to be received. Message items are placed in the message registers (MRs) of the UTCB of the sending thread. Receive items are placed in the buffer registers (BRs) of the UTCB of the receiving thread.

Message items are usually two-word data structures. The first word denotes the type of the message item (for example a memory flex-page, io flex-page or object flex-page) and the second word contains information depending on the type. There is actually one exception that is a small (one word) receive buffer item for a single capability.

13.86.2 Enumeration Type Documentation

13.86.2.1 l4_msg_item_consts_t

```
enum l4_msg_item_consts_t
```

Constants for message items.

Enumerator

L4_ITEM_MAP	Identify a message item as *map item*.
L4_ITEM_CONT	Denote that the following item shall be put into the same receive item as this one.
L4_MAP_ITEM_GRANT	Flag as *grant* instead of *map* operation.
L4_MAP_ITEM_MAP	Flag as usual *map* operation.
L4_RCV_ITEM_SINGLE_CAP	Mark the receive buffer to be a small receive item that describes a buffer for a single capability.
L4_RCV_ITEM_LOCAL_ID	The receiver requests to receive a local ID instead of a mapping whenever possible.

Definition at line 193 of file consts.h.

13.86.3 Function Documentation

13.86.3.1 l4_map_control()

```
l4_umword_t l4_map_control (
            l4_umword_t spot,
            unsigned char cache,
            unsigned grant )  [inline]
```

Create the first word for a map item for the memory space.

Parameters

spot	Hot spot address, used to determine what is actually mapped when send and receive flex page have differing sizes.
cache	Cacheability hints for memory flex pages. See Cacheability options
grant	Indicates if it is a map or a grant item. Allowed values: L4_MAP_ITEM_MAP, L4_MAP_ITEM_GRANT.

Returns

The value to be used as first word in a map item for memory.

Definition at line 703 of file __l4_fpage.h.

References L4_FPAGE_CONTROL_MASK, and L4_ITEM_MAP.

Referenced by l4_map_obj_control().

Here is the caller graph for this function:

13.86.3.2 l4_map_obj_control()

```
l4_umword_t l4_map_obj_control (
            l4_umword_t spot,
            unsigned grant )  [inline]
```

Create the first word for a map item for the object space.

Parameters

spot	Hot spot address, used to determine what is actually mapped when send and receive flex pages have different size.
grant	Indicates if it is a map item or a grant item. Allowed values: L4_MAP_ITEM_MAP, L4_MAP_ITEM_GRANT.

Returns

The value to be used as first word in a map item for kernel objects or IO-ports.

Definition at line 710 of file __l4_fpage.h.

References l4_map_control().

Referenced by L4::Cap_base::snd_base().

Here is the call graph for this function:

Here is the caller graph for this function:

13.87 Message Registers (MRs)

Collaboration diagram for Message Registers (MRs):

Modules

- Exception registers

  Overly definition of the MRs for exception messages.

Data Structures

- union l4_msg_regs_t

  Encapsulation of the message-register block in the UTCB.

Typedefs

- typedef union l4_msg_regs_t l4_msg_regs_t

  Encapsulation of the message-register block in the UTCB.

13.87.1 Detailed Description

13.88 Message Tag

API related to the message tag data type.

Collaboration diagram for Message Tag:

Data Structures

- struct l4_msgtag_t

  Message tag data structure.

Typedefs

- typedef struct l4_msgtag_t l4_msgtag_t

  Message tag data structure.

Enumerations

- enum L4_platform_ctl_proto { L4_PROTO_PLATFORM_CTL = 0 }

  Predefined protocol type for messages to platform-control objects.
- enum l4_msgtag_protocol {
  L4_PROTO_NONE = 0 , L4_PROTO_ALLOW_SYSCALL = 1 , L4_PROTO_PF_EXCEPTION = 1 , L4_PROTO_IRQ = -1L ,
  L4_PROTO_PAGE_FAULT = -2L , L4_PROTO_PREEMPTION = -3L , L4_PROTO_SYS_EXCEPTION = -4L , L4_PROTO_EXCEPTION = -5L ,
  L4_PROTO_SIGMA0 = -6L , L4_PROTO_IO_PAGE_FAULT = -8L , L4_PROTO_KOBJECT = -10L , L4_PROTO_TASK = -11L ,
  L4_PROTO_THREAD = -12L , L4_PROTO_LOG = -13L , L4_PROTO_SCHEDULER = -14L , L4_PROTO_FACTORY = -15L ,
  L4_PROTO_VM = -16L , L4_PROTO_DMA_SPACE = -17L , L4_PROTO_IRQ_SENDER = -18L , L4_PROTO_IRQ_MUX = -19L ,
  L4_PROTO_SEMAPHORE = -20L , L4_PROTO_META = -21L , L4_PROTO_IOMMU = -22L , L4_PROTO_DEBUGGER = -23L ,
  L4_PROTO_SMCCC = -24L }

  Message tag for IPC operations.
- enum l4_msgtag_flags { L4_MSGTAG_ERROR , L4_MSGTAG_TRANSFER_FPU , L4_MSGTAG_SCHEDULE , **L4_MSGTAG_PROPAGATE** = 0x4000 , L4_MSGTAG_FLAGS }

  Flags for message tags.

Functions

- l4_msgtag_t l4_msgtag (long label, unsigned words, unsigned items, unsigned flags) L4_NOTHROW

  Create a message tag from the specified values.
- long l4_msgtag_label (l4_msgtag_t t) L4_NOTHROW

  Get the protocol of tag.
- unsigned l4_msgtag_words (l4_msgtag_t t) L4_NOTHROW

  Get the number of untyped words.
- unsigned l4_msgtag_items (l4_msgtag_t t) L4_NOTHROW

  Get the number of typed items.
- unsigned l4_msgtag_flags (l4_msgtag_t t) L4_NOTHROW

  Get the flags.
- unsigned l4_msgtag_has_error (l4_msgtag_t t) L4_NOTHROW

  Test for error indicator flag.
- unsigned l4_msgtag_is_page_fault (l4_msgtag_t t) L4_NOTHROW

  Test for page-fault protocol.
- unsigned l4_msgtag_is_preemption (l4_msgtag_t t) L4_NOTHROW

  Test for preemption protocol.
- unsigned l4_msgtag_is_sys_exception (l4_msgtag_t t) L4_NOTHROW

  Test for system-exception protocol.
- unsigned l4_msgtag_is_exception (l4_msgtag_t t) L4_NOTHROW

  Test for exception protocol.
- unsigned l4_msgtag_is_sigma0 (l4_msgtag_t t) L4_NOTHROW

  Test for sigma0 protocol.
- unsigned l4_msgtag_is_io_page_fault (l4_msgtag_t t) L4_NOTHROW

  Test for IO-page-fault protocol.

13.88.1 Detailed Description

API related to the message tag data type.

Include File

```
#include <l4/sys/types.h>
```

13.88.2 Typedef Documentation

13.88.2.1 l4_msgtag_t

```
typedef struct l4_msgtag_t l4_msgtag_t
```

Message tag data structure.

Include File

```
#include <l4/sys/types.h>
```

Describes the details of an IPC operation, in particular which parts of the UTCB have to be transmitted, and also flags to enable real-time and FPU extensions.

The message tag also contains a user-defined label that could be used to specify a protocol ID. Some negative values are reserved for kernel protocols such as page faults and exceptions.

The type must be treated completely opaque.

13.88.3 Enumeration Type Documentation

13.88.3.1 l4_msgtag_flags

```
enum l4_msgtag_flags
```

Flags for message tags.

Enumerator

L4_MSGTAG_ERROR	Error indicator flag.
L4_MSGTAG_TRANSFER_FPU	Enable FPU transfer flag for IPC. By enabling this flag when sending IPC, the sender indicates that the contents of the FPU shall be transferred to the receiving thread. However, the receiver has to indicate its willingness to receive FPU context in its buffer descriptor register (BDR).
L4_MSGTAG_SCHEDULE	Enable schedule in IPC flag. Usually IPC operations donate the remaining time slice of a thread to the called thread. Enabling this flag when sending IPC does a real scheduling decision. However, this flag decreases IPC performance.
L4_MSGTAG_FLAGS	Mask for all flags.

Definition at line 95 of file types.h.

13.88.3.2 l4_msgtag_protocol

`enum l4_msgtag_protocol`

Message tag for IPC operations.

All predefined protocols used by the kernel.

Enumerator

L4_PROTO_NONE	Default protocol tag to reply to kernel.
L4_PROTO_ALLOW_SYSCALL	Allow an alien the system call.
L4_PROTO_PF_EXCEPTION	Make an exception out of a page fault.
L4_PROTO_IRQ	IRQ message.
L4_PROTO_PAGE_FAULT	Page fault message.
L4_PROTO_PREEMPTION	Preemption message.
L4_PROTO_SYS_EXCEPTION	System exception.
L4_PROTO_EXCEPTION	Exception.
L4_PROTO_SIGMA0	Sigma0 protocol.
L4_PROTO_IO_PAGE_FAULT	I/O page fault message.
L4_PROTO_KOBJECT	Protocol for messages to a generic kobject.
L4_PROTO_TASK	Protocol for messages to a task object.
L4_PROTO_THREAD	Protocol for messages to a thread object.
L4_PROTO_LOG	Protocol for messages to a log object.
L4_PROTO_SCHEDULER	Protocol for messages to a scheduler object.
L4_PROTO_FACTORY	Protocol for messages to a factory object.
L4_PROTO_VM	Protocol for messages to a virtual machine object.
L4_PROTO_DMA_SPACE	Protocol for (creating) kernel DMA space objects.
L4_PROTO_IRQ_SENDER	Protocol for IRQ senders (IRQ -> IPC)
L4_PROTO_IRQ_MUX	Protocol for IRQ mux (IRQ -> n x IRQ)
L4_PROTO_SEMAPHORE	Protocol for semaphore objects.
L4_PROTO_META	Meta information protocol.
L4_PROTO_IOMMU	Protocol ID for IO-MMUs.
L4_PROTO_DEBUGGER	Protocol ID for the debugger.
L4_PROTO_SMCCC	Protocol ID for ARM SMCCC calls.

Definition at line 49 of file types.h.

13.88.3.3 L4_platform_ctl_proto

`enum L4_platform_ctl_proto`

Predefined protocol type for messages to platform-control objects.

Enumerator

L4_PROTO_PLATFORM_CTL	Protocol messages to a platform control object. See L4_platform_ctl_ops for allowed operations.

Definition at line 179 of file platform_control.h.

13.88.4 Function Documentation

13.88.4.1 l4_msgtag()

```
l4_msgtag_t l4_msgtag (
            long label,
            unsigned words,
            unsigned items,
            unsigned flags )  [inline]
```

Create a message tag from the specified values.

Message tag functions.

Parameters

label	The user-defined label
words	The number of untyped words within the UTCB
items	The number of typed items (e.g., flex pages) within the UTCB
flags	The IPC flags for realtime and FPU extensions

Returns

Message tag

Examples

examples/sys/aliens/main.c, examples/sys/ipc/ipc_example.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/mai and examples/sys/utcb-ipc/main.c.

Definition at line 411 of file types.h.

Referenced by L4::Irqep_t< Derived, BASE, bool >::dispatch(), L4::Server< LOOP_HOOKS >::internal_loop(), l4_irq_trigger_u(), l4util_ioport_map(), L4::Thread::modify_senders(), and L4::Ipc_svr::Exc_dispatch< R, Exc >::operator()().

Here is the caller graph for this function:

13.88.4.2 l4_msgtag_flags()

```
unsigned l4_msgtag_flags (
            l4_msgtag_t t )  [inline]
```

Get the flags.

The flag are defined by l4_msgtag_flags.

Parameters

t	The tag

Returns

Flags

Definition at line 435 of file types.h.

13.88.4.3 l4_msgtag_has_error()

```
unsigned l4_msgtag_has_error (
            l4_msgtag_t t )  [inline]
```

Test for error indicator flag.

Parameters

t	The tag

Returns

>0 for yes, 0 for no

Return whether the kernel operation caused a communication error, e.g. with IPC. if true: utcb->error is valid, otherwise utcb->error is not valid

Examples

examples/sys/aliens/main.c, examples/sys/singlestep/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 440 of file types.h.

References L4_MSGTAG_ERROR.

Referenced by l4_ipc_error(), and l4_vcon_write_u().

Here is the caller graph for this function:

13.88.4.4 l4_msgtag_is_exception()

```
unsigned l4_msgtag_is_exception (
            l4_msgtag_t t ) [inline]
```

Test for exception protocol.

Parameters

t	The tag

Returns

Boolean value

Examples

examples/sys/aliens/main.c, examples/sys/singlestep/main.c, and examples/sys/start-with-exc/main.c.

Definition at line 454 of file types.h.

References l4_msgtag_label(), and L4_PROTO_EXCEPTION.

Here is the call graph for this function:

13.88.4.5 l4_msgtag_is_io_page_fault()

```
unsigned l4_msgtag_is_io_page_fault (
            l4_msgtag_t t ) [inline]
```

Test for IO-page-fault protocol.

Parameters

t	The tag

Returns

Boolean value

Definition at line 460 of file types.h.

References l4_msgtag_label(), and L4_PROTO_IO_PAGE_FAULT.

Here is the call graph for this function:

13.88.4.6 l4_msgtag_is_page_fault()

```
unsigned l4_msgtag_is_page_fault (
            l4_msgtag_t t ) [inline]
```

Test for page-fault protocol.

Parameters

t	The tag

Returns

Boolean value

Definition at line 445 of file types.h.

References l4_msgtag_label(), and L4_PROTO_PAGE_FAULT.

Here is the call graph for this function:

13.88.4.7 l4_msgtag_is_preemption()

```
unsigned l4_msgtag_is_preemption (
            l4_msgtag_t t ) [inline]
```

Test for preemption protocol.

Parameters

t	The tag

Returns

Boolean value

Definition at line 448 of file types.h.

References l4_msgtag_label(), and L4_PROTO_PREEMPTION.

Here is the call graph for this function:

13.88.4.8 l4_msgtag_is_sigma0()

```
unsigned l4_msgtag_is_sigma0 (
            l4_msgtag_t t ) [inline]
```

Test for sigma0 protocol.

Parameters

t	The tag

Returns

Boolean value

Definition at line 457 of file types.h.

References l4_msgtag_label(), and L4_PROTO_SIGMA0.

Here is the call graph for this function:

13.88.4.9 l4_msgtag_is_sys_exception()

```
unsigned l4_msgtag_is_sys_exception (
            l4_msgtag_t t ) [inline]
```

Test for system-exception protocol.

Parameters

t	The tag

Returns

Boolean value

Definition at line 451 of file types.h.

References l4_msgtag_label(), and L4_PROTO_SYS_EXCEPTION.

Here is the call graph for this function:

13.88.4.10 l4_msgtag_items()

```
unsigned l4_msgtag_items (
            l4_msgtag_t t ) [inline]
```

Get the number of typed items.

Parameters

t	The tag

Returns

Number of items.

Definition at line 431 of file types.h.

Referenced by l4util_ioport_map().

Here is the caller graph for this function:

13.88.4.11 l4_msgtag_label()

```
long l4_msgtag_label (
            l4_msgtag_t t ) [inline]
```

Get the protocol of tag.

Parameters

t	The tag

Returns

Label

Examples

examples/sys/singlestep/main.c, and examples/sys/start-with-exc/main.c.

Definition at line 423 of file types.h.

Referenced by l4_msgtag_is_exception(), l4_msgtag_is_io_page_fault(), l4_msgtag_is_page_fault(), l4_msgtag_is_preemption(), l4_msgtag_is_sigma0(), and l4_msgtag_is_sys_exception().

Here is the caller graph for this function:

13.88.4.12 l4_msgtag_words()

```
unsigned l4_msgtag_words (
            l4_msgtag_t t )  [inline]
```

Get the number of untyped words.

Parameters

t	The tag

Returns

Number of words

Examples

examples/sys/utcb-ipc/main.c.

Definition at line 427 of file types.h.

13.89 Name-space API

API for name spaces that store capabilities.

Collaboration diagram for Name-space API:

Data Structures

- class L4Re::Namespace

  Name-space interface.

13.89.1 Detailed Description

API for name spaces that store capabilities.

This is a basic abstraction for managing a mapping from human-readable names to capabilities. In particular, a name can also be mapped to a capability that refers to another name space object. By this means name spaces can be constructed hierarchically.

Name spaces play a central role in L4Re, because the implementation of the name space objects determines the policy which capabilities (which objects) are accessible to a client of a name space.

13.90 Namespace interface

Namespace C interface.

Collaboration diagram for Namespace interface:

Enumerations

- enum l4re_ns_register_flags

  Namespace register flags.

Functions

- long l4re_ns_query_to_srv (l4re_namespace_t srv, char const *name, l4_cap_idx_t const cap, int timeout) L4_NOTHROW

  Query the name space for the object named by `name`.
- long l4re_ns_query_srv (l4re_namespace_t srv, char const *name, l4_cap_idx_t const cap) L4_NOTHROW

  Query the name space for the object named by `name`.
- long l4re_ns_register_obj_srv (l4re_namespace_t srv, char const *name, l4_cap_idx_t const obj, unsigned flags) L4_NOTHROW

  Register an object with a name.

13.90.1 Detailed Description

Namespace C interface.

13.90.2 Enumeration Type Documentation

13.90.2.1 l4re_ns_register_flags

```
enum l4re_ns_register_flags
```

Namespace register flags.

See also

L4Re::Namespace::Register_flags

Definition at line 39 of file namespace.h.

13.90.3 Function Documentation

13.90.3.1 l4re_ns_query_srv()

```
long l4re_ns_query_srv (
            l4re_namespace_t srv,
            char const * name,
            l4_cap_idx_t const cap )  [inline]
```

Query the name space for the object named by `name`.

Parameters

srv	Name space server to use for the query.
name	String to query.
cap	Capability slot where the received capability will be stored.

Return values

0	Name could be fully resolved.
>0	Name could only be partly resolved. The number of remaining characters is returned.
-L4_ENOENT	Entry could not be found.
-L4_EAGAIN	Entry exists but no object is yet attached. Try again later.
<0	IPC errors, see l4_error_code_t.

Definition at line 105 of file namespace.h.

References l4re_ns_query_to_srv().

Here is the call graph for this function:

13.90.3.2 l4re_ns_query_to_srv()

```
long l4re_ns_query_to_srv (
            l4re_namespace_t srv,
            char const * name,
            l4_cap_idx_t const cap,
            int timeout )
```

Query the name space for the object named by `name`.

Parameters

timeout	Timeout of query in milliseconds. The client will only wait if a name already has been registered with the server but no object has been attached yet.
srv	Name space server to use for the query.
name	String to query.
cap	Capability slot where the received capability will be stored.

Return values

0	Name could be fully resolved.
>0	Name could only be partly resolved. The number of remaining characters is returned.
-L4_ENOENT	Entry could not be found.
-L4_EAGAIN	Entry exists but no object is yet attached. Try again later.
<0	IPC errors, see l4_error_code_t.

Referenced by l4re_ns_query_srv().

Here is the caller graph for this function:

13.90.3.3 l4re_ns_register_obj_srv()

```
long l4re_ns_register_obj_srv (
            l4re_namespace_t srv,
            char const * name,
            l4_cap_idx_t const obj,
            unsigned flags )
```

Register an object with a name.

Parameters

srv	Name space server to use for the query.
name	Name under which the object should be registered.
obj	Capability to object to register. An invalid capability may be given to only reserve the name for later use.
flags	Flags to assign to the entry, see L4Re::Namespace::Register_flags. Note that the rights that are assigned to a capability are not only determined by the rights given in these flags but also by the rights with which the `obj` capability was mapped to the name space.

Return values

0	Object was successfully registered with *name*.
-L4_EEXIST	Name already registered.
-L4_EPERM	Caller doesn't have necessary permissions.
-L4_ENOMEM	Server has insufficient resources.

Return values

-L4_EINVAL	Invalid parameter.
<0	IPC errors, see l4_error_code_t.

Precondition

requires capability rights: {RW}

13.91 Object Invocation

API for L4 object invocation.

Collaboration diagram for Object Invocation:

Modules

- Error Handling

  Error handling for L4 object invocation.
- Message Items

  Message item related functions.
- Message Tag

  API related to the message tag data type.
- Realtime API
- Timeouts

  All kinds of timeouts and time related functions.
- Virtual Registers (UTCBs)

  L4 Virtual Registers (UTCB).

Files

- file utcb.h

  UTCB definitions.

Enumerations

- enum l4_syscall_flags_t {
  L4_SYSF_NONE , L4_SYSF_SEND , L4_SYSF_RECV , L4_SYSF_OPEN_WAIT ,
  L4_SYSF_REPLY , L4_SYSF_CALL , L4_SYSF_WAIT , L4_SYSF_SEND_AND_WAIT ,
  L4_SYSF_REPLY_AND_WAIT }

  Capability selector flags.

Functions

- l4_msgtag_t l4_ipc_send (l4_cap_idx_t dest, l4_utcb_t ∗utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW

  *Send a message to an object (do **not** wait for a reply).*
- l4_msgtag_t l4_ipc_wait (l4_utcb_t ∗utcb, l4_umword_t ∗label, l4_timeout_t timeout) L4_NOTHROW

  Wait for an incoming message from any possible sender.
- l4_msgtag_t l4_ipc_receive (l4_cap_idx_t object, l4_utcb_t ∗utcb, l4_timeout_t timeout) L4_NOTHROW

  Wait for a message from a specific source.
- l4_msgtag_t l4_ipc_call (l4_cap_idx_t object, l4_utcb_t ∗utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW

  Object call (usual invocation).
- l4_msgtag_t l4_ipc_reply_and_wait (l4_utcb_t ∗utcb, l4_msgtag_t tag, l4_umword_t ∗label, l4_timeout_t timeout) L4_NOTHROW

  Reply and wait operation (uses the reply capability).
- l4_msgtag_t l4_ipc_send_and_wait (l4_cap_idx_t dest, l4_utcb_t ∗utcb, l4_msgtag_t tag, l4_umword_t ∗label, l4_timeout_t timeout) L4_NOTHROW

  Send a message and do an open wait.
- l4_msgtag_t l4_ipc (l4_cap_idx_t dest, l4_utcb_t ∗utcb, l4_umword_t flags, l4_umword_t slabel, l4_msgtag_t tag, l4_umword_t ∗rlabel, l4_timeout_t timeout) L4_NOTHROW

  Generic L4 object invocation.
- l4_msgtag_t l4_ipc_sleep (l4_timeout_t timeout) L4_NOTHROW

  Sleep for an amount of time.
- int l4_sndfpage_add (l4_fpage_t const snd_fpage, unsigned long snd_base, l4_msgtag_t ∗tag) L4_NOTHROW

  Add a flex-page to be sent to the UTCB.

13.91.1 Detailed Description

API for L4 object invocation.

Include File

```
#include <l4/sys/ipc.h>
```

General abstractions for L4 object invocation. The basic principle is that all objects are denoted by a capability that is accessed via a capability selector (see Capabilities).

This set of functions is common to all kinds of objects provided by the L4 micro kernel. The concrete semantics of an invocation depends on the object that shall be invoked.

Objects may be invoked in various ways, the most common way is to use a *call* operation (l4_ipc_call()). However, there are a lot more flavours available that have a semantics depending on the object.

See also

IPC-Gate API

13.91.2 Enumeration Type Documentation

13.91.2.1 l4_syscall_flags_t

```
enum l4_syscall_flags_t
```

Capability selector flags.

These flags determine the concrete operation when a kernel object is invoked.

Enumerator

L4_SYSF_NONE	Default flags (call to a kernel object). Using this value as flags in the capability selector for an invocation indicates a call (send and wait for a reply).
L4_SYSF_SEND	Send-phase flag. Setting this flag in a capability selector induces a send phase, this means a message is sent to the object denoted by the capability. For receive phase see L4_SYSF_RECV.
L4_SYSF_RECV	Receive-phase flag. Setting this flag in a capability selector induces a receive phase, this means the invoking thread waits for a message from the object denoted by the capability. For a send phase see L4_SYSF_SEND.
L4_SYSF_OPEN_WAIT	Open-wait flag. This flag indicates that the receive operation (see L4_SYSF_RECV) shall be an *open wait*. *Open wait* means that the invoking thread shall wait for a message from any possible sender and *not* from the sender denoted by the capability.
L4_SYSF_REPLY	Reply flag. This flag indicates that the send phase shall use the in-kernel reply capability instead of the capability denoted by the selector index.
L4_SYSF_CALL	Call flags (combines send and receive). Combines L4_SYSF_SEND and L4_SYSF_RECV.
L4_SYSF_WAIT	Wait flags (combines receive and open wait). Combines L4_SYSF_RECV and L4_SYSF_OPEN_WAIT.
L4_SYSF_SEND_AND_WAIT	Send-and-wait flags. Combines L4_SYSF_SEND and L4_SYSF_WAIT.
L4_SYSF_REPLY_AND_WAIT	Reply-and-wait flags. Combines L4_SYSF_SEND, L4_SYSF_REPLY, and L4_SYSF_WAIT.

Definition at line 39 of file consts.h.

13.91.3 Function Documentation

13.91.3.1 l4_ipc()

```
l4_msgtag_t l4_ipc (
            l4_cap_idx_t dest,
            l4_utcb_t * utcb,
            l4_umword_t flags,
            l4_umword_t slabel,
            l4_msgtag_t tag,
            l4_umword_t * rlabel,
            l4_timeout_t timeout )  [inline]
```

Generic L4 object invocation.

Parameters

	dest	Destination object. L4_INVALID_CAP denotes the current thread. An IPC to the current thread will always abort after the specified timeout and can be used for sleeping without busy waiting.
	utcb	UTCB of the caller.
	flags	Invocation flags (see l4_syscall_flags_t).
	slabel	Send label if applicable (may be seen by the receiver).
	tag	Sending message tag.
out	*rlabel*	Receiving label.
	timeout	Timeout pair (see l4_timeout_t).

Returns

return tag

Definition at line 34 of file ipc.h.

Referenced by l4_ipc_call(), l4_ipc_receive(), l4_ipc_reply_and_wait(), l4_ipc_send(), l4_ipc_send_and_wait(), and l4_ipc_wait().

Here is the caller graph for this function:

13.91.3.2 l4_ipc_call()

```
l4_msgtag_t l4_ipc_call (
            l4_cap_idx_t object,
            l4_utcb_t * utcb,
            l4_msgtag_t tag,
            l4_timeout_t timeout )  [inline]
```

Object call (usual invocation).

Parameters

object	Capability selector for the object to call. A value of L4_INVALID_CAP denotes the current thread and will abort the IPC after the time specified in the `snd` part of the `timeout` parameter has expired.
utcb	UTCB of the caller.
tag	Message tag to describe the message to be sent.
timeout	Timeout pair for send an receive phase (see l4_timeout_t).

Returns

result tag

A message is sent to the object and the invoker waits for a reply from the object. Messages from other sources are not accepted.

Note

The send-to-receive transition needs no time, the object can reply with a send timeout of zero.

Examples

examples/sys/aliens/main.c, examples/sys/ipc/ipc_example.c, and examples/sys/singlestep/main.c.

Definition at line 463 of file ipc.h.

References l4_ipc(), and L4_SYSF_CALL.

Referenced by L4::Ipc::Iostream::call(), l4util_ioport_map(), and L4::Thread::modify_senders().

Here is the call graph for this function:

Here is the caller graph for this function:

13.91.3.3 l4_ipc_receive()

```
l4_msgtag_t l4_ipc_receive (
            l4_cap_idx_t object,
            l4_utcb_t * utcb,
            l4_timeout_t timeout )  [inline]
```

Wait for a message from a specific source.

Parameters

object	Object to receive a message from. A value of L4_INVALID_CAP denotes the current thread. It could be used for sleeping without busy waiting for the time specified in the `rcv` part of the `timeout` parameter.
timeout	Timeout pair (see l4_timeout_t, only the receive part matters).
utcb	UTCB of the caller.

Returns

result tag.

This operation waits for a message from the specified object. Messages from other sources are not accepted by this operation. The operation does not include a send phase, this means no message is sent to the object.

Note

This operation is usually used to receive messages from a specific IRQ or thread. However, it is not common to use this operation for normal applications.

Examples

examples/sys/aliens/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 505 of file ipc.h.

References l4_ipc(), L4_SYSF_RECV, and l4_msgtag_t::raw.

Referenced by l4_ipc_sleep(), and l4_thread_yield().

Here is the call graph for this function:

Here is the caller graph for this function:

13.91.3.4 l4_ipc_reply_and_wait()

```
l4_msgtag_t l4_ipc_reply_and_wait (
            l4_utcb_t * utcb,
            l4_msgtag_t tag,
            l4_umword_t * label,
            l4_timeout_t timeout )  [inline]
```

Reply and wait operation (uses the *reply* capability).

Parameters

	utcb	UTCB of the caller.
	tag	Describes the message to be sent as reply.
out	*label*	Label assigned to the source object of the received message.
	timeout	Timeout pair (see l4_timeout_t).

Returns

result tag

A message is sent to the previous caller using the implicit reply capability. Afterwards the invoking thread waits for a message from any source.

Note

This is the standard server operation: it sends a reply to the actual client and waits for the next incoming request, which may come from any other client.

Examples

examples/sys/ipc/ipc_example.c.

Definition at line 471 of file ipc.h.

References L4_INVALID_CAP, l4_ipc(), and L4_SYSF_REPLY_AND_WAIT.

Referenced by L4::Ipc::Iostream::reply_and_wait(), and L4::Server< LOOP_HOOKS >::reply_n_wait().

Here is the call graph for this function:

Here is the caller graph for this function:

13.91.3.5 l4_ipc_send()

```
l4_msgtag_t l4_ipc_send (
            l4_cap_idx_t dest,
            l4_utcb_t * utcb,
            l4_msgtag_t tag,
            l4_timeout_t timeout )  [inline]
```

Send a message to an object (do **not** wait for a reply).

Parameters

dest	Capability selector for the destination object. A value of L4_INVALID_CAP denotes the current thread and could be used for sleeping without busy waiting for the time specified in the `snd` part of the `timeout` parameter.
utcb	UTCB of the caller.
tag	Descriptor for the message to be sent.
timeout	Timeout pair (see l4_timeout_t) only send part is relevant.

Returns

result tag

A message is sent to the destination object. There is no receive phase included. The invoker continues working after sending the message.

Attention

This is a special-purpose message transfer, objects usually support only invocation via l4_ipc_call().

Examples

examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 488 of file ipc.h.

References l4_ipc(), and L4_SYSF_SEND.

Referenced by l4_irq_trigger_u(), L4::Server< LOOP_HOOKS >::reply_n_wait(), and L4::Ipc::Ostream::send().

Here is the call graph for this function:

Here is the caller graph for this function:

13.91.3.6 l4_ipc_send_and_wait()

```
l4_msgtag_t l4_ipc_send_and_wait (
            l4_cap_idx_t dest,
            l4_utcb_t * utcb,
            l4_msgtag_t tag,
            l4_umword_t * label,
            l4_timeout_t timeout )  [inline]
```

Send a message and do an open wait.

Parameters

	dest	Object to send a message to. A value of L4_INVALID_CAP denotes the current thread and will abort the IPC after the time specified in the `snd` part of the `timeout` parameter has expired.
	utcb	UTCB of the caller.
	tag	Describes the message that shall be sent.
`out`	*label*	Label assigned to the source object of the receive phase.
	timeout	Timeout pair (see l4_timeout_t).

Returns

result tag

A message is sent to the destination object and the invoking thread waits for a reply from any source.

Note

This is a special-purpose operation and shall not be used in general applications.

Definition at line 479 of file ipc.h.

References l4_ipc(), and L4_SYSF_SEND_AND_WAIT.

Here is the call graph for this function:

13.91.3.7 l4_ipc_sleep()

```
l4_msgtag_t l4_ipc_sleep (
            l4_timeout_t timeout ) [inline]
```

Sleep for an amount of time.

Parameters

timeout	Timeout pair (see l4_timeout_t, the receive part matters).

Returns

error code:

- L4_IPC_RETIMEOUT: success
- L4_IPC_RECANCELED woken up by a different thread (l4_thread_ex_regs()).

The invoking thread waits until the timeout is expired or the wait was aborted by another thread by l4_thread_ex_regs().

Definition at line 514 of file ipc.h.

References L4_INVALID_CAP, and l4_ipc_receive().

Referenced by l4_sleep_forever(), and l4_usleep().

Here is the call graph for this function:

Here is the caller graph for this function:

13.91.3.8 l4_ipc_wait()

```
l4_msgtag_t l4_ipc_wait (
            l4_utcb_t * utcb,
            l4_umword_t * label,
            l4_timeout_t timeout )  [inline]
```

Wait for an incoming message from any possible sender.

Parameters

	utcb	UTCB of the caller.
out	*label*	Label assigned to the source object (IPC gate or IRQ).
	timeout	Timeout pair (see l4_timeout_t, only the receive part is used).

Returns

return tag

This operation does an open wait, and therefore needs no capability to denote the possible source of a message. This means the calling thread waits for an incoming message from any possible source. There is no send phase included in this operation.

The usual usage of this function is to call that function when entering a server loop in a user-level server that implements user-level objects, see also l4_ipc_reply_and_wait().

Examples

examples/sys/ipc/ipc_example.c.

Definition at line 496 of file ipc.h.

References L4_INVALID_CAP, l4_ipc(), L4_SYSF_WAIT, and l4_msgtag_t::raw.

Referenced by L4::Server< LOOP_HOOKS >::reply_n_wait(), and L4::Ipc::Istream::wait().

Here is the call graph for this function:

Here is the caller graph for this function:

13.91.3.9 l4_sndfpage_add()

```
int l4_sndfpage_add (
            l4_fpage_t const snd_fpage,
            unsigned long snd_base,
            l4_msgtag_t * tag )  [inline]
```

Add a flex-page to be sent to the UTCB.

Parameters

snd_fpage	Flex-page.
snd_base	Send base.
tag	Tag to be modified.

Return values

tag	Modified tag, the number of items will be increased, all other values in the tag will be retained.

Returns

0 on success, negative error code otherwise

Definition at line 574 of file ipc.h.

13.92 Parent API

Parent interface.

Collaboration diagram for Parent API:

Data Structures

- class L4Re::Parent

  Parent interface.

13.92.1 Detailed Description

Parent interface.

The parent interface provides means for an L4 task to signal changes in its execution state. The main purpose is to signal program termination to the program that started it, so that its resources can be reclaimed. In a typical L4Re system, this program will be Moe or Ned.

See also

L4Re::Parent for information about the concrete interface.

13.93 Platform Control C API

C interface for controlling platform-wide properties, see L4::Platform_control for the C++ interface.

Collaboration diagram for Platform Control C API:

Functions

- l4_msgtag_t l4_platform_ctl_system_suspend (l4_cap_idx_t pfc, l4_umword_t extras) L4_NOTHROW

  Enter suspend to RAM.
- l4_msgtag_t l4_platform_ctl_system_shutdown (l4_cap_idx_t pfc, l4_umword_t reboot) L4_NOTHROW

  Shutdown or reboot the system.
- l4_msgtag_t l4_platform_ctl_cpu_allow_shutdown (l4_cap_idx_t pfc, l4_umword_t phys_id, l4_umword_t enable) L4_NOTHROW

  Allow a CPU to be shut down.
- l4_msgtag_t l4_platform_ctl_cpu_enable (l4_cap_idx_t pfc, l4_umword_t phys_id) L4_NOTHROW

  Enable an offline CPU.
- l4_msgtag_t l4_platform_ctl_cpu_disable (l4_cap_idx_t pfc, l4_umword_t phys_id) L4_NOTHROW

  Disable an online CPU.

13.93.1 Detailed Description

C interface for controlling platform-wide properties, see L4::Platform_control for the C++ interface.

Include File

```
#include <l4/sys/platform_control.h>
```

The API allows a client to suspend, reboot or shutdown the system.

For the C++ interface refer to L4::Platform_control

13.93.2 Function Documentation

13.93.2.1 l4_platform_ctl_cpu_allow_shutdown()

```
l4_msgtag_t l4_platform_ctl_cpu_allow_shutdown (
            l4_cap_idx_t pfc,
            l4_umword_t phys_id,
            l4_umword_t enable )  [inline]
```

Allow a CPU to be shut down.

Parameters

pfc	Capability selector for the platform-control object.
phys↩ _id	Physical CPU id of CPU (e.g. local APIC id) to enable.
enable	Allow shutdown when 1, disallow when 0.

Returns

Syscall return tag

Definition at line 247 of file platform_control.h.

13.93.2.2 l4_platform_ctl_cpu_disable()

```
l4_msgtag_t l4_platform_ctl_cpu_disable (
            l4_cap_idx_t pfc,
            l4_umword_t phys_id )  [inline]
```

Disable an online CPU.

Parameters

pfc	Capability to the platform control object.
phys↩ _id	Physical CPU id of CPU (e.g. local APIC id) to disable.

Returns

System call message tag

This function is currently only supported on the ARM EXYNOS platform.

Definition at line 286 of file platform_control.h.

13.93.2.3 l4_platform_ctl_cpu_enable()

```
l4_msgtag_t l4_platform_ctl_cpu_enable (
            l4_cap_idx_t pfc,
            l4_umword_t phys_id )  [inline]
```

Enable an offline CPU.

Parameters

pfc	Capability to the platform control object.
phys↩ _id	Physical CPU id of CPU (e.g. local APIC id) to enable.

Returns

System call message tag

This function is currently only supported on the ARM EXYNOS platform.

Definition at line 279 of file platform_control.h.

13.93.2.4 l4_platform_ctl_system_shutdown()

```
l4_msgtag_t l4_platform_ctl_system_shutdown (
            l4_cap_idx_t pfc,
            l4_umword_t reboot )  [inline]
```

Shutdown or reboot the system.

Parameters

pfc	Capability selector for the platform-control object.
reboot	Shutdown when 0, or reboot when 1.

Returns

Syscall return tag

Definition at line 226 of file platform_control.h.

13.93.2.5 l4_platform_ctl_system_suspend()

```
l4_msgtag_t l4_platform_ctl_system_suspend (
            l4_cap_idx_t pfc,
            l4_umword_t extras )  [inline]
```

Enter suspend to RAM.

Parameters

pfc	Capability selector for the platform-control object.
extras	Some extra platform-specific information needed to enter suspend to RAM. On x86 platforms and when using the Platform_control object provided by Fiasco, the value defines the sleep state. The sleep states are defined in the ACPI table. Other platforms as well as Io's Platform_control object don't make use of this value at the moment.

Returns

Syscall return tag

Definition at line 219 of file platform_control.h.

13.94 Producer

Collaboration diagram for Producer:

Functions

- long l4shmc_chunk_try_to_take (l4shmc_chunk_t ∗chunk)

  Try to mark chunk busy.
- long l4shmc_chunk_ready (l4shmc_chunk_t ∗chunk, l4_umword_t size)

  Mark chunk as filled (ready).
- long l4shmc_chunk_ready_sig (l4shmc_chunk_t ∗chunk, l4_umword_t size)

  Mark chunk as filled (ready) and signal consumer.
- long l4shmc_is_chunk_clear (l4shmc_chunk_t ∗chunk)

  Check whether chunk is free.

13.94.1 Detailed Description

13.94.2 Function Documentation

13.94.2.1 l4shmc_chunk_ready()

```
long l4shmc_chunk_ready (
            l4shmc_chunk_t * chunk,
            l4_umword_t size )  [inline]
```

Mark chunk as filled (ready).

Parameters

chunk	chunk.
size	Size of data in the chunk, in bytes.

Return values

0	Success.
<0	Error.

13.94.2.2 l4shmc_chunk_ready_sig()

```
long l4shmc_chunk_ready_sig (
            l4shmc_chunk_t * chunk,
            l4_umword_t size ) [inline]
```

Mark chunk as filled (ready) and signal consumer.

Parameters

chunk	chunk.
size	Size of data in the chunk, in bytes.

Return values

0	Success.
<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.94.2.3 l4shmc_chunk_try_to_take()

```
long l4shmc_chunk_try_to_take (
            l4shmc_chunk_t * chunk ) [inline]
```

Try to mark chunk busy.

Parameters

chunk	chunk to mark.

Return values

0	Chunk could be taken.
<0	Chunk could not be taken, try again.

Examples

examples/libs/shmc/prodcons.c.

13.94.2.4 l4shmc_is_chunk_clear()

```
long l4shmc_is_chunk_clear (
            l4shmc_chunk_t * chunk )  [inline]
```

Check whether chunk is free.

Parameters

chunk	Chunk to check.

Return values

0	Success.
<0	Error.

13.95 Producer

Collaboration diagram for Producer:

Functions

- long l4shmc_trigger (l4shmc_signal_t *signal)

  Trigger a signal.

13.95.1 Detailed Description

13.95.2 Function Documentation

13.95.2.1 l4shmc_trigger()

```
long l4shmc_trigger (
            l4shmc_signal_t * signal ) [inline]
```

Trigger a signal.

Parameters

signal	Signal to trigger.

Return values

0	Success.
<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.96 Random number support

Collaboration diagram for Random number support:

Functions

- l4_uint32_t l4util_rand (void)

  Deliver next random number.
- void l4util_srand (l4_uint32_t seed)

  Initialize random number generator.

13.96.1 Detailed Description

13.96.2 Function Documentation

13.96.2.1 l4util_rand()

```
l4_uint32_t l4util_rand (
            void  )
```

Deliver next random number.

Returns

A new random number

13.96.2.2 l4util_srand()

```
void l4util_srand (
            l4_uint32_t seed )
```

Initialize random number generator.

Parameters

seed	Value to initialize

13.97 Realtime API

Collaboration diagram for Realtime API:

13.98 Region map API

Virtual address-space management.

Collaboration diagram for Region map API:

Data Structures

- class L4Re::Rm

  Region map.

13.98.1 Detailed Description

Virtual address-space management.

A region map object implements two protocols. The first protocol is the kernel page-fault protocol, to resolve page faults for threads running in an L4 task. The second protocol is the region map protocol itself, which allows managing the virtual memory address space of an L4 task.

There are two basic concepts provided by the region map abstraction:

- **Areas** are reserved ranges in the virtual memory address space.
- **Regions** are areas that are backed by (part of) a dataspace, i.e. accessing them results in access to the physical memory the dataspace manages.

Areas can be reserved for special use or for attaching a dataspace at a later time. When attaching a dataspace, the user can instruct the region map to search for an appropriate range to attach to. Regions are skipped in this search since they already have dataspaces attached to them, and areas are skipped because they are reserved.

When a region map receives a page fault IPC, the region map will check if the faulting virtual address lies in a region. If yes, it will answer the page fault IPC with a mapping from the backing dataspace. If not, an error is returned.

See also

L4Re::Dataspace, L4Re::Rm,

Memory management - Data Spaces and the Region Map

13.99 Region map interface

Region map C interface.

Collaboration diagram for Region map interface:

Enumerations

- enum l4re_rm_flags_values {
  L4RE_RM_F_R = L4RE_DS_F_R , **L4RE_RM_F_W** = L4RE_DS_F_W , **L4RE_RM_F_X** = L4RE_DS_F_X , **L4RE_RM_F_RX** = L4RE_DS_F_RX ,
  L4RE_RM_F_RW = L4RE_DS_F_RW , **L4RE_RM_F_RWX** = L4RE_DS_F_RWX , L4RE_RM_F_NO_ALIAS = 0x200 , L4RE_RM_F_PAGER = 0x400 ,
  L4RE_RM_F_RESERVED = 0x800 , L4RE_RM_CACHING_SHIFT = 4 , L4RE_RM_F_CACHING = L4RE↩_DS_F_CACHING_MASK , L4RE_RM_REGION_FLAGS = 0xffff ,
  L4RE_RM_F_CACHE_NORMAL = L4RE_DS_F_NORMAL , L4RE_RM_F_CACHE_BUFFERED = L4↩RE_DS_F_BUFFERABLE , L4RE_RM_F_CACHE_UNCACHED = L4RE_DS_F_UNCACHEABLE ,
  L4RE_RM_F_SEARCH_ADDR = 0x20000 ,
  L4RE_RM_F_IN_AREA = 0x40000 , L4RE_RM_F_EAGER_MAP = 0x80000 , L4RE_RM_F_ATTACH_FLAGS = 0xf0000 }

  Flags for region operations.

Functions

- int l4re_rm_reserve_area (l4_addr_t ∗start, unsigned long size, l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW

  Reserve the given area in the region map.
- int l4re_rm_free_area (l4_addr_t addr) L4_NOTHROW

  Free an area from the region map.
- int l4re_rm_attach (void ∗∗start, unsigned long size, l4re_rm_flags_t flags, l4re_ds_t mem, l4re_rm_offset_t offs, unsigned char align) L4_NOTHROW

  Attach a data space to a region.
- int l4re_rm_detach (void ∗addr) L4_NOTHROW

  Detach and unmap a region from the address space in the current task.
- int l4re_rm_detach_ds (void ∗addr, l4re_ds_t ∗ds) L4_NOTHROW

  Detach and unmap a region and return affected dataspace in the current task.
- int l4re_rm_detach_unmap (l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW

  Detach and unmap in specified task.
- int l4re_rm_detach_ds_unmap (void ∗addr, l4re_ds_t ∗ds, l4_cap_idx_t task) L4_NOTHROW

  Detach and unmap in specified task.
- int l4re_rm_find (l4_addr_t ∗addr, unsigned long ∗size, l4re_rm_offset_t ∗offset, l4re_rm_flags_t ∗flags, l4re_ds_t ∗m) L4_NOTHROW

  Find a region given an address and size.
- void l4re_rm_show_lists (void) L4_NOTHROW

  Dump region map internal data structures.
- int l4re_rm_reserve_area_srv (l4_cap_idx_t rm, l4_addr_t ∗start, unsigned long size, l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW
- int l4re_rm_free_area_srv (l4_cap_idx_t rm, l4_addr_t addr) L4_NOTHROW
- int l4re_rm_attach_srv (l4_cap_idx_t rm, void ∗∗start, unsigned long size, l4re_rm_flags_t flags, l4re_ds_t mem, l4re_rm_offset_t offs, unsigned char align) L4_NOTHROW
- int l4re_rm_detach_srv (l4_cap_idx_t rm, l4_addr_t addr, l4re_ds_t ∗ds, l4_cap_idx_t task) L4_NOTHROW
- int l4re_rm_find_srv (l4_cap_idx_t rm, l4_addr_t ∗addr, unsigned long ∗size, l4re_rm_offset_t ∗offset, l4re↩_rm_flags_t ∗flags, l4re_ds_t ∗m) L4_NOTHROW
- void l4re_rm_show_lists_srv (l4_cap_idx_t rm) L4_NOTHROW

  Dump region map internal data structures.

13.99.1 Detailed Description

Region map C interface.

13.99.2 Enumeration Type Documentation

13.99.2.1 l4re_rm_flags_values

```
enum l4re_rm_flags_values
```

Flags for region operations.

Enumerator

L4RE_RM_F_R	Region is read-only.
L4RE_RM_F_NO_ALIAS	The region contains exclusive memory that is not mapped anywhere else.
L4RE_RM_F_PAGER	Region has a pager.
L4RE_RM_F_RESERVED	Region is reserved (blocked)
L4RE_RM_CACHING_SHIFT	Start of region mapper cache bits.
L4RE_RM_F_CACHING	Mask of all region manager cache bits.
L4RE_RM_REGION_FLAGS	Mask of all region flags.
L4RE_RM_F_CACHE_NORMAL	Cache bits for normal cacheable memory.
L4RE_RM_F_CACHE_BUFFERED	Cache bits for buffered (write combining) memory.
L4RE_RM_F_CACHE_UNCACHED	Cache bits for uncached memory.
L4RE_RM_F_SEARCH_ADDR	Search for a suitable address range.
L4RE_RM_F_IN_AREA	Search only in area, or map into area.
L4RE_RM_F_EAGER_MAP	Eagerly map the attached data space in.
L4RE_RM_F_ATTACH_FLAGS	Mask of all attach flags.

Definition at line 40 of file rm.h.

13.99.3 Function Documentation

13.99.3.1 l4re_rm_attach()

```
int l4re_rm_attach (
            void ** start,
            unsigned long size,
            l4re_rm_flags_t flags,
            l4re_ds_t mem,
            l4re_rm_offset_t offs,
            unsigned char align )  [inline]
```

Attach a data space to a region.

Parameters

in,out	*start*	Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If L4Re::Rm::F::Search_addr is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If L4Re::Rm::F::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) to attach the data space to.
	size	Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size.
	flags	The flags control how and with which rights the dataspace is attached to the region. See L4Re::Rm::F::Attach_flags and L4Re::Rm::F::Region_flags. The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the `F::Eager_map` flag is set this function may also return L4Re::Dataspace::map error codes if the mapping fails.
	mem	Data space.
	offs	Offset into the data space to use.
	align	Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used.

Return values

0	Success
-L4_ENOENT	No area could be found (see L4Re::Rm::F::In_area)
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAVAIL	The given address is not available.
<0	IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see L4Re::Rm::find).

See also

L4Re::Rm::attach

This function is using the L4::Env::env()->rm() service.

Examples

examples/libs/l4re/c/ma+rm.c.

Definition at line 284 of file rm.h.

References l4re_rm_attach_srv().

Here is the call graph for this function:

13.99.3.2 l4re_rm_attach_srv()

```
int l4re_rm_attach_srv (
            l4_cap_idx_t rm,
            void ** start,
            unsigned long size,
            l4re_rm_flags_t flags,
            l4re_ds_t mem,
            l4re_rm_offset_t offs,
            unsigned char align )
```

See also

L4Re::Rm::attach

Referenced by l4re_rm_attach().

Here is the caller graph for this function:

13.99.3.3 l4re_rm_detach()

```
int l4re_rm_detach (
            void * addr )  [inline]
```

Detach and unmap a region from the address space in the current task.

Parameters

addr	Address of the region to detach.

Return values

L4Re::Rm::Detach_result	On success.
-L4_ENOENT	No region found.
<0	IPC errors

Frees a region in the virtual address space given by addr. The corresponding part of the address space is now available again.

Also

See also

L4Re::Rm::detach

This function is using the L4::Env::env()->rm() service.

Definition at line 294 of file rm.h.

References l4re_rm_detach_srv().

Here is the call graph for this function:

13.99.3.4 l4re_rm_detach_ds()

```
int l4re_rm_detach_ds (
            void * addr,
            l4re_ds_t * ds )  [inline]
```

Detach and unmap a region and return affected dataspace in the current task.

Parameters

	addr	Address of the region to detach.
out	*ds*	Returns dataspace that is affected.

Return values

L4Re::Rm::Detach_result	On success.
-L4_ENOENT	No region found.
<0	IPC errors

Frees a region in the virtual address space given by addr. The corresponding part of the address space is now available again.

Also

See also

L4Re::Rm::detach

This function is using the L4::Env::env()->rm() service.

Examples

examples/libs/l4re/c/ma+rm.c.

Definition at line 307 of file rm.h.

References l4re_rm_detach_srv().

Here is the call graph for this function:

13.99.3.5 l4re_rm_detach_ds_unmap()

```
int l4re_rm_detach_ds_unmap (
            void * addr,
            l4re_ds_t * ds,
            l4_cap_idx_t task )  [inline]
```

Detach and unmap in specified task.

Parameters

addr	Address of the region to detach.

Return values

ds	Returns dataspace that is affected.

Parameters

task	Task to unmap pages from, specify L4_INVALID_CAP to not unmap

Returns

0 on success, <0 on error

Also

See also

L4Re::Rm::detach

This function is using the L4::Env::env()->rm() service.

Definition at line 314 of file rm.h.

References l4re_rm_detach_srv().

Here is the call graph for this function:

13.99.3.6 l4re_rm_detach_srv()

```
int l4re_rm_detach_srv (
            l4_cap_idx_t rm,
            l4_addr_t addr,
            l4re_ds_t * ds,
            l4_cap_idx_t task )
```

See also

L4Re::Rm::detach

Referenced by l4re_rm_detach(), l4re_rm_detach_ds(), l4re_rm_detach_ds_unmap(), and l4re_rm_detach_unmap().

Here is the caller graph for this function:

13.99.3.7 l4re_rm_detach_unmap()

```
int l4re_rm_detach_unmap (
            l4_addr_t addr,
            l4_cap_idx_t task ) [inline]
```

Detach and unmap in specified task.

Parameters

addr	Address of the region to detach.
task	Task to unmap pages from, specify L4_INVALID_CAP to not unmap

Returns

0 on success, <0 on error

Also

See also

L4Re::Rm::detach

This function is using the L4::Env::env()->rm() service.

Definition at line 301 of file rm.h.

References l4re_rm_detach_srv().

Here is the call graph for this function:

13.99.3.8 l4re_rm_find()

```
int l4re_rm_find (
            l4_addr_t * addr,
            unsigned long * size,
            l4re_rm_offset_t * offset,
            l4re_rm_flags_t * flags,
            l4re_ds_t * m )  [inline]
```

Find a region given an address and size.

Parameters

in,out	*addr*	Address to look for. Returns the start address of the found region.
in,out	*size*	Size of the area to look for (in bytes). Returns the size of the found region (in bytes).
out	*offset*	Offset at the beginning of the region within the associated dataspace.
out	*flags*	Region flags, see F::Region_flags (and F::In_area).
out	*m*	Associated dataspace or paging service.

Return values

0	Success
-L4_EPERM	Operation not allowed.
-L4_ENOENT	No region found.
<0	IPC errors

This function returns the properties of the region that contains the area described by the addr and size parameter. If no such region is found but a reserved area, the area is returned and F::In_area is set in `flags`. Note, in the case of an area the `offset` and `m` return values are invalid.

size-out

```
              /            \
             /              \
         addr-out            \
            ^_________________\
------------|----------------|------------------
|           | Region         |       Dataspace |
|           |_______|___|____|                 |
------------|-------|---|-----------------------
 \         /        |\ /
  \       /         | |> size-in
  offset-out        |
                    |> addr-in
```

Note

The value of the size input parameter should be 1 to assure that a region can be determined unambiguously.

See also

L4Re::Rm::find

Definition at line 321 of file rm.h.

References l4re_rm_find_srv().

Here is the call graph for this function:

13.99.3.9 l4re_rm_find_srv()

```
int l4re_rm_find_srv (
            l4_cap_idx_t rm,
            l4_addr_t * addr,
            unsigned long * size,
            l4re_rm_offset_t * offset,
            l4re_rm_flags_t * flags,
            l4re_ds_t * m )
```

See also

L4Re::Rm::find

Referenced by l4re_rm_find().

Here is the caller graph for this function:

13.99.3.10 l4re_rm_free_area()

```
int l4re_rm_free_area (
            l4_addr_t addr ) [inline]
```

Free an area from the region map.

Parameters

addr	An address within the area to free.

Return values

0	Success
-L4_ENOENT	No area found.
<0	IPC errors

Note

The data spaces that are attached to that area are not detached by this operation.

See also

reserve_area() for more information about areas.

L4Re::Rm::free_area

This function is using the L4::Env::env()->rm() service.

Definition at line 278 of file rm.h.

References l4re_rm_free_area_srv().

Here is the call graph for this function:

13.99.3.11 l4re_rm_free_area_srv()

```
int l4re_rm_free_area_srv (
            l4_cap_idx_t rm,
            l4_addr_t addr )
```

See also

L4Re::Rm::free_area

Referenced by l4re_rm_free_area().

Here is the caller graph for this function:

13.99.3.12 l4re_rm_reserve_area()

```
int l4re_rm_reserve_area (
            l4_addr_t * start,
            unsigned long size,
            l4re_rm_flags_t flags,
            unsigned char align )  [inline]
```

Reserve the given area in the region map.

Parameters

in,out	*start*	The virtual start address of the area to reserve. Returns the start address of the area.
	size	The size of the area to reserve (in bytes).
	flags	Flags for the reserved area (see L4Re::Rm::F::Region_flags and L4Re::Rm::F::Attach_flags).
	align	Alignment of area if searched as bits (log2 value).

Return values

0	Success
-L4_EADDRNOTAVAIL	The given area cannot be reserved.
<0	IPC errors

This function reserves an area within the virtual address space managed by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = L4Re::Rm::F::Reserved), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the L4Re::Rm::F::In_area flag and a start address within the area itself.

Note

When searching for a free place in the virtual address space (with *flags* = L4Re::Rm::F::Search_addr), the space between *start* and the end of the virtual address space is searched.

See also

L4Re::Rm::reserve_area

This function is using the L4::Env::env()->rm() service.

Definition at line 270 of file rm.h.

References l4re_rm_reserve_area_srv().

Here is the call graph for this function:

13.99.3.13 l4re_rm_reserve_area_srv()

```
int l4re_rm_reserve_area_srv (
            l4_cap_idx_t rm,
            l4_addr_t * start,
            unsigned long size,
            l4re_rm_flags_t flags,
            unsigned char align )
```

See also

L4Re::Rm::reserve_area

Referenced by l4re_rm_reserve_area().

Here is the caller graph for this function:

13.99.3.14 l4re_rm_show_lists()

```
void l4re_rm_show_lists (
            void ) [inline]
```

Dump region map internal data structures.

This function is using the L4::Env::env()->rm() service.

Definition at line 329 of file rm.h.

References l4re_rm_show_lists_srv().

Here is the call graph for this function:

13.100 Scheduler

C interface of the Scheduler kernel object, see L4::Scheduler for the C interface.

Collaboration diagram for Scheduler:

Data Structures

- struct l4_sched_cpu_set_t

  CPU sets.
- struct l4_sched_param_t

  Scheduler parameter set.

Typedefs

- typedef struct l4_sched_cpu_set_t l4_sched_cpu_set_t

  CPU sets.
- typedef struct l4_sched_param_t l4_sched_param_t

  Scheduler parameter set.

Enumerations

- enum L4_scheduler_classes { L4_SCHEDULER_CLASS_FIXED_PRIO = 1UL << 1 , L4_SCHEDULER_CLASS_WFQ = 1UL << 2 }

  Supported scheduler classes.
- enum L4_scheduler_ops { L4_SCHEDULER_INFO_OP = 0UL , L4_SCHEDULER_RUN_THREAD_OP = 1UL , L4_SCHEDULER_IDLE_TIME_OP = 2UL }

  Operations on the Scheduler object.

Functions

- l4_sched_cpu_set_t l4_sched_cpu_set (l4_umword_t offset, unsigned char granularity, l4_umword_t map=1) L4_NOTHROW
- l4_msgtag_t l4_scheduler_info (l4_cap_idx_t scheduler, l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus) L4_NOTHROW

  Get scheduler information.
- l4_sched_param_t l4_sched_param (unsigned prio, l4_umword_t quantum=0) L4_NOTHROW

  Construct scheduler parameter.
- l4_msgtag_t l4_scheduler_run_thread (l4_cap_idx_t scheduler, l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW

  Run a thread on a Scheduler.
- l4_msgtag_t l4_scheduler_idle_time (l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus, l4_kernel_clock_t *us) L4_NOTHROW

  Query the idle time (in µs) of a CPU.
- int l4_scheduler_is_online (l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW

  Query if a CPU is online.

13.100.1 Detailed Description

C interface of the Scheduler kernel object, see L4::Scheduler for the C interface.

The Scheduler interface allows a client to manage CPU resources. The API provides functions to query scheduler information, check the online state of CPUs, query CPU idle time and to start threads on defined CPU sets.

The scheduler offers a virtual device IRQ which triggers when the number of online cores changes, e.g. due to hotplug events. In contrast to hardware IRQs, this IRQ implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

It depends on the platform, which hotplug events actually trigger the IRQ. Many platforms only support triggering the IRQ when a CPU core different from the boot CPU goes online.

Include File

```
#include <l4/sys/scheduler.h>
```

13.100.2 Enumeration Type Documentation

13.100.2.1 L4_scheduler_classes

```
enum L4_scheduler_classes
```

Supported scheduler classes.

Enumerator

L4_SCHEDULER_CLASS_FIXED_PRIO	Fixed-priority scheduler.
L4_SCHEDULER_CLASS_WFQ	Weighted fair queuing scheduler.

Definition at line 57 of file scheduler.h.

13.100.2.2 L4_scheduler_ops

```
enum L4_scheduler_ops
```

Operations on the Scheduler object.

Enumerator

L4_SCHEDULER_INFO_OP	Query infos about the scheduler.
L4_SCHEDULER_RUN_THREAD_OP	Run a thread on this scheduler.
L4_SCHEDULER_IDLE_TIME_OP	Query idle time for the scheduler.

Definition at line 234 of file scheduler.h.

13.100.3 Function Documentation

13.100.3.1 l4_sched_cpu_set()

```
l4_sched_cpu_set_t l4_sched_cpu_set (
            l4_umword_t offset,
            unsigned char granularity,
            l4_umword_t map = 1 )  [inline]
```

Parameters

offset	Offset. Must be a multiple of 2^granularity.
granularity	Granularity in log2 notation.
map	Bitmap of CPUs, defaults to 1 in C++.

Returns

CPU set.

Examples

examples/sys/migrate/thread_migrate.cc.

Definition at line 244 of file scheduler.h.

References l4_sched_cpu_set_t::gran_offset, and l4_sched_cpu_set_t::map.

Referenced by l4_sched_param().

Here is the caller graph for this function:

13.100.3.2 l4_scheduler_idle_time()

```
l4_msgtag_t l4_scheduler_idle_time (
            l4_cap_idx_t scheduler,
            l4_sched_cpu_set_t const * cpus,
            l4_kernel_clock_t * us )  [inline]
```

Query the idle time (in µs) of a CPU.

Parameters

	scheduler	Scheduler object.
	cpus	Set of CPUs to query. Only the idle time of the first selected CPU in `cpus.map` is queried.
`out`	*us*	Idle time of queried CPU in µs.

Return values

0	Success.
-L4_EINVAL	Invalid CPU requested in cpu set.

This function retrieves the idle time in µs of the first selected CPU in `cpus.map`. The idle time is the accumulated time a CPU has spent in the idle thread since its last reset. To calculate a load estimate `l` one has to retrieve the idle time at the beginning (`i1`) and the end (`i2`) of a known time interval `t`. The load is then calculated as l = 1 - (i2 - i1)/t.

The idle time is only defined for online CPUs. Reading the idle time from offline CPUs is undefined and may result in either getting -L4_EINVAL or calculating an estimated (incorrect) load of 1.

Note

The idle time statistics of remote CPUs is updated on context switch events only, hence may not be up-to-date when requested cross-CPU. To get up-to-date idle time you should use a thread running on the same CPU of which the idle time is requested.

Definition at line 368 of file scheduler.h.

13.100.3.3 l4_scheduler_info()

```
l4_msgtag_t l4_scheduler_info (
            l4_cap_idx_t scheduler,
            l4_umword_t * cpu_max,
            l4_sched_cpu_set_t * cpus )  [inline]
```

Get scheduler information.

Parameters

	scheduler	Scheduler object.
`out`	*cpu_max*	Maximum number of CPUs ever available. Optional, can be NULL.
`in,out`	*cpus*	*cpus.offset* is first CPU of interest. *cpus.granularity* (see l4_sched_cpu_set_t). *cpus.map* Bitmap of online CPUs. Pass NULL if this information is not required.

Return values

0	Success.
-L4_EINVAL	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 346 of file scheduler.h.

13.100.3.4 l4_scheduler_is_online()

```
int l4_scheduler_is_online (
            l4_cap_idx_t scheduler,
            l4_umword_t cpu )  [inline]
```

Query if a CPU is online.

Parameters

scheduler	Scheduler object.
cpu	CPU number whose online status should be queried.

Return values

true	The CPU is online.
false	The CPU is offline

Definition at line 375 of file scheduler.h.

13.100.3.5 l4_scheduler_run_thread()

```
l4_msgtag_t l4_scheduler_run_thread (
            l4_cap_idx_t scheduler,
            l4_cap_idx_t thread,
            l4_sched_param_t const * sp )  [inline]
```

Run a thread on a Scheduler.

Parameters

scheduler	Scheduler object.
thread	Capability of the thread to run.
sp	Scheduling parameters.

Return values

0	Success.

Return values

-L4_EINVAL	Invalid size of the scheduling parameter.

This function launches a thread on a CPU determined by the scheduling parameter `sp.affinity`. A thread can be intentionally stopped by migrating it on an offline or an invalid CPU. The thread is only guaranteed to run if the CPU it is migrated to is currently online.

Note

A scheduler may impose a policy with regard to selecting CPUs. However the scheduler is required to ensure the following two properties:

- Two threads with disjoint CPU sets must be scheduled to different physical CPUs.
- Two threads with a single identical CPU selected in the CPU set must be scheduled to the same physical CPU.

Examples

examples/sys/aliens/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 361 of file scheduler.h.

13.101 Server-Side IPC framework

Server-Side framework for implementing object-oriented servers.

Data Structures

- class L4::Ipc_svr::Server_iface

  Interface for server-loop related functions.
- class L4::Basic_registry

  This registry returns the corresponding server object based on the label of an Ipc_gate.
- struct L4::Ipc_svr::Ignore_errors

  Mix in for LOOP_HOOKS to ignore IPC errors.
- struct L4::Ipc_svr::Default_timeout

  Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.
- struct L4::Ipc_svr::Compound_reply

  Mix in for LOOP_HOOKS to always use compound reply and wait.
- struct L4::Ipc_svr::Default_setup_wait

  Mix in for LOOP_HOOKS for setup_wait no op.
- class L4::Ipc_svr::Br_manager_no_buffers

  Empty implementation of Server_iface.
- struct L4::Ipc_svr::Default_loop_hooks

  Default LOOP_HOOKS.
- class L4::Server< LOOP_HOOKS >

  Basic server loop for handling client requests.
- class L4::Server_object

Abstract server object to be used with L4::Server and L4::Basic_registry.

- struct L4::Server_object_t< IFACE, BASE >

  Base class (template) for server implementing server objects.
- struct L4::Server_object_x< Derived, IFACE, BASE >

  Helper class to implement p_dispatch based server objects.
- struct L4::Irq_handler_object

  Server object base class for handling IRQ messages.
- class L4::Ipc_svr::Timeout

  Callback interface for Timeout_queue.
- class L4::Ipc_svr::Timeout_queue

  Timeout queue to be used in l4re server loop.
- class L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >

  Loop hooks mixin for integrating a timeout queue into the server loop.

Enumerations

- enum L4::Ipc_svr::Reply_mode { L4::Ipc_svr::Reply_compound , L4::Ipc_svr::Reply_separate }

  Reply mode for server loop.

13.101.1 Detailed Description

Server-Side framework for implementing object-oriented servers.

´

13.101.2 Enumeration Type Documentation

13.101.2.1 Reply_mode

`enum L4::Ipc_svr::Reply_mode`

Reply mode for server loop.

The reply mode specifies if the server loop shall do a compound reply and wait operation (Reply_compound), which is the most performant method. Note, setup_wait() is called before the reply. The other way is to call reply and wait separately and call setup_wait in between.

The actual mode is determined by the return value of the before_reply() hook in the LOOP_HOOKS of L4::Server.

Enumerator

Reply_compound	Server shall use a compound reply and wait (fast).
Reply_separate	Server shall call reply and wait separately.

Definition at line 50 of file ipc_server_loop.

13.102 Shared Memory Library

L4SHM provides a shared memory infrastructure that establishes a shared memory area between multiple parties and uses a fast notification mechanism.

Collaboration diagram for Shared Memory Library:

Modules

- Chunks
- Signals

Functions

- long l4shmc_create (const char *shmc_name, l4_umword_t shm_size)

  Create a shared memory area.
- long l4shmc_attach (const char *shmc_name, l4shmc_area_t *shmarea)

  Attach to a shared memory area.
- long l4shmc_attach_to (const char *shmc_name, l4_umword_t timeout_ms, l4shmc_area_t *shmarea)

  Attach to a shared memory area, with limited waiting.
- long l4shmc_connect_chunk_signal (l4shmc_chunk_t *chunk, l4shmc_signal_t *signal)

  Connect a signal with a chunk.
- long l4shmc_area_size (l4shmc_area_t *shmarea)

  Get size of shared memory area.
- long l4shmc_area_size_free (l4shmc_area_t *shmarea)

  Get free size of shared memory area.
- long l4shmc_area_overhead (void)

  Get memory overhead per area that is not available for chunks.
- long l4shmc_chunk_overhead (void)

  Get memory overhead required in addition to the chunk capacity for adding one chunk.

13.102.1 Detailed Description

L4SHM provides a shared memory infrastructure that establishes a shared memory area between multiple parties and uses a fast notification mechanism.

A shared memory area consists of chunks and signals. A chunk is a defined chunk of memory within the memory area with a maximum size. A chunk is filled (written) by a producer and read by a consumer. When a producer has finished writing to the chunk it signals a data ready notification to the consumer.

A consumer attaches to a chunk and waits for the producer to fill the chunk. After reading out the chunk it marks the chunk free again.

A shared memory area can have multiple chunks.

The interface is divided in three roles.

- The master role, responsible for setting up a shared memory area.
- A producer, generating data into a chunk
- A consumer, receiving data.

A signal can be connected with a chunk or can be used independently (e.g. for multiple chunks).

13.102.2 Function Documentation

13.102.2.1 l4shmc_area_overhead()

```
long l4shmc_area_overhead (
            void  )
```

Get memory overhead per area that is not available for chunks.

Returns

size of the overhead in bytes

13.102.2.2 l4shmc_area_size()

```
long l4shmc_area_size (
            l4shmc_area_t * shmarea )  [inline]
```

Get size of shared memory area.

Parameters

shmarea	Shared memory area.

Return values

>0	Size of the shared memory area.
<0	Error.

13.102.2.3 l4shmc_area_size_free()

```
long l4shmc_area_size_free (
            l4shmc_area_t * shmarea )
```

Get free size of shared memory area.

To get the max size to pass to l4shmc_add_chunk, substract l4shmc_chunk_overhead().

Parameters

shmarea	Shared memory area.

Return values

0	Free capacity in the area.
<0	Error.

13.102.2.4 l4shmc_attach()

```
long l4shmc_attach (
            const char * shmc_name,
            l4shmc_area_t * shmarea )  [inline]
```

Attach to a shared memory area.

Parameters

	shmc_name	Name of the shared memory area.
out	*shmarea*	Pointer to shared memory area descriptor to be filled with information for the shared memory area.

Return values

0	Success.

Return values

<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.102.2.5 l4shmc_attach_to()

```
long l4shmc_attach_to (
            const char * shmc_name,
            l4_umword_t timeout_ms,
            l4shmc_area_t * shmarea )
```

Attach to a shared memory area, with limited waiting.

Parameters

	shmc_name	Name of the shared memory area.
	timeout_ms	Timeout to wait for shm area in milliseconds.
out	*shmarea*	Pointer to shared memory area descriptor to be filled with information for the shared memory area.

Return values

0	Success.
<0	Error.

13.102.2.6 l4shmc_chunk_overhead()

```
long l4shmc_chunk_overhead (
            void )
```

Get memory overhead required in addition to the chunk capacity for adding one chunk.

Returns

size of the overhead in bytes

13.102.2.7 l4shmc_connect_chunk_signal()

```
long l4shmc_connect_chunk_signal (
            l4shmc_chunk_t * chunk,
            l4shmc_signal_t * signal )
```

Connect a signal with a chunk.

Parameters

chunk	Chunk to attach the signal to.
signal	Signal to attach.

Returns

0 on success, <0 on error

Examples

examples/libs/shmc/prodcons.c.

13.102.2.8 l4shmc_create()

```
long l4shmc_create (
            const char * shmc_name,
            l4_umword_t shm_size )
```

Create a shared memory area.

Parameters

shmc_name	Name of the shared memory area.
shm_size	Size of the whole shared memory area.

Return values

0	Success.
<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.103 Sigma0 API

Sigma0 API bindings.

Collaboration diagram for Sigma0 API:

Modules

- Internal constants

  Internal sigma0 definitions.

Files

- file sigma0.h

  Sigma0 interface.

Enumerations

- enum l4sigma0_return_flags_t {
  L4SIGMA0_OK , L4SIGMA0_NOTALIGNED , L4SIGMA0_IPCERROR , L4SIGMA0_NOFPAGE ,
  L4SIGMA0_4 , **L4SIGMA0_5** , L4SIGMA0_SMALLERFPAGE }

  Return flags of libsigma0 functions.

Functions

- l4_kernel_info_t ∗ l4sigma0_map_kip (l4_cap_idx_t sigma0, void ∗addr, unsigned log2_size)

  Map the kernel info page from sigma0 to addr.
- int l4sigma0_map_mem (l4_cap_idx_t sigma0, l4_addr_t phys, l4_addr_t virt, l4_addr_t size)

  Request a memory mapping from sigma0.
- int l4sigma0_map_iomem (l4_cap_idx_t sigma0, l4_addr_t phys, l4_addr_t virt, l4_addr_t size, int cached)

  Request IO memory from sigma0.
- int l4sigma0_map_anypage (l4_cap_idx_t sigma0, l4_addr_t map_area, unsigned log2_map_size, l4_addr_t ∗base, unsigned sz)

  Request an arbitrary free page of RAM.
- void l4sigma0_debug_dump (l4_cap_idx_t sigma0)

  Request sigma0 to dump internal debug information.
- int l4sigma0_new_client (l4_cap_idx_t sigma0, l4_cap_idx_t gate)

  Create a new IPC gate for a new Sigma0 client.
- char const ∗ l4sigma0_map_errstr (int err)

  Get user readable error messages for the return codes.

13.103.1 Detailed Description

Sigma0 API bindings.

Convenience bindings for the Sigma0 protocol.

13.103.2 Enumeration Type Documentation

13.103.2.1 l4sigma0_return_flags_t

```
enum l4sigma0_return_flags_t
```

Return flags of libsigma0 functions.

Enumerator

L4SIGMA0_OK	Ok.
L4SIGMA0_NOTALIGNED	Phys, virt or size not aligned.
L4SIGMA0_IPCERROR	IPC error.
L4SIGMA0_NOFPAGE	No fpage received.
L4SIGMA0_SMALLERFPAGE	Superpage requested but smaller flexpage received.

Definition at line 80 of file sigma0.h.

13.103.3 Function Documentation

13.103.3.1 l4sigma0_debug_dump()

```
void l4sigma0_debug_dump (
            l4_cap_idx_t sigma0 )
```

Request sigma0 to dump internal debug information.

Parameters

sigma0	Capability selector for the sigma0 gate.

The debug information, such as internal memory maps, as well as statistics about the internal allocators is dumped to the kernel debugger.

13.103.3.2 l4sigma0_map_anypage()

```
int l4sigma0_map_anypage (
            l4_cap_idx_t sigma0,
            l4_addr_t map_area,
            unsigned log2_map_size,
            l4_addr_t * base,
            unsigned sz )
```

Request an arbitrary free page of RAM.

Parameters

	sigma0	Capability selector for the sigma0 gate.
	map_area	The base address of the local virtual memory area where the page should be mapped.
	log2_map_size	The size of the requested page log 2 (the size in bytes is 2^log2_map_size). This must be at least the minimal page size. By specifing larger sizes the largest possible hardware page size will be used.
out	*base*	Physical address of the page received (i.e. the send base of the received mapping if any).
	sz	Size to map by the server in 2^sz bytes.

Return values

0	Success.
-L4SIGMA0_IPCERROR	IPC error.
-L4SIGMA0_NOFPAGE	No fpage received.

This function requests arbitrary free memory from sigma0. It should be used whenever spare memory is needed, instead of requesting specific physical memory with l4sigma0_map_mem().

See l4sigma0_map_errstr() to get a description of the return value.

13.103.3.3 l4sigma0_map_errstr()

```
char const * l4sigma0_map_errstr (
            int err )  [inline]
```

Get user readable error messages for the return codes.

Parameters

err	The error code reported by the *map* functions.

Returns

A string containing the error message.

Definition at line 227 of file sigma0.h.

13.103.3.4 l4sigma0_map_iomem()

```
int l4sigma0_map_iomem (
            l4_cap_idx_t sigma0,
            l4_addr_t phys,
            l4_addr_t virt,
            l4_addr_t size,
            int cached )
```

Request IO memory from sigma0.

Parameters

sigma0	Capability selector for the sigma0 gate.
phys	The physical address to be requested (page aligned).
virt	The virtual address where the memory should be mapped to (page aligned).
size	The size of the IO memory area to be mapped (multiple of page size)
cached	Requests cacheable IO memory if 1 and uncached if 0.

Return values

0	Success.
-L4SIGMA0_NOTALIGNED	`phys`, `virt`, or `size` are not aligned.
-L4SIGMA0_IPCERROR	IPC error.
-L4SIGMA0_NOFPAGE	No fpage received.

This function is similar to l4sigma0_map_mem(), the difference is that it requests IO memory. IO memory is everything that is not known to be normal RAM. Also ACPI tables or the BIOS memory is treated as IO memory.

See l4sigma0_map_errstr() to get a description of the return value.

13.103.3.5 l4sigma0_map_kip()

```
l4_kernel_info_t* l4sigma0_map_kip (
            l4_cap_idx_t sigma0,
            void * addr,
            unsigned log2_size )
```

Map the kernel info page from sigma0 to addr.

Parameters

sigma0	Capability selector for the sigma0 gate.
addr	Start of the receive window to receive KIP in.
log2_size	Size of the receive window to receive KIP in.

Returns

Address KIP was mapped to, 0 indicates an error.

13.103.3.6 l4sigma0_map_mem()

```
int l4sigma0_map_mem (
            l4_cap_idx_t sigma0,
            l4_addr_t phys,
            l4_addr_t virt,
            l4_addr_t size )
```

Request a memory mapping from sigma0.

Parameters

sigma0	Capability selector for the sigma0 gate.
phys	The physical address of the requested page (must be at least aligned to the minimum page size).
virt	The virtual address where the paged should be mapped in the local address space (must be at least aligned to the minimum page size).
size	The size of the requested page, this must be a multiple of the minimum page size.

Return values

0	Success.
-L4SIGMA0_NOTALIGNED	`phys`, `virt`, or `size` are not aligned.
-L4SIGMA0_IPCERROR	IPC error.
-L4SIGMA0_NOFPAGE	No fpage received.

This function only maps normal RAM. To map other memory, use l4sigma0_map_iomem(). See also there for the distinction between both memory types.

This is the direct method to request memory from sigma0. There is also the indirect method where sigma0 will answer page faults with a mapping that is one-to-one between the faulting virtual page and the backing physical page. See L4::Pager::page_fault(). For an overview of the memory hierarchy, see Memory management - Data Spaces and the Region Map.

See l4sigma0_map_errstr() to get a description of the return value.

13.103.3.7 l4sigma0_new_client()

```
int l4sigma0_new_client (
            l4_cap_idx_t sigma0,
            l4_cap_idx_t gate )
```

Create a new IPC gate for a new Sigma0 client.

Parameters

	sigma0	Capability selector for the sigma0 gate.
`out`	*gate*	Capability selector to use for the new gate.

Return values

0	Success.

Return values

-L4SIGMA0_IPCERROR	IPC error.
-L4SIGMA0_NOFPAGE	No fpage received.

The result is identical to sending a factory create message (see L4::Factory.create()) with object L4_PROTO_SIGMA0 to `sigma0`.

13.104 Signals

Collaboration diagram for Signals:

Modules

- Consumer
- Producer

Functions

- long l4shmc_add_signal (l4shmc_area_t ∗shmarea, const char ∗signal_name, l4shmc_signal_t ∗signal)

  Add a signal for the shared memory area.
- long l4shmc_attach_signal (l4shmc_area_t ∗shmarea, const char ∗signal_name, l4_cap_idx_t thread, l4shmc_signal_t ∗signal)

  Attach to signal.
- long l4shmc_attach_signal_to (l4shmc_area_t ∗shmarea, const char ∗signal_name, l4_cap_idx_t thread, l4_umword_t timeout_ms, l4shmc_signal_t ∗signal)

  Attach to signal, with timeout.
- long l4shmc_get_signal_to (l4shmc_area_t ∗shmarea, const char ∗signal_name, l4_umword_t timeout_ms, l4shmc_signal_t ∗signal)

  Get signal object from the shared memory area.
- l4_cap_idx_t l4shmc_signal_cap (l4shmc_signal_t ∗signal)

  Get the signal capability of a signal.
- long l4shmc_check_magic (l4shmc_chunk_t ∗chunk)

  Check magic value of a chunk.

13.104.1 Detailed Description

13.104.2 Function Documentation

13.104.2.1 l4shmc_add_signal()

```
long l4shmc_add_signal (
            l4shmc_area_t * shmarea,
            const char * signal_name,
            l4shmc_signal_t * signal )
```

Add a signal for the shared memory area.

Parameters

	shmarea	The shared memory area to put the chunk in.
	signal_name	Name of the signal.
out	*signal*	Signal structure to fill in.

Return values

0	Success.
<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.104.2.2 l4shmc_attach_signal()

```
long l4shmc_attach_signal (
            l4shmc_area_t * shmarea,
            const char * signal_name,
            l4_cap_idx_t thread,
            l4shmc_signal_t * signal )  [inline]
```

Attach to signal.

Parameters

	shmarea	Shared memory area.
	signal_name	Name of the signal.
	thread	Thread capability index to attach the signal to.
out	*signal*	Signal data structure to fill.

Return values

0	Success.
<0	Error.

13.104.2.3 l4shmc_attach_signal_to()

```
long l4shmc_attach_signal_to (
            l4shmc_area_t * shmarea,
            const char * signal_name,
            l4_cap_idx_t thread,
            l4_umword_t timeout_ms,
            l4shmc_signal_t * signal )
```

Attach to signal, with timeout.

Parameters

	shmarea	Shared memory area.
	signal_name	Name of the signal.
	thread	Thread capability index to attach the signal to.
	timeout_ms	Timeout in milliseconds to wait for the chunk to appear in the shared memory area.
out	*signal*	Signal data structure to fill.

Return values

0	Success.
<0	Error.

Examples

examples/libs/shmc/prodcons.c.

13.104.2.4 l4shmc_check_magic()

```
long l4shmc_check_magic (
            l4shmc_chunk_t * chunk )  [inline]
```

Check magic value of a chunk.

Parameters

chunk	Chunk.

Return values

0	Magic value is not valid.
>0	Chunk is ok, the magic value is valid.

13.104.2.5 l4shmc_get_signal_to()

```
long l4shmc_get_signal_to (
            l4shmc_area_t * shmarea,
            const char * signal_name,
            l4_umword_t timeout_ms,
            l4shmc_signal_t * signal )
```

Get signal object from the shared memory area.

Parameters

	shmarea	Shared memory area.
	signal_name	Name of the signal.
	timeout_ms	Timeout in milliseconds to wait for signal of a chunk to appear in the shared memory area.
out	*signal*	Signal data structure to fill.

Return values

0	Success.
<0	Error.

13.104.2.6 l4shmc_signal_cap()

```
l4_cap_idx_t l4shmc_signal_cap (
            l4shmc_signal_t * signal )  [inline]
```

Get the signal capability of a signal.

Parameters

signal	Signal.

Returns

Capability of the signal object.

13.105 Small C++ Template Library

Data Structures

- class L4::Alloc_list

  A simple list-based allocator.
- class cxx::Auto_ptr< T >

  Smart pointer with automatic deletion.
- class cxx::Bitmap_base

  Basic bitmap abstraction.
- class cxx::Bitmap< BITS >

  A static bit map.
- class cxx::List_item

  Basic list item.
- struct cxx::Pair< First, Second >

  Pair of two values.
- class cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >

  Basic slab allocator.
- class cxx::Slab< Type, Slab_size, Max_free, Alloc >

  Slab allocator for object of type `Type`.
- class cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >

  Merged slab allocator (allocators for objects of the same size are merged together).
- class cxx::Slab_static< Type, Slab_size, Max_free, Alloc >

  Merged slab allocator (allocators for objects of the same size are merged together).
- class cxx::Nothrow

  Helper type to distinguish the `oeprator new` *version that does not throw exceptions.*
- class cxx::New_allocator< _Type >

  Standard allocator based on `operator new ()` .
- class L4::String

  A null-terminated string container class.

Functions

- template<typename T1 >
  T1 cxx::min (T1 a, T1 b)

  Get the minimum of a and b.
- template<typename T1 >
  T1 cxx::max (T1 a, T1 b)

  Get the maximum of a and b.
- void ∗ operator new (size_t, void ∗mem, cxx::Nothrow const &) throw ()

  Simple placement new operator.
- void ∗ operator new (size_t, cxx::Nothrow const &) throw ()

  New operator that does not throw exceptions.
- void operator delete (void ∗, cxx::Nothrow const &) throw ()

  Delete operator complementing the new operator not throwing exceptions.

13.105.1 Detailed Description

13.105.2 Function Documentation

13.105.2.1 max()

```
template<typename T1 >
T1 cxx::max (
            T1 a,
            T1 b )  [inline]
```

Get the maximum of *a* and *b*.

Parameters

a	the first value.
b	the second value.

Definition at line 46 of file minmax.

Referenced by cxx::List_alloc::alloc_max(), L4virtio::Svr::Data_buffer::copy_to(), L4::Scheduler::info(), L4virtio::Svr::Driver_mem_list_t
L4Re::Util::Dataspace_svr::map_hook(), and L4::Type_info::Demand::operator|().

Here is the caller graph for this function:

13.105.2.2 min()

```
template<typename T1 >
T1 cxx::min (
            T1 a,
            T1 b )  [inline]
```

Get the minimum of *a* and *b*.

Parameters

a	the first value.
b	the second value.

Examples

tmpfs/lib/src/fs.cc.

Definition at line 35 of file minmax.

Referenced by cxx::List_alloc::alloc_max(), L4virtio::Svr::Data_buffer::copy_to(), L4Re::Util::Dataspace_svr::map_hook(), operator>>(), L4virtio::Svr::Data_buffer::skip(), and cxx::String::substr().

Here is the caller graph for this function:

13.105.2.3 operator new()

```
void* operator new (
            size_t ,
            void * mem,
            cxx::Nothrow const &  ) throw ( )   [inline]
```

Simple placement new operator.

Parameters

mem	the address of the memory block to place the new object.

Returns

the address given by *mem*.

Definition at line 39 of file std_alloc.

13.106 Task

C interface of the Task kernel object, see L4::Task for the C++ interface.

Collaboration diagram for Task:

Enumerations

- enum l4_unmap_flags_t { L4_FP_ALL_SPACES , L4_FP_DELETE_OBJ , L4_FP_OTHER_SPACES }

  Flags for the unmap operation.

Functions

- l4_msgtag_t l4_task_vgicc_map (l4_cap_idx_t task, l4_fpage_t vgicc_fpage) L4_NOTHROW

  Map the GIC's virtual GICC page to the task.
- l4_msgtag_t l4_task_map (l4_cap_idx_t dst_task, l4_cap_idx_t src_task, l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW

  Map resources available in the source task to a destination task.
- l4_msgtag_t l4_task_unmap (l4_cap_idx_t task, l4_fpage_t fpage, l4_umword_t map_mask) L4_NOTHROW

  Revoke rights from the task.

- l4_msgtag_t l4_task_unmap_batch (l4_cap_idx_t task, l4_fpage_t const *fpages, unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW

  Revoke rights from a task.
- l4_msgtag_t l4_task_delete_obj (l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW

  Release capability and delete object.
- l4_msgtag_t l4_task_release_cap (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW

  Release object capability.
- l4_msgtag_t l4_task_cap_valid (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW

  Check whether a capability is present (refers to an object).
- l4_msgtag_t l4_task_cap_equal (l4_cap_idx_t task, l4_cap_idx_t cap_a, l4_cap_idx_t cap_b) L4_NOTHROW

  Test whether two capabilities point to the same object with the same rights.
- l4_msgtag_t l4_task_add_ku_mem (l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW

  Add kernel-user memory.

13.106.1 Detailed Description

C interface of the Task kernel object, see L4::Task for the C++ interface.

A task represents a combination of the address spaces provided by the L4Re micro kernel. A task consists of at least a memory address space and an object address space. On IA32 there is also an IO-port address space.

Task objects are created using the Factory interface.

Include File

```
#include <l4/sys/task.h>
```

13.106.2 Enumeration Type Documentation

13.106.2.1 l4_unmap_flags_t

```
enum l4_unmap_flags_t
```

Flags for the unmap operation.

See also

L4::Task::unmap() and l4_task_unmap()

Enumerator

L4_FP_ALL_SPACES	Flag to tell the unmap operation to unmap all child mappings including the mapping in the invoked task. **Note** Object capabilities are not hierarchical – they have no children. The result of the map operation on an object capability is a copy of that capability in the object space of the destination task. An unmap operation on object capabilities is a no-op if this flag is not specified. **See also** L4::Task::unmap() l4_task_unmap()
L4_FP_DELETE_OBJ	Flag that indicates that an unmap operation on object capabilities shall try to delete the corresponding objects. **See also** L4::Task::unmap() l4_task_unmap()
L4_FP_OTHER_SPACES	Counterpart to L4_FP_ALL_SPACES, unmap only child mappings. **See also** L4::Task::unmap() l4_task_unmap()

Definition at line 157 of file consts.h.

13.106.3 Function Documentation

13.106.3.1 l4_task_add_ku_mem()

```
l4_msgtag_t l4_task_add_ku_mem (
            l4_cap_idx_t task,
            l4_fpage_t ku_mem )  [inline]
```

Add kernel-user memory.

Parameters

task	Capability selector of the task to add the memory to
ku_mem	Flexpage describing the virtual area the memory goes to.

Returns

Syscall return tag

Definition at line 418 of file task.h.

13.106.3.2 l4_task_cap_equal()

```
l4_msgtag_t l4_task_cap_equal (
            l4_cap_idx_t task,
            l4_cap_idx_t cap_a,
            l4_cap_idx_t cap_b )  [inline]
```

Test whether two capabilities point to the same object with the same rights.

Parameters

task	Capability selector of the destination task to do the lookup in
cap↩ _a	Capability selector to compare
cap↩ _b	Capability selector to compare

Returns

label contains 1 if equal, 0 if not equal

Definition at line 411 of file task.h.

13.106.3.3 l4_task_cap_valid()

```
l4_msgtag_t l4_task_cap_valid (
            l4_cap_idx_t task,
            l4_cap_idx_t cap )  [inline]
```

Check whether a capability is present (refers to an object).

Parameters

task	Task to check the capability in.
cap	Valid capability to check for presence.

Return values

tag.label()	> 0 Capability is present (refers to an object).
tag.label()	== 0 No capability present (void object).

A capability is considered present when it refers to an existing kernel object.

Precondition

`cap` must be a valid capability index (i.e. not L4_INVALID_CAP or the like).

Definition at line 405 of file task.h.

13.106.3.4 l4_task_delete_obj()

```
l4_msgtag_t l4_task_delete_obj (
            l4_cap_idx_t task,
            l4_cap_idx_t obj )  [inline]
```

Release capability and delete object.

Parameters

task	Capability selector of destination task
obj	Capability selector of object to delete

Returns

Syscall return tag

Initiates the deletion of the object if `obj` has the delete permission. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference. No error will be reported if the rights are insufficient, however, the capability is removed in all cases from the destination task.

This operation calls l4_task_unmap() with L4_FP_DELETE_OBJ.

Definition at line 384 of file task.h.

13.106.3.5 l4_task_map()

```
l4_msgtag_t l4_task_map (
            l4_cap_idx_t dst_task,
            l4_cap_idx_t src_task,
            l4_fpage_t snd_fpage,
            l4_umword_t snd_base )  [inline]
```

Map resources available in the source task to a destination task.

Parameters

dst_task	Capability selector of the destination task.
src_task	Capability selector of the source task.
snd_fpage	Send flexpage that describes an area in the address space or object space of the source task.
snd_base	Send base that describes an offset in the receive window of the destination task. The lower bits contain additional map control flags (see l4_fpage_cacheability_opt_t for memory mappings, L4_obj_fpage_ctl for object mappings, and L4_MAP_ITEM_GRANT; also see l4_map_control() and l4_map_obj_control()).

Returns

Syscall return tag.

This method allows for asynchronous transfer of capabilities, memory mappings, and IO-port mappings (on IA32) from one task to another. The receive window is the whole address space of `dst_task`.

For more information on spaces and mappings, see Spaces and Mappings. The flexpage API is described in more detail at Flex pages.

Definition at line 354 of file task.h.

Referenced by L4::Cap_base::copy(), and L4::Cap_base::move().

Here is the caller graph for this function:

13.106.3.6 l4_task_release_cap()

```
l4_msgtag_t l4_task_release_cap (
            l4_cap_idx_t task,
            l4_cap_idx_t cap )  [inline]
```

Release object capability.

Parameters

task	Capability selector of destination task
cap	Capability selector of object to release

Returns

Syscall return tag

This operation unmaps the capability from the specified task.

Note

If the reference counter of the kernel object referenced by `cap` goes down to zero, deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Definition at line 399 of file task.h.

13.106.3.7 l4_task_unmap()

```
l4_msgtag_t l4_task_unmap (
            l4_cap_idx_t task,
            l4_fpage_t fpage,
            l4_umword_t map_mask ) [inline]
```

Revoke rights from the task.

Parameters

task	Capability selector of destination task
fpage	Flexpage that describes an area in one capability space of the destination task
map_mask	Unmap mask, see l4_unmap_flags_t

Returns

Syscall return tag

This method allows to revoke rights from the destination task. For a flex page describing IO ports or memory, it also revokes rights from all the tasks that got the rights delegated from the destination task (i.e., this operation does a recursive rights revocation). If the set of rights is empty after the revocation, the capability is unmapped.

Note

If the reference counter of a kernel object referenced in `fpage` goes down to zero (as a result of deleting capabilities), the deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Examples

examples/sys/utcb-ipc/main.c.

Definition at line 361 of file task.h.

Referenced by L4Re::Util::Cap_alloc_base::free(), L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free(), and L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::release().

Here is the caller graph for this function:

13.106.3.8 l4_task_unmap_batch()

```
l4_msgtag_t l4_task_unmap_batch (
            l4_cap_idx_t task,
            l4_fpage_t const * fpages,
            unsigned num_fpages,
            l4_umword_t map_mask )  [inline]
```

Revoke rights from a task.

Parameters

task	Capability selector of destination task
fpages	An array of flexpages. Each item describes an area in one capability space of the destination task.
num_fpages	The size of the fpages array in elements (number of fpages sent).
map_mask	Unmap mask, see l4_unmap_flags_t

Returns

Syscall return tag

Revoke rights specified in an array of flexpages, see l4_task_unmap for details.

Precondition

The caller needs to take care that num_fpages is not bigger than L4_UTCB_GENERIC_DATA_SIZE - 2.

Definition at line 368 of file task.h.

13.106.3.9 l4_task_vgicc_map()

```
l4_msgtag_t l4_task_vgicc_map (
            l4_cap_idx_t task,
            l4_fpage_t vgicc_fpage )  [inline]
```

Map the GIC's virtual GICC page to the task.

Parameters

task	Capability selector of destination task
vgicc_fpage	Flexpage that describes an area in the address space of the destination task to map the vGICC page to

Returns

Syscall return tag

Definition at line 56 of file __task-arm.h.

References l4_utcb().

Here is the call graph for this function:

13.107 Thread

C Thread object interface, see L4::Thread for the C++ interface.

Collaboration diagram for Thread:

Modules

- Thread control

  API for Thread Control method.
- vCPU API

  vCPU API.

Enumerations

- enum L4_thread_control_flags { L4_THREAD_CONTROL_SET_PAGER = 0x0010000 , L4_THREAD_CONTROL_BIND_TASK = 0x0200000 , L4_THREAD_CONTROL_ALIEN = 0x0400000 , L4_THREAD_CONTROL_UX_NATIVE = 0x0800000 , L4_THREAD_CONTROL_SET_EXC_HANDLER = 0x1000000 }

  Flags for the thread control operation.
- enum L4_thread_control_mr_indices { L4_THREAD_CONTROL_MR_IDX_FLAGS = 0 , L4_THREAD_CONTROL_MR_IDX_PAGER = 1 , L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER = 2 , L4_THREAD_CONTROL_MR_IDX_FLAG_VALS = 4 , L4_THREAD_CONTROL_MR_IDX_BIND_UTCB = 5 , L4_THREAD_CONTROL_MR_IDX_BIND_TASK = 6 }

  Indices for the values in the message register for thread control.
- enum L4_thread_ex_regs_flags { L4_THREAD_EX_REGS_CANCEL = 0x10000UL , L4_THREAD_EX_REGS_TRIGGER_EXC = 0x20000UL }

  Flags for the thread ex-regs operation.

Functions

- l4_msgtag_t l4_thread_ex_regs (l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp, l4_umword_t flags) L4_NOTHROW

  Exchange basic thread registers.
- l4_msgtag_t l4_thread_ex_regs_u (l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp, l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW

  Exchange basic thread registers.
- l4_msgtag_t l4_thread_ex_regs_ret (l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp, l4_umword_t *flags) L4_NOTHROW

  Exchange basic thread registers and return previous values.
- l4_msgtag_t l4_thread_ex_regs_ret_u (l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp, l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW

  Exchange basic thread registers and return previous values.
- l4_msgtag_t l4_thread_yield (void) L4_NOTHROW

  Yield current time slice.
- l4_msgtag_t l4_thread_switch (l4_cap_idx_t to_thread) L4_NOTHROW

  Switch to another thread (and donate the remaining time slice).
- l4_msgtag_t l4_thread_stats_time (l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW

  Get consumed time of thread in µs.
- l4_msgtag_t l4_thread_vcpu_resume_start (void) L4_NOTHROW

  vCPU return from event handler.
- l4_msgtag_t l4_thread_vcpu_resume_commit (l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW

  Commit vCPU resume.
- l4_msgtag_t l4_thread_vcpu_control (l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW

  Enable the vCPU feature for the thread.
- l4_msgtag_t l4_thread_vcpu_control_u (l4_cap_idx_t thread, l4_addr_t vcpu_state, l4_utcb_t *utcb) L4_NOTHROW

  Enable the vCPU feature for the thread.
- l4_msgtag_t l4_thread_vcpu_control_ext (l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW

  Enable the extended vCPU feature for the thread.
- l4_msgtag_t l4_thread_vcpu_control_ext_u (l4_cap_idx_t thread, l4_addr_t ext_vcpu_state, l4_utcb_t *utcb) L4_NOTHROW

  Enable the extended vCPU feature for the thread.
- l4_msgtag_t l4_thread_register_del_irq (l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW

  Register an IRQ that will trigger upon deletion events.
- l4_msgtag_t l4_thread_modify_sender_start (void) L4_NOTHROW

  Start a thread sender modification sequence.
- int l4_thread_modify_sender_add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits, l4_msgtag_t *tag) L4_NOTHROW

  Add a modification pattern to a sender modification sequence.
- l4_msgtag_t l4_thread_modify_sender_commit (l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW

  Apply (commit) a sender modification sequence.
- l4_msgtag_t l4_thread_arm_set_tpidruro (l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW

  Set the TPIDRURO thread specific register.

13.107.1 Detailed Description

C Thread object interface, see L4::Thread for the C++ interface.

An L4 thread is a thread of execution in the L4 context. Usually user-level and kernel threads are mapped 1:1 to each other. Thread kernel objects are created using a factory, see Factory (l4_factory_create_thread()).

Amongst other things an L4 thread encapsulates:

- CPU state
  - General-purpose registers
  - Program counter
  - Stack pointer
- FPU state
- Scheduling parameters, see the Scheduler API
- Execution state
  - Blocked, Runnable, Running

Thread objects provide an API for

- Thread configuration and manipulation
- Thread switching.

The thread control functions are used to control various aspects of a thread. See l4_thread_control_start() for more information.

Include File

```
#include <l4/sys/thread.h>
```

For the C++ interface refer to L4::Thread.

13.107.2 Enumeration Type Documentation

13.107.2.1 L4_thread_control_flags

```
enum L4_thread_control_flags
```

Flags for the thread control operation.

Enumerator

L4_THREAD_CONTROL_SET_PAGER	The pager will be given.
L4_THREAD_CONTROL_BIND_TASK	The task to bind the thread to will be given.
L4_THREAD_CONTROL_ALIEN	Alien state of the thread is set.
L4_THREAD_CONTROL_UX_NATIVE	Fiasco-UX only: pass-through of host system calls is set.
L4_THREAD_CONTROL_SET_EXC_HANDLER	The exception handler of the thread will be given.

Definition at line 701 of file thread.h.

13.107.2.2 L4_thread_control_mr_indices

```
enum L4_thread_control_mr_indices
```

Indices for the values in the message register for thread control.

Enumerator

L4_THREAD_CONTROL_MR_IDX_FLAGS	**See also** L4_thread_control_flags.
L4_THREAD_CONTROL_MR_IDX_PAGER	Index for pager cap.
L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER	Index for exception handler.
L4_THREAD_CONTROL_MR_IDX_FLAG_VALS	Index for feature values.
L4_THREAD_CONTROL_MR_IDX_BIND_UTCB	Index for UTCB address for bind.
L4_THREAD_CONTROL_MR_IDX_BIND_TASK	Index for task flex-page for bind.

Definition at line 724 of file thread.h.

13.107.2.3 L4_thread_ex_regs_flags

```
enum L4_thread_ex_regs_flags
```

Flags for the thread ex-regs operation.

Enumerator

L4_THREAD_EX_REGS_CANCEL	Cancel ongoing IPC in the thread.
L4_THREAD_EX_REGS_TRIGGER_EXCEPTION	Trigger artificial exception in thread.

Definition at line 739 of file thread.h.

13.107.3 Function Documentation

13.107.3.1 l4_thread_arm_set_tpidruro()

```
l4_msgtag_t l4_thread_arm_set_tpidruro (
            l4_cap_idx_t thread,
            l4_addr_t tpidruro )  [inline]
```

Set the TPIDRURO thread specific register.

Parameters

thread	Thread to manipulate
tpidruro	The value to be set

Returns

System call return tag

Definition at line 59 of file thread.h.

13.107.3.2 l4_thread_ex_regs()

```
l4_msgtag_t l4_thread_ex_regs (
            l4_cap_idx_t thread,
            l4_addr_t ip,
            l4_addr_t sp,
            l4_umword_t flags )  [inline]
```

Exchange basic thread registers.

Parameters

thread	Capability selector of the thread to manipulate.
ip	New instruction pointer, use ~0UL to leave the instruction pointer unchanged.
sp	New stack pointer, use ~0UL to leave the stack pointer unchanged.
flags	Ex-regs flags, see L4_thread_ex_regs_flags.

Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if L4_THREAD_EX_REGS_TRIGGER_EXCEPTION forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

The thread is started using l4_scheduler_run_thread(). However, if at the time l4_scheduler_run_thread() is called, the instruction pointer of the thread is invalid, a later call to l4_thread_ex_regs() with a valid instruction pointer might start the thread.

Examples

examples/sys/aliens/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 890 of file thread.h.

References l4_thread_ex_regs_u(), and l4_utcb().

Here is the call graph for this function:

13.107.3.3 l4_thread_ex_regs_ret()

```
l4_msgtag_t l4_thread_ex_regs_ret (
            l4_cap_idx_t thread,
            l4_addr_t * ip,
            l4_addr_t * sp,
            l4_umword_t * flags )  [inline]
```

Exchange basic thread registers and return previous values.

Parameters

	thread	Capability selector of the thread to manipulate.
in,out	*ip*	New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer.
in,out	*sp*	New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer.
in,out	*flags*	Ex-regs flags, see L4_thread_ex_regs_flags, return previous CPU flags of the thread.

Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if L4_THREAD_EX_REGS_TRIGGER_EXCEPTION forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

The thread is started using l4_scheduler_run_thread(). However, if at the time l4_scheduler_run_thread() is called, the instruction pointer of the thread is invalid, a later call to l4_thread_ex_regs() with a valid instruction pointer might start the thread.

Returned values are valid only if function returns successfully.

Definition at line 897 of file thread.h.

References l4_thread_ex_regs_ret_u(), and l4_utcb().

Here is the call graph for this function:

13.107.3.4 l4_thread_ex_regs_ret_u()

```
l4_msgtag_t l4_thread_ex_regs_ret_u (
            l4_cap_idx_t thread,
            l4_addr_t * ip,
            l4_addr_t * sp,
            l4_umword_t * flags,
            l4_utcb_t * utcb )  [inline]
```

Exchange basic thread registers and return previous values.

Parameters

	thread	Capability selector of the thread to manipulate.
in,out	*ip*	New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer.
in,out	*sp*	New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer.
in,out	*flags*	Ex-regs flags, see L4_thread_ex_regs_flags, return previous CPU flags of the thread.
	utcb	UTCB to use for this operation.

Returns

System call return tag. [out] parameters are only valid if the function returns successfully. Use l4_error() to check.

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if

L4_THREAD_EX_REGS_TRIGGER_EXCEPTION forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

The thread is started using L4::Scheduler::run_thread(). However, if at the time L4::Scheduler::run_thread() is called, the instruction pointer of the thread is invalid, a later call to ex_regs() with a valid instruction pointer might start the thread.

Definition at line 763 of file thread.h.

Referenced by L4::Thread::ex_regs(), and l4_thread_ex_regs_ret().

Here is the caller graph for this function:

13.107.3.5 l4_thread_ex_regs_u()

```
l4_msgtag_t l4_thread_ex_regs_u (
            l4_cap_idx_t thread,
            l4_addr_t ip,
            l4_addr_t sp,
            l4_umword_t flags,
            l4_utcb_t * utcb )  [inline]
```

Exchange basic thread registers.

Parameters

thread	Capability selector of the thread to manipulate.
ip	New instruction pointer, use ~0UL to leave the instruction pointer unchanged.
sp	New stack pointer, use ~0UL to leave the stack pointer unchanged.
flags	Ex-regs flags, see L4_thread_ex_regs_flags.
utcb	UTCB to use for this operation.

Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and

to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if L4_THREAD_EX_REGS_TRIGGER_EXCEPTION forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

The thread is started using L4::Scheduler::run_thread(). However, if at the time L4::Scheduler::run_thread() is called, the instruction pointer of the thread is invalid, a later call to ex_regs() with a valid instruction pointer might start the thread.

Definition at line 752 of file thread.h.

Referenced by L4::Thread::ex_regs(), and l4_thread_ex_regs().

Here is the caller graph for this function:

13.107.3.6 l4_thread_modify_sender_add()

```
int l4_thread_modify_sender_add (
            l4_umword_t match_mask,
            l4_umword_t match,
            l4_umword_t del_bits,
            l4_umword_t add_bits,
            l4_msgtag_t * tag )  [inline]
```

Add a modification pattern to a sender modification sequence.

Parameters

tag	Tag received from l4_thread_modify_sender_start() or previous l4_thread_modify_sender_add() calls from the same sequence.
match_mask	Bitmask of bits to match the label.
match	Bitmask that must be equal to the label after applying match_mask.
del_bits	Bits to be deleted from the label.
add_bits	Bits to be added to the label.

Returns

0 on sucess, <0 on error

In pseudo code: if ((sender_label & match_mask) == match) { sender_label = (sender_label & ~del_bits) | add_bits; }

Only the first match is applied.

See also

l4_thread_modify_sender_start

l4_thread_modify_sender_commit

Definition at line 1070 of file thread.h.

13.107.3.7 l4_thread_modify_sender_commit()

```
l4_msgtag_t l4_thread_modify_sender_commit (
            l4_cap_idx_t thread,
            l4_msgtag_t tag ) [inline]
```

Apply (commit) a sender modification sequence.

The modification rules are applied to all IPCs to the thread (whether directly or by IPC gate) that are already in flight, that is that the sender is already blocking on.

See also

l4_thread_modify_sender_start

l4_thread_modify_sender_add

Definition at line 1081 of file thread.h.

13.107.3.8 l4_thread_modify_sender_start()

```
l4_msgtag_t l4_thread_modify_sender_start (
            void ) [inline]
```

Start a thread sender modification sequence.

Add modification rules with l4_thread_modify_sender_add() and commit with l4_thread_modify_sender_commit(). Do not touch the UTCB between l4_thread_modify_sender_start() and l4_thread_modify_sender_commit().

This mechanism shall be used to change the source object labels of every pending IPC of an IPC gate or an IRQ if the labels in such pending IPC become invalid for the receiving thread, potentially because:

- a thread was unbound from an IPC gate / IRQ, or
- an IPC gate /IRQ was removed, or
- the label of an IPC gate /IRQ bound to a thread was changed.

It is not required to perform the modify_sender mechanism after an IPC gate or an IRQ was bound to a thread for the first time.

See also

l4_thread_modify_sender_add

l4_thread_modify_sender_commit

Definition at line 1064 of file thread.h.

13.107.3.9 l4_thread_register_del_irq()

```
l4_msgtag_t l4_thread_register_del_irq (
            l4_cap_idx_t thread,
            l4_cap_idx_t irq ) [inline]
```

Register an IRQ that will trigger upon deletion events.

Parameters

thread	Thread to register IRQ for.
irq	Capability selector for the IRQ object to be triggered.

Returns

System call return tag containing the return code.

Return values

L4_EPERM	L4_CAP_FPAGE_W missing on `irq`

In case the `irq` is already bound to an interrupt source, it is unbound first. When `irq` is deleted, it will be deregistered first. A registered deletion Irq can only be deregistered by deleting the Irq or the thread.

List of deletion events:

- deletion of one or several IPC gates bound to this thread.

When the deletion event is delivered, there is no indication about which IPC gate was deleted.

Definition at line 991 of file thread.h.

13.107.3.10 l4_thread_stats_time()

```
l4_msgtag_t l4_thread_stats_time (
            l4_cap_idx_t thread,
            l4_kernel_clock_t * us ) [inline]
```

Get consumed time of thread in µs.

Parameters

	thread	Thread to get the consumed time from.
`out`	*us*	Consumed time in µs.

Returns

system call return tag

Definition at line 959 of file thread.h.

13.107.3.11 l4_thread_switch()

```
l4_msgtag_t l4_thread_switch (
            l4_cap_idx_t to_thread ) [inline]
```

Switch to another thread (and donate the remaining time slice).

Parameters

to_thread	The thread to switch to.

Returns

system call return tag

Definition at line 950 of file thread.h.

13.107.3.12 l4_thread_vcpu_control()

```
l4_msgtag_t l4_thread_vcpu_control (
            l4_cap_idx_t thread,
            l4_addr_t vcpu_state ) [inline]
```

Enable the vCPU feature for the thread.

Parameters

thread	Capability selector of the thread for which the vCPU feature shall be enabled.
vcpu_state	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see l4_task_add_ku_mem()).

Returns

Syscall return tag.

This function enables the vCPU feature of the `thread`.

The kernel-user memory area starting at `vcpu_state` must be at least 128-byte aligned and must cover the size of l4_vcpu_state_t.

Note

Disabling of the vCPU feature is optional and currently not supported.

Definition at line 1008 of file thread.h.

References l4_thread_vcpu_control_u(), and l4_utcb().

Here is the call graph for this function:

13.107.3.13 l4_thread_vcpu_control_ext()

```
l4_msgtag_t l4_thread_vcpu_control_ext (
            l4_cap_idx_t thread,
            l4_addr_t ext_vcpu_state )  [inline]
```

Enable the extended vCPU feature for the thread.

Parameters

thread	Capability selector of the thread for which the extended vCPU feature shall be enabled.
ext_vcpu_state	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see l4_task_add_ku_mem()).

Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT or AMD's SVM.

This function enables the extended vCPU feature of the `thread`. Enabling the extended vCPU feature also enables the vCPU feature.

The kernel-user memory area starting at `ext_vcpu_state` must be at least 4 KiB aligned and must cover a size of L4_PAGESIZE. It includes the data of l4_vcpu_state_t at offset 0, the extended vCPU state at offset L4_VCPU↩_OFFSET_EXT_STATE, and, on some platforms, the extended vCPU information at offset L4_VCPU_OFFSET_↩EXT_INFOS.

Note

Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.

Disabling of the extended vCPU feature is currently not supported.

Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 1023 of file thread.h.

References l4_thread_vcpu_control_ext_u(), and l4_utcb().

Here is the call graph for this function:

13.107.3.14 l4_thread_vcpu_control_ext_u()

```
l4_msgtag_t l4_thread_vcpu_control_ext_u (
            l4_cap_idx_t thread,
            l4_addr_t ext_vcpu_state,
            l4_utcb_t * utcb )  [inline]
```

Enable the extended vCPU feature for the thread.

Parameters

thread	Capability selector of the thread for which the extended vCPU feature shall be enabled.
ext_vcpu_state	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()).
utcb	UTCB to use for this operation.

Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT or AMD's SVM.

This function enables the extended vCPU feature of `this` thread. Enabling the extended vCPU feature also enables the vCPU feature.

The kernel-user memory area starting at `ext_vcpu_state` must be at least 4 KiB aligned and must cover a size of L4_PAGESIZE. It includes the data of l4_vcpu_state_t at offset 0, the extended vCPU state at offset L4_VCPU↩_OFFSET_EXT_STATE, and, on some platforms, the extended vCPU information at offset L4_VCPU_OFFSET_↩EXT_INFOS.

Note

Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.

Disabling of the extended vCPU feature is currently not supported.

Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 1013 of file thread.h.

Referenced by l4_thread_vcpu_control_ext(), and L4::Thread::vcpu_control_ext().

Here is the caller graph for this function:

13.107.3.15 l4_thread_vcpu_control_u()

```
l4_msgtag_t l4_thread_vcpu_control_u (
            l4_cap_idx_t thread,
            l4_addr_t vcpu_state,
            l4_utcb_t * utcb )  [inline]
```

Enable the vCPU feature for the thread.

Parameters

thread	Capability selector of the thread for which the vCPU feature shall be enabled.
vcpu_state	A virtual address pointing to a l4_vcpu_state_t. It must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()).
utcb	UTCB to use for this operation.

Returns

Syscall return tag.

This function enables the vCPU feature of `this` thread

The kernel-user memory starting at `vcpu_state` must be at least 128-byte aligned and must cover the size of l4_vcpu_state_t.

The asynchronous IPC handling is described at vCPU API.

Note

Disabling of the vCPU feature is optional and currently not supported.

Definition at line 998 of file thread.h.

Referenced by l4_thread_vcpu_control(), and L4::Thread::vcpu_control().

Here is the caller graph for this function:

13.107.3.16 l4_thread_vcpu_resume_commit()

```
l4_msgtag_t l4_thread_vcpu_resume_commit (
            l4_cap_idx_t thread,
            l4_msgtag_t tag ) [inline]
```

Commit vCPU resume.

Parameters

thread	Thread to be resumed, the invalid cap can be used for the current thread.
tag	Tag to use, returned by l4_thread_vcpu_resume_start()

Returns

System call result message tag. In extended vCPU mode and when the virtual interrupts are cleared, the return code 1 flags an incoming IPC message, whereas 0 indicates a VM exit. An error is returned upon:

- Insufficient rights on the given task capability (-L4_EPERM).
- Given task capability is invalid (-L4_ENOENT).
- A supplied mapping failed.

To resume into another address space the capability to the target task must be set in the vCPU-state, with all lower bits in the task capability cleared (see L4_CAP_MASK). The kernel adds the L4_SYSF_SEND flag to this field

to indicate that the capability has been referenced in the kernel. Consecutive resumes will not reference the task capability again until all bits are cleared again. To release a task use the different task capability or use an invalid capability with the L4_SYSF_REPLY flag set.

See also

l4_vcpu_state_t

Definition at line 971 of file thread.h.

13.107.3.17 l4_thread_vcpu_resume_start()

```
l4_msgtag_t l4_thread_vcpu_resume_start (
            void ) [inline]
```

vCPU return from event handler.

Returns

Message tag to be used for l4_sndfpage_add() and l4_thread_vcpu_resume_commit()

The vCPU resume functionality is split in multiple functions to allow the specification of additional send-flex-pages using l4_sndfpage_add().

Definition at line 965 of file thread.h.

13.107.3.18 l4_thread_yield()

```
l4_msgtag_t l4_thread_yield (
            void ) [inline]
```

Yield current time slice.

Returns

system call return tag

Definition at line 839 of file thread.h.

References L4_INVALID_CAP, and l4_ipc_receive().

Here is the call graph for this function:

13.108 Thread Control Registers (TCRs)

Collaboration diagram for Thread Control Registers (TCRs):

Data Structures

- struct l4_thread_regs_t

  Encapsulation of the thread-control-register block of the UTCB.

Typedefs

- typedef struct l4_thread_regs_t l4_thread_regs_t

  Encapsulation of the thread-control-register block of the UTCB.

13.108.1 Detailed Description

13.109 Thread control

API for Thread Control method.

Collaboration diagram for Thread control:

Functions

- void l4_thread_control_start (void) L4_NOTHROW

  Start a thread control API sequence.
- void l4_thread_control_pager (l4_cap_idx_t pager) L4_NOTHROW

  Set the pager.
- void l4_thread_control_exc_handler (l4_cap_idx_t exc_handler) L4_NOTHROW

  Set the exception handler.
- void l4_thread_control_bind (l4_utcb_t *thread_utcb, l4_cap_idx_t task) L4_NOTHROW

  Bind the thread to a task.
- void l4_thread_control_alien (int on) L4_NOTHROW

  Enable alien mode.
- void l4_thread_control_ux_host_syscall (int on) L4_NOTHROW

  Enable pass through of native host (Linux) system calls.
- l4_msgtag_t l4_thread_control_commit (l4_cap_idx_t thread) L4_NOTHROW

  Commit the thread control parameters.

13.109.1 Detailed Description

API for Thread Control method.

The thread control API provides access to almost any parameter of a thread object. The API is based on a single invocation of the thread object. However, because of the huge amount of parameters, the API provides a set of functions to set specific parameters of a thread and a commit function to commit the thread control call (see l4_thread_control_commit()).

A thread control operation must always start with l4_thread_control_start() and be committed with l4_thread_control_commit(). All other thread control parameter setter functions must be called between these two functions.

An example for a sequence of thread control API calls can be found below.

```
l4_utcb_t *u = l4_utcb();
l4_thread_control_start(u);
l4_thread_control_pager(u, pager_cap);
l4_thread_control_bind (u, thread_utcb, task);
l4_thread_control_commit(u, thread_cap);
```

13.109.2 Function Documentation

13.109.2.1 l4_thread_control_alien()

```
void l4_thread_control_alien (
            int on )  [inline]
```

Enable alien mode.

Parameters

on	Boolean value defining the state of the feature.

For a thread in alien mode the kernel produces just an exception IPC for each IPC and exception caused by the alien thread instead of handling these events regularly. (Page faults of alien threads and interrupts occuring while the alien thread is running are always handled regularly.) While the alien thread is blocking, the execption handler can inspect and modify the state of the alien thread and potentially also the systemcall arguments. If the exception handler replies with L4_PROTO_ALLOW_SYSCALL as message tag, the kernel handles the next IPC or exception of the alien thread in a regular way. If the exception handler leaves certain thread state unchanged (in particular the instruction pointer), this will be the IPC or exception that caused the call of the exception handler. For a regularly processed IPC or exception of the alien thread the kernel also performs an exception IPC on kernel exit.

This feature can be used to attach a debugger to a thread and trace all object invocations and their results. It could also be used to handle other systems that use the same syscall instruction as L4Re.

Examples

examples/sys/aliens/main.c, and examples/sys/singlestep/main.c.

Definition at line 929 of file thread.h.

13.109.2.2 l4_thread_control_bind()

```
void l4_thread_control_bind (
            l4_utcb_t * thread_utcb,
            l4_cap_idx_t task )  [inline]
```

Bind the thread to a task.

Parameters

thread_utcb	The thread's UTCB address within the task it shall be bound to. The address must be aligned (architecture dependent; at least word aligned) and it must point to at least L4_UTCB_OFFSET bytes of kernel-user memory.
task	The task the thread shall be bound to.

Precondition

The thread must not be bound to a task yet.

A thread may execute code in the context of a task if and only if the thread is bound to the task. To actually start execution, l4_thread_ex_regs() needs to be used. Execution in the context of the task means that the code has access to all the task's resources (and only those). The executed code itself must be one of those resources. A thread can be bound at most once to a task.

Note

The UTCBs of different threads in the same task should not overlap in order to prevent data corruption.

Examples

examples/sys/aliens/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 923 of file thread.h.

13.109.2.3 l4_thread_control_commit()

```
l4_msgtag_t l4_thread_control_commit (
            l4_cap_idx_t thread ) [inline]
```

Commit the thread control parameters.

Parameters

thread	Capability selector of target thread to commit to.

Returns

system call return tag

Examples

examples/sys/aliens/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 941 of file thread.h.

13.109.2.4 l4_thread_control_exc_handler()

```
void l4_thread_control_exc_handler (
            l4_cap_idx_t exc_handler ) [inline]
```

Set the exception handler.

Parameters

exc_handler	Capability selector invoked to send an exception IPC.

Note

The exception-handler capability selector is interpreted in the task the thread is bound to (executes in).

Examples

examples/sys/aliens/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 916 of file thread.h.

13.109.2.5 l4_thread_control_pager()

```
void l4_thread_control_pager (
            l4_cap_idx_t pager )  [inline]
```

Set the pager.

Parameters

pager	Capability selector invoked to send a page-fault IPC.

Note

The pager capability selector is interpreted in the task the thread is bound to (executes in).

Examples

examples/sys/aliens/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 910 of file thread.h.

13.109.2.6 l4_thread_control_start()

```
void l4_thread_control_start (
            void  )  [inline]
```

Start a thread control API sequence.

This function starts a sequence of thread control API functions. After this functions any of following functions may be called in any order.

- l4_thread_control_pager()
- l4_thread_control_exc_handler()
- l4_thread_control_bind()
- l4_thread_control_alien()
- l4_thread_control_ux_host_syscall() (Fiasco-UX only)

To commit the changes to the thread l4_thread_control_commit() must be called in the end.

Note

The thread control API calls store the parameters for the thread in the UTCB of the caller, this means between l4_thread_control_start() and l4_thread_control_commit() no functions that modify the UTCB contents must be called.

Examples

examples/sys/aliens/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 904 of file thread.h.

13.109.2.7 l4_thread_control_ux_host_syscall()

```
void l4_thread_control_ux_host_syscall (
            int on )  [inline]
```

Enable pass through of native host (Linux) system calls.

Parameters

on	Boolean value defining the state of the feature.

Precondition

Running on Fiasco-UX

This enables the thread to do host system calls. This feature is only available in Fiasco-UX and ignored in other environments.

Definition at line 935 of file thread.h.

13.110 Timeouts

All kinds of timeouts and time related functions.

Collaboration diagram for Timeouts:

Data Structures

- struct l4_timeout_s

  Basic timeout specification.
- union l4_timeout_t

  Timeout pair.

Macros

- #define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})

  Timeout constants.
- #define L4_IPC_TIMEOUT_NEVER ((l4_timeout_s){0})

  never timeout
- #define L4_IPC_NEVER_INITIALIZER {0}

  never timeout, initializer
- #define L4_IPC_NEVER ((l4_timeout_t){0})

  never timeout
- #define L4_IPC_RECV_TIMEOUT_0 ((l4_timeout_t){0x00000400})

  0 receive timeout
- #define L4_IPC_SEND_TIMEOUT_0 ((l4_timeout_t){0x04000000})

  0 send timeout
- #define L4_IPC_BOTH_TIMEOUT_0 ((l4_timeout_t){0x04000400})

  0 receive and send timeout

Typedefs

- typedef struct l4_timeout_s l4_timeout_s

  Basic timeout specification.
- typedef union l4_timeout_t l4_timeout_t

  Timeout pair.

Functions

- l4_timeout_s l4_timeout_rel (unsigned man, unsigned exp) L4_NOTHROW

  Get relative timeout consisting of mantissa and exponent.
- l4_timeout_t l4_ipc_timeout (unsigned snd_man, unsigned snd_exp, unsigned rcv_man, unsigned rcv_exp) L4_NOTHROW

  Convert explicit timeout values to l4_timeout_t type.
- l4_timeout_t l4_timeout (l4_timeout_s snd, l4_timeout_s rcv) L4_NOTHROW

  Combine send and receive timeout in a timeout.
- void l4_snd_timeout (l4_timeout_s snd, l4_timeout_t *to) L4_NOTHROW

  Set send timeout in given to timeout.
- void l4_rcv_timeout (l4_timeout_s rcv, l4_timeout_t *to) L4_NOTHROW

  Set receive timeout in given to timeout.
- l4_kernel_clock_t l4_timeout_rel_get (l4_timeout_s to) L4_NOTHROW

  Get clock value of out timeout.
- unsigned l4_timeout_is_absolute (l4_timeout_s to) L4_NOTHROW

  Return whether the given timeout is absolute or not.
- l4_kernel_clock_t l4_timeout_get (l4_kernel_clock_t cur, l4_timeout_s to) L4_NOTHROW

  Get clock value for a clock + a timeout.
- l4_timeout_s l4_timeout_abs (l4_kernel_clock_t pint, int br) L4_NOTHROW

  Set an absolute timeout.
- unsigned l4_utcb_mr64_idx (unsigned idx) L4_NOTHROW

  Get index into 64bit message registers alias from native-sized index.

13.110.1 Detailed Description

All kinds of timeouts and time related functions.

13.110.2 Macro Definition Documentation

13.110.2.1 L4_IPC_TIMEOUT_0

```
#define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})
```

Timeout constants.

0 timeout

Definition at line 79 of file __timeout.h.

13.110.3 Typedef Documentation

13.110.3.1 l4_timeout_s

```
typedef struct l4_timeout_s l4_timeout_s
```

Basic timeout specification.

Basically a floating point number with 10 bits mantissa and 5 bits exponent (t = m∗2^e).

If bit 15 == 1 the timeout is absolute and the lower 6 bits encode the index of the UTCB buffer register(s) holding the absolute 64-bit timeout value. On 32-bit systems, two consecutive UTCB buffer registers are used.

13.110.3.2 l4_timeout_t

```
typedef union l4_timeout_t l4_timeout_t
```

Timeout pair.

For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

13.110.4 Function Documentation

13.110.4.1 l4_ipc_timeout()

```
l4_timeout_t l4_ipc_timeout (
            unsigned snd_man,
            unsigned snd_exp,
            unsigned rcv_man,
            unsigned rcv_exp )  [inline]
```

Convert explicit timeout values to l4_timeout_t type.

Parameters

snd_man	Mantissa of send timeout.
snd_exp	Exponent of send timeout.
rcv_man	Mantissa of receive timeout.
rcv_exp	Exponent of receive timeout.

Definition at line 187 of file __timeout.h.

References l4_timeout_t::p, l4_timeout_t::rcv, l4_timeout_t::snd, and l4_timeout_s::t.

13.110.4.2 l4_rcv_timeout()

```
void l4_rcv_timeout (
            l4_timeout_s rcv,
            l4_timeout_t * to )  [inline]
```

Set receive timeout in given to timeout.

Parameters

	rcv	Receive timeout
out	*to*	L4 timeout

Definition at line 215 of file __timeout.h.

13.110.4.3 l4_snd_timeout()

```
void l4_snd_timeout (
            l4_timeout_s snd,
            l4_timeout_t * to )  [inline]
```

Set send timeout in given to timeout.

Parameters

	snd	Send timeout
out	*to*	L4 timeout

Definition at line 208 of file __timeout.h.

13.110.4.4 l4_timeout()

```
l4_timeout_t l4_timeout (
```

```
l4_timeout_s snd,
l4_timeout_s rcv )  [inline]
```

Combine send and receive timeout in a timeout.

Parameters

snd	Send timeout
rcv	Receive timeout

Returns

L4 timeout

Definition at line 198 of file __timeout.h.

References l4_timeout_t::p, l4_timeout_t::rcv, and l4_timeout_t::snd.

Referenced by L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::timeout().

Here is the caller graph for this function:

13.110.4.5 l4_timeout_abs()

```
l4_timeout_s l4_timeout_abs (
            l4_kernel_clock_t pint,
            int br )  [inline]
```

Set an absolute timeout.

Parameters

pint	Point in time in clocks
br	The buffer register the timeout shall be placed in. (

Note

On 32bit architectures the timeout needs two consecutive buffers.)

The absolute timeout value will be placed into the buffer register *br* of the current thread.

Returns

timeout value

Definition at line 383 of file utcb.h.

Referenced by L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::timeout().

Here is the caller graph for this function:

13.110.4.6 l4_timeout_get()

```
l4_kernel_clock_t l4_timeout_get (
            l4_kernel_clock_t cur,
            l4_timeout_s to ) [inline]
```

Get clock value for a clock + a timeout.

Parameters

cur	Clock value
to	L4 timeout

Returns

Clock sum

Definition at line 245 of file __timeout.h.

References l4_timeout_is_absolute(), and l4_timeout_rel_get().

Here is the call graph for this function:

13.110.4.7 l4_timeout_is_absolute()

```
unsigned l4_timeout_is_absolute (
            l4_timeout_s to ) [inline]
```

Return whether the given timeout is absolute or not.

Parameters

to	L4 timeout

Returns

!= 0 if absolute, 0 if relative

Definition at line 238 of file __timeout.h.

Referenced by l4_timeout_get().

Here is the caller graph for this function:

13.110.4.8 l4_timeout_rel()

```
l4_timeout_s l4_timeout_rel (
            unsigned man,
            unsigned exp )  [inline]
```

Get relative timeout consisting of mantissa and exponent.

Parameters

man	Mantissa of timeout
exp	Exponent of timeout

Returns

timeout value

Definition at line 222 of file __timeout.h.

13.110.4.9 l4_timeout_rel_get()

```
l4_kernel_clock_t l4_timeout_rel_get (
            l4_timeout_s to )  [inline]
```

Get clock value of out timeout.

Parameters

to	L4 timeout

Returns

Clock value

Definition at line 229 of file __timeout.h.

Referenced by l4_timeout_get().

Here is the caller graph for this function:

13.110.4.10 l4_utcb_mr64_idx()

```
unsigned l4_utcb_mr64_idx (
            unsigned idx )  [inline]
```

Get index into 64bit message registers alias from native-sized index.

Parameters

idx	Index to native-sized message register

Returns

Index to 64bit message register alias

Definition at line 386 of file utcb.h.

13.111 Timestamp Counter

Collaboration diagram for Timestamp Counter:

Files

- file rdtsc.h

  time stamp counter related functions
- file rdtsc.h

  time stamp counter related functions

Functions

- l4_cpu_time_t l4_rdtsc (void)

  Read current value of CPU-internal time stamp counter.
- l4_uint32_t l4_rdtsc_32 (void)

  Read the lest significant 32 bit of the TSC.
- l4_uint64_t l4_rdpmc (int ecx)

  Return current value of CPU-internal performance measurement counter.
- l4_uint32_t l4_rdpmc_32 (int ecx)

  Return the least significant 32 bit of a performance counter.

- l4_uint64_t l4_tsc_to_ns (l4_cpu_time_t tsc)

  Convert time stamp to ns value.
- l4_uint64_t l4_tsc_to_us (l4_cpu_time_t tsc)

  Convert time stamp into micro seconds value.
- void l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)

  Convert time stamp to s.ns value.
- l4_cpu_time_t l4_ns_to_tsc (l4_uint64_t ns)

  Convert nano seconds into CPU ticks.
- void l4_busy_wait_ns (l4_uint64_t ns)

  Wait busy for a small amount of time.
- void l4_busy_wait_us (l4_uint64_t us)

  Wait busy for a small amount of time.
- l4_uint32_t l4_calibrate_tsc (l4_kernel_info_t *kip)

  Determine scalers for time stamp calculations.
- l4_uint32_t l4_tsc_init (l4_kernel_info_t *kip)

  Initialize scaler for TSC calibrations from the kernel.
- l4_uint32_t l4_get_hz (void)

  Get CPU frequency in Hz.

13.111.1 Detailed Description

13.111.2 Function Documentation

13.111.2.1 l4_busy_wait_ns()

```
void l4_busy_wait_ns (
            l4_uint64_t ns )  [inline]
```

Wait busy for a small amount of time.

Parameters

ns	nano seconds to wait

Attention

Not intended for any use!

Definition at line 264 of file rdtsc.h.

References l4_ns_to_tsc(), and l4_rdtsc().

Here is the call graph for this function:

13.111.2.2 l4_busy_wait_us()

```
void l4_busy_wait_us (
            l4_uint64_t us ) [inline]
```

Wait busy for a small amount of time.

Parameters

us	micro seconds to wait

Attention

Not intended for any use!

Definition at line 274 of file rdtsc.h.

References l4_ns_to_tsc(), and l4_rdtsc().

Here is the call graph for this function:

13.111.2.3 l4_calibrate_tsc()

```
l4_uint32_t l4_calibrate_tsc (
            l4_kernel_info_t * kip )  [inline]
```

Determine scalers for time stamp calculations.

Determine some scalers to be able to convert between real time and CPU ticks. Just calls l4_tsc_init().

Examples

examples/sys/aliens/main.c.

Definition at line 161 of file rdtsc.h.

References l4_tsc_init().

Here is the call graph for this function:

13.111.2.4 l4_get_hz()

```
l4_uint32_t l4_get_hz (
            void  )
```

Get CPU frequency in Hz.

Returns

frequency in Hz

13.111.2.5 l4_ns_to_tsc()

```
l4_cpu_time_t l4_ns_to_tsc (
            l4_uint64_t ns )  [inline]
```

Convert nano seconds into CPU ticks.

Parameters

ns	nano seconds

Returns

CPU ticks

Definition at line 250 of file rdtsc.h.

Referenced by l4_busy_wait_ns(), and l4_busy_wait_us().

Here is the caller graph for this function:

13.111.2.6 l4_rdpmc()

```
l4_uint64_t l4_rdpmc (
            int ecx ) [inline]
```

Return current value of CPU-internal performance measurement counter.

Parameters

ecx	ECX value for the rdpmc instruction. For details see the Intel IA-32 Architectures Software Developer's Manual.

Returns

64-bit PMC

Definition at line 177 of file rdtsc.h.

13.111.2.7 l4_rdpmc_32()

```
l4_uint32_t l4_rdpmc_32 (
            int ecx )  [inline]
```

Return the least significant 32 bit of a performance counter.

Useful for smaller differences, needs less cycles.

Definition at line 197 of file rdtsc.h.

13.111.2.8 l4_rdtsc()

```
l4_cpu_time_t l4_rdtsc (
            void  )  [inline]
```

Read current value of CPU-internal time stamp counter.

Returns

64-bit time stamp

Examples

examples/sys/aliens/main.c.

Definition at line 167 of file rdtsc.h.

Referenced by l4_busy_wait_ns(), and l4_busy_wait_us().

Here is the caller graph for this function:

13.111.2.9 l4_rdtsc_32()

```
l4_uint32_t l4_rdtsc_32 (
            void ) [inline]
```

Read the lest significant 32 bit of the TSC.

Useful for smaller differences, needs less cycles.

Definition at line 187 of file rdtsc.h.

13.111.2.10 l4_tsc_init()

```
l4_uint32_t l4_tsc_init (
            l4_kernel_info_t * kip )
```

Initialize scaler for TSC calibrations from the kernel.

Initialize the scalers needed by l4_tsc_to_ns()/l4_ns_to_tsc() and so on. Use the kernel-provided frequency.

Parameters

kip	KIP pointer

Returns

0 on error (no scalers exported by kernel) otherwise returns (2^32 / (tsc per µsec)). This value has the same semantics as the value returned by the calibrate_delay_loop() function of the Linux kernel.

Referenced by l4_calibrate_tsc().

Here is the caller graph for this function:

13.111.2.11 l4_tsc_to_ns()

```
l4_uint64_t l4_tsc_to_ns (
            l4_cpu_time_t tsc ) [inline]
```

Convert time stamp to ns value.

Parameters

tsc	time value in CPU ticks

Returns

time value in ns

Examples

examples/sys/aliens/main.c.

Definition at line 207 of file rdtsc.h.

13.111.2.12 l4_tsc_to_s_and_ns()

```
void l4_tsc_to_s_and_ns (
            l4_cpu_time_t tsc,
            l4_uint32_t * s,
            l4_uint32_t * ns )  [inline]
```

Convert time stamp to s.ns value.

Parameters

tsc	time value in CPU ticks

Return values

s	seconds
ns	nano seconds

Definition at line 235 of file rdtsc.h.

13.111.2.13 l4_tsc_to_us()

```
l4_uint64_t l4_tsc_to_us (
            l4_cpu_time_t tsc )  [inline]
```

Convert time stamp into micro seconds value.

Parameters

tsc	time value in CPU ticks

Returns

time value in micro seconds

Definition at line 221 of file rdtsc.h.

13.112 Utility Functions

Utilities, generic file.

Collaboration diagram for Utility Functions:

Modules

- Atomic Instructions
- Bit Manipulation
- Bitmap graphics and fonts

  This library provides some functions for bitmap handling in frame buffers.
- CPU related functions
- Comfortable Command Line Parsing
- ELF binary format

  Functions and types related to ELF binaries.
- Functions to manipulate the local IDT
- IA32 Port I/O API
- Internal functions
- Kernel Interface Page API
- Low-Level Thread Functions
- Random number support
- Timestamp Counter

Files

- file rand.h

  Simple Pseudo-Random Number Generator.

Functions

- void l4_sleep_forever (void) L4_NOTHROW)

  Go sleep and never wake up.
- long l4util_splitlog2_hdl (l4_addr_t start, l4_addr_t end, long(∗handler)(l4_addr_t s, l4_addr_t e, int log2size))

  Split a range into log2 base and size aligned chunks.
- l4_addr_t l4util_splitlog2_size (l4_addr_t start, l4_addr_t end)

  Return log2 base and size aligned length of a range.
- l4_timeout_s l4util_micros2l4to (unsigned int mus) L4_NOTHROW

  Calculate l4 timeouts.
- void l4_sleep (int ms) L4_NOTHROW

  Suspend thread for a period of ms milliseconds.
- void l4_usleep (int us) L4_NOTHROW

  Suspend thread for a period of us microseconds.
- void l4_touch_ro (const void ∗addr, unsigned size) L4_NOTHROW

  Touch data area to force mapping (read-only)
- void l4_touch_rw (const void ∗addr, unsigned size) L4_NOTHROW

  Touch data areas to force mapping (read-write)

13.112.1 Detailed Description

Utilities, generic file.

13.112.2 Function Documentation

13.112.2.1 l4_sleep()

```
void l4_sleep (
            int ms )
```

Suspend thread for a period of *ms* milliseconds.

Parameters

ms	Time in milliseconds

13.112.2.2 l4_touch_ro()

```
void l4_touch_ro (
            const void * addr,
            unsigned size )  [inline]
```

Touch data area to force mapping (read-only)

Parameters

addr	Start of memory area to touch.
size	Size of area to touch.

Examples

examples/sys/singlestep/main.c.

Definition at line 94 of file util.h.

References L4_PAGESIZE, and l4_trunc_page().

Here is the call graph for this function:

13.112.2.3 l4_touch_rw()

```
void l4_touch_rw (
            const void * addr,
            unsigned size )  [inline]
```

Touch data areas to force mapping (read-write)

Parameters

addr	Start of memory area to touch.
size	Size of area to touch.

Examples

examples/sys/aliens/main.c, and examples/sys/singlestep/main.c.

Definition at line 107 of file util.h.

References L4_PAGESIZE, and l4_trunc_page().

Here is the call graph for this function:

13.112.2.4 l4_usleep()

```
void l4_usleep (
            int us )
```

Suspend thread for a period of *us* microseconds.

Parameters

us	Time in microseconds

WARNING: This function is mostly bogus since the timer resolution of current L4 implementations is about 1ms!

References L4_IPC_NEVER, and l4_ipc_sleep().

Here is the call graph for this function:

13.112.2.5 l4util_micros2l4to()

```
l4_timeout_s l4util_micros2l4to (
            unsigned int mus )
```

Calculate l4 timeouts.

Parameters

mus	time in microseconds. Special cases: • 0 - > timeout 0 • ~0U -> timeout NEVER

Returns

the corresponding l4_timeout value

13.112.2.6 l4util_splitlog2_hdl()

```
long l4util_splitlog2_hdl (
            l4_addr_t start,
            l4_addr_t end,
            long(*)(l4_addr_t s, l4_addr_t e, int log2size) handler )  [inline]
```

Split a range into log2 base and size aligned chunks.

Parameters

start	Start of range
end	End of range (inclusive) (e.g. 2-4 is len 3)
handler	Handler function that is called with start and end (both inclusive) of the chunk. On success, the handler must return 0, if it returns !=0 the function will immediately return with the return code of the handler.

Returns

0 on success, != 0 otherwise

Definition at line 53 of file splitlog2.h.

References L4_EINVAL, and l4util_splitlog2_size().

Here is the call graph for this function:

13.112.2.7 l4util_splitlog2_size()

```
l4_addr_t l4util_splitlog2_size (
            l4_addr_t start,
            l4_addr_t end )  [inline]
```

Return log2 base and size aligned length of a range.

Parameters

start	Start of range
end	End of range (inclusive) (e.g. 2-4 is len 3)

Returns

length of elements in log2size (length is 1 << log2size)

Definition at line 72 of file splitlog2.h.

References l4util_bsf(), and l4util_bsr().

Referenced by l4util_splitlog2_hdl().

Here is the call graph for this function:

Here is the caller graph for this function:

13.113 VM API for SVM

Virtual machine API for SVM.

Collaboration diagram for VM API for SVM:

Data Structures

- struct l4_vm_svm_vmcb_control_area

  VMCB structure for SVM VMs.
- struct l4_vm_svm_vmcb_state_save_area_seg

  State save area segment selector struct.
- struct l4_vm_svm_vmcb_state_save_area

  State save area structure for SVM VMs.
- struct l4_vm_svm_vmcb_t

  Control structure for SVM VMs.

Typedefs

- typedef struct l4_vm_svm_vmcb_control_area l4_vm_svm_vmcb_control_area_t

  VMCB structure for SVM VMs.
- typedef struct l4_vm_svm_vmcb_state_save_area_seg l4_vm_svm_vmcb_state_save_area_seg_t

  State save area segment selector struct.
- typedef struct l4_vm_svm_vmcb_state_save_area l4_vm_svm_vmcb_state_save_area_t

  State save area structure for SVM VMs.
- typedef struct l4_vm_svm_vmcb_t l4_vm_svm_vmcb_t

  Control structure for SVM VMs.

13.113.1 Detailed Description

Virtual machine API for SVM.

13.114 VM API for TZ

Virtual Machine API for ARM TrustZone.

Collaboration diagram for VM API for TZ:

Data Structures

- struct l4_vm_tz_state

  state structure for TrustZone VMs

13.114.1 Detailed Description

Virtual Machine API for ARM TrustZone.

13.115 VM API for VMX

Virtual machine API for VMX.

Collaboration diagram for VM API for VMX:

Enumerations

- enum L4_vm_vmx_caps_regs {
  L4_VM_VMX_BASIC_REG = 0 , L4_VM_VMX_TRUE_PINBASED_CTLS_REG = 1 , L4_VM_VMX_TRUE_PROCBASED_CTL
  = 2 , L4_VM_VMX_TRUE_EXIT_CTLS_REG = 3 ,
  L4_VM_VMX_TRUE_ENTRY_CTLS_REG = 4 , L4_VM_VMX_MISC_REG = 5 , L4_VM_VMX_CR0_FIXED0_REG
  = 6 , L4_VM_VMX_CR0_FIXED1_REG = 7 ,
  L4_VM_VMX_CR4_FIXED0_REG = 8 , L4_VM_VMX_CR4_FIXED1_REG = 9 , L4_VM_VMX_VMCS_ENUM_REG
  = 0xa , L4_VM_VMX_PROCBASED_CTLS2_REG = 0xb ,
  L4_VM_VMX_EPT_VPID_CAP_REG = 0xc , L4_VM_VMX_NUM_CAPS_REGS }

  Exported VMX capability registers.
- enum L4_vm_vmx_dfl1_regs {
  L4_VM_VMX_PINBASED_CTLS_DFL1_REG = 0x1 , L4_VM_VMX_PROCBASED_CTLS_DFL1_REG =
  0x2 , L4_VM_VMX_EXIT_CTLS_DFL1_REG = 0x3 , L4_VM_VMX_ENTRY_CTLS_DFL1_REG = 0x4 ,
  L4_VM_VMX_NUM_DFL1_REGS }

  Exported VMX capability registers (default to 1 bits).
- enum {
  L4_VM_VMX_VMCS_CR2 = 0x683e , L4_VM_VMX_VMCS_XCR0 = 0x2840 , L4_VM_VMX_VMCS_MSR_SYSCALL_MASK
  = 0x2842 , L4_VM_VMX_VMCS_MSR_LSTAR = 0x2844 ,
  L4_VM_VMX_VMCS_MSR_CSTAR = 0x2846 , L4_VM_VMX_VMCS_MSR_TSC_AUX = 0x2848 ,
  L4_VM_VMX_VMCS_MSR_STAR = 0x284a , L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE = 0x284c }

  Additional (virtual) VMCS fields.

Functions

- l4_uint64_t l4_vm_vmx_get_caps (void const ∗vcpu_state, unsigned cap_msr) L4_NOTHROW

  Get a capability register for VMX.
- l4_uint32_t l4_vm_vmx_get_caps_default1 (void const ∗vcpu_state, unsigned cap_msr) L4_NOTHROW

  Get a default to one capability register for VMX.
- unsigned l4_vm_vmx_field_len (unsigned field) L4_NOTHROW

  Return length in bytes of a VMCS field.
- unsigned l4_vm_vmx_field_order (unsigned field) L4_NOTHROW

  Return length in power of two (bytes) of a VMCS field.
- void l4_vm_vmx_clear (void ∗vmcs, void ∗user_vmcs) L4_NOTHROW

  Saves cached state from the kernel VMCS to the user VMCS.
- void l4_vm_vmx_ptr_load (void ∗vmcs, void ∗user_vmcs) L4_NOTHROW

  Loads the user_vmcs as the current VMCS.
- l4_uint32_t l4_vm_vmx_get_cr2_index (void const ∗vmcs) L4_NOTHROW

  Get the VMCS field index of the virtual CR2 register.
- l4_umword_t l4_vm_vmx_read_nat (void ∗vmcs, unsigned field) L4_NOTHROW

  Read a natural width VMCS field.
- l4_uint16_t l4_vm_vmx_read_16 (void ∗vmcs, unsigned field) L4_NOTHROW

  Read a 16bit VMCS field.
- l4_uint32_t l4_vm_vmx_read_32 (void ∗vmcs, unsigned field) L4_NOTHROW

  Read a 32bit VMCS field.
- l4_uint64_t l4_vm_vmx_read_64 (void ∗vmcs, unsigned field) L4_NOTHROW

  Read a 64bit VMCS field.
- l4_uint64_t l4_vm_vmx_read (void ∗vmcs, unsigned field) L4_NOTHROW

  Read any VMCS field.
- void l4_vm_vmx_write_nat (void ∗vmcs, unsigned field, l4_umword_t val) L4_NOTHROW

  Write to a natural width VMCS field.
- void l4_vm_vmx_write_16 (void ∗vmcs, unsigned field, l4_uint16_t val) L4_NOTHROW

Write to a 16bit VMCS field.

- void l4_vm_vmx_write_32 (void *vmcs, unsigned field, l4_uint32_t val) L4_NOTHROW

  Write to a 32bit VMCS field.

- void l4_vm_vmx_write_64 (void *vmcs, unsigned field, l4_uint64_t val) L4_NOTHROW

  Write to a 64bit VMCS field.

- void l4_vm_vmx_write (void *vmcs, unsigned field, l4_uint64_t val) L4_NOTHROW

  Write to an arbitrary VMCS field.

13.115.1 Detailed Description

Virtual machine API for VMX.

13.115.2 Enumeration Type Documentation

13.115.2.1 anonymous enum

```
anonymous enum
```

Additional (virtual) VMCS fields.

The VMCS offsets defined here are actually not in the hardware VMCS. However our VMMs run in user mode and need to have access to certain registers available in kernel mode only. So we put them into our version of the VMCS.

Enumerator

L4_VM_VMX_VMCS_CR2	VMCS offset for CR2. **Note** You usually need to check this value against the value you get from l4_vm_vmx_get_cr2_index() to make sure you are running on a compatible kernel.
L4_VM_VMX_VMCS_XCR0	VMCS offset of extended control register XCR0.
L4_VM_VMX_VMCS_MSR_SYSCALL_MASK	VMCS offset of system call flag mask MSR.
L4_VM_VMX_VMCS_MSR_LSTAR	VMCS offset of IA32e mode system call target address MSR.
L4_VM_VMX_VMCS_MSR_CSTAR	VMCS offset of IA32 mode system call target address MSR.
L4_VM_VMX_VMCS_MSR_TSC_AUX	VMCS offset of auxiliary TSC signature MSR.
L4_VM_VMX_VMCS_MSR_STAR	VMCS offset of system call target address MSR.
L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE	VMCS offset of GS base address swap target MSR.

Definition at line 105 of file __vm-vmx.h.

13.115.2.2 L4_vm_vmx_caps_regs

```
enum L4_vm_vmx_caps_regs
```

Exported VMX capability registers.

Enumerator

L4_VM_VMX_BASIC_REG	Basic VMX capabilities.
L4_VM_VMX_TRUE_PINBASED_CTLS_REG	True pin-based control caps.
L4_VM_VMX_TRUE_PROCBASED_CTLS_REG	True processor based control caps.
L4_VM_VMX_TRUE_EXIT_CTLS_REG	True exit control caps.
L4_VM_VMX_TRUE_ENTRY_CTLS_REG	True entry control caps.
L4_VM_VMX_MISC_REG	Misc caps.
L4_VM_VMX_CR0_FIXED0_REG	Fixed to 0 bits of CR0.
L4_VM_VMX_CR0_FIXED1_REG	Fixed to 1 bits of CR0.
L4_VM_VMX_CR4_FIXED0_REG	Fixed to 0 bits of CR4.
L4_VM_VMX_CR4_FIXED1_REG	Fixed to 1 bits of CR4.
L4_VM_VMX_VMCS_ENUM_REG	VMCS enumeration info.
L4_VM_VMX_PROCBASED_CTLS2_REG	Processor based control 2 caps.
L4_VM_VMX_EPT_VPID_CAP_REG	EPT and VPID caps.
L4_VM_VMX_NUM_CAPS_REGS	Total number of VMX capability registers.

Definition at line 39 of file __vm-vmx.h.

13.115.2.3 L4_vm_vmx_dfl1_regs

```
enum L4_vm_vmx_dfl1_regs
```

Exported VMX capability registers (default to 1 bits).

Enumerator

L4_VM_VMX_PINBASED_CTLS_DFL1_REG	Default 1 bits in pin-based controls.
L4_VM_VMX_PROCBASED_CTLS_DFL1_REG	Default 1 bits in processor-based controls.
L4_VM_VMX_EXIT_CTLS_DFL1_REG	Default 1 bits in exit controls.
L4_VM_VMX_ENTRY_CTLS_DFL1_REG	Default 1 bits in entry controls.
L4_VM_VMX_NUM_DFL1_REGS	Total number of default on registers.

Definition at line 62 of file __vm-vmx.h.

13.115.3 Function Documentation

13.115.3.1 l4_vm_vmx_clear()

```
void l4_vm_vmx_clear (
            void * vmcs,
            void * user_vmcs )  [inline]
```

Saves cached state from the kernel VMCS to the user VMCS.

Parameters

vmcs	Pointer to the kernel VMCS.
user_vmcs	Pointer to the user VMCS.

This function is comparable to VMX vmclear.

Definition at line 433 of file __vm-vmx.h.

Referenced by l4_vm_vmx_ptr_load().

Here is the caller graph for this function:

13.115.3.2 l4_vm_vmx_field_len()

```
unsigned l4_vm_vmx_field_len (
            unsigned field )  [inline]
```

Return length in bytes of a VMCS field.

Parameters

field	Field number.

Returns

Width of field in bytes.

Definition at line 357 of file __vm-vmx.h.

References l4_vm_vmx_field_order().

Here is the call graph for this function:

13.115.3.3 l4_vm_vmx_field_order()

```
unsigned l4_vm_vmx_field_order (
            unsigned field ) [inline]
```

Return length in power of two (bytes) of a VMCS field.

Parameters

field	Field number.

Returns

Width of field in power of two (bytes).

Definition at line 342 of file __vm-vmx.h.

Referenced by l4_vm_vmx_field_len().

Here is the caller graph for this function:

13.115.3.4 l4_vm_vmx_get_caps()

```
l4_uint64_t l4_vm_vmx_get_caps (
            void const * vcpu_state,
            unsigned cap_msr ) [inline]
```

Get a capability register for VMX.

Parameters

vcpu_state	Pointer to the VCPU state of the VCPU.
cap_msr	Caps register index (see L4_vm_vmx_caps_regs).

Returns

The value of the capability register.

Definition at line 529 of file __vm-vmx.h.

References L4_VCPU_OFFSET_EXT_INFOS.

13.115.3.5 l4_vm_vmx_get_caps_default1()

```
l4_uint32_t l4_vm_vmx_get_caps_default1 (
            void const * vcpu_state,
            unsigned cap_msr )  [inline]
```

Get a default to one capability register for VMX.

Parameters

vcpu_state	Pointer to the VCPU state of the VCPU.
cap_msr	Default 1 caps register index (see L4_vm_vmx_dfl1_regs).

Returns

The value of the capability register.

Definition at line 537 of file __vm-vmx.h.

References L4_VCPU_OFFSET_EXT_INFOS, L4_VM_VMX_NUM_CAPS_REGS, and L4_VM_VMX_PINBASED_CTLS_DFL1_REG

13.115.3.6 l4_vm_vmx_get_cr2_index()

```
l4_uint32_t l4_vm_vmx_get_cr2_index (
            void const * vmcs )  [inline]
```

Get the VMCS field index of the virtual CR2 register.

Parameters

vmcs	Pointer to the software VMCS.

Returns

The field index used for the virtual CR2 register as used by the current Fiasco.OC interface.

The CR2 register is actually not in the hardware VMCS, however our VMMs run in user mode and need to have access to this register so we put it into our software version of the VMCS.

See also

L4_VM_VMX_VMCS_CR2

Definition at line 545 of file __vm-vmx.h.

13.115.3.7 l4_vm_vmx_ptr_load()

```
void l4_vm_vmx_ptr_load (
            void * vmcs,
            void * user_vmcs )  [inline]
```

Loads the user_vmcs as the current VMCS.

Parameters

vmcs	Pointer to the kernel VMCS.
user_vmcs	Pointer to the user VMCS.

This function is comparable to VMX vmptrld.

Definition at line 445 of file __vm-vmx.h.

References l4_vm_vmx_clear().

Here is the call graph for this function:

13.115.3.8 l4_vm_vmx_read()

```
l4_uint64_t l4_vm_vmx_read (
            void * vmcs,
            unsigned field )  [inline]
```

Read any VMCS field.

Parameters

vmcs	Pointer to the software VMCS.
field	The VMCS field index as used on VMX hardware.

Returns

The value of the VMCS field with the given index.

Definition at line 481 of file __vm-vmx.h.

References l4_vm_vmx_read_16(), l4_vm_vmx_read_32(), l4_vm_vmx_read_64(), and l4_vm_vmx_read_nat().

Here is the call graph for this function:

13.115.3.9 l4_vm_vmx_read_16()

```
l4_uint16_t l4_vm_vmx_read_16 (
            void * vmcs,
            unsigned field )  [inline]
```

Read a 16bit VMCS field.

Parameters

vmcs	Pointer to the software VMCS.
field	The VMCS field index as used on VMX hardware.

Returns

The value of the VMCS field with the given index.

Definition at line 466 of file __vm-vmx.h.

Referenced by l4_vm_vmx_read().

Here is the caller graph for this function:

13.115.3.10 l4_vm_vmx_read_32()

```
l4_uint32_t l4_vm_vmx_read_32 (
            void * vmcs,
            unsigned field )  [inline]
```

Read a 32bit VMCS field.

Parameters

vmcs	Pointer to the software VMCS.
field	The VMCS field index as used on VMX hardware.

Returns

The value of the VMCS field with the given index.

Definition at line 471 of file __vm-vmx.h.

Referenced by l4_vm_vmx_read().

Here is the caller graph for this function:

13.115.3.11 l4_vm_vmx_read_64()

```
l4_uint64_t l4_vm_vmx_read_64 (
            void * vmcs,
            unsigned field ) [inline]
```

Read a 64bit VMCS field.

Parameters

vmcs	Pointer to the software VMCS.
field	The VMCS field index as used on VMX hardware.

Returns

The value of the VMCS field with the given index.

Definition at line 476 of file __vm-vmx.h.

Referenced by l4_vm_vmx_read().

Here is the caller graph for this function:

13.115.3.12 l4_vm_vmx_read_nat()

```
l4_umword_t l4_vm_vmx_read_nat (
            void * vmcs,
            unsigned field ) [inline]
```

Read a natural width VMCS field.

Parameters

vmcs	Pointer to the software VMCS.
field	The VMCS field index as used on VMX hardware.

Returns

The value of the VMCS field with the given index.

Definition at line 461 of file __vm-vmx.h.

Referenced by l4_vm_vmx_read().

Here is the caller graph for this function:

13.115.3.13 l4_vm_vmx_write()

```
void l4_vm_vmx_write (
            void * vmcs,
            unsigned field,
            l4_uint64_t val )  [inline]
```

Write to an arbitrary VMCS field.

Parameters

vmcs	Pointer to the software VMCS.
field	The VMCS field index as used on VMX hardware.
val	The value that shall be written to the given field.

Definition at line 516 of file __vm-vmx.h.

References l4_vm_vmx_write_16(), l4_vm_vmx_write_32(), l4_vm_vmx_write_64(), and l4_vm_vmx_write_nat().

Here is the call graph for this function:

13.115.3.14 l4_vm_vmx_write_16()

```
void l4_vm_vmx_write_16 (
            void * vmcs,
            unsigned field,
            l4_uint16_t val )  [inline]
```

Write to a 16bit VMCS field.

Parameters

vmcs	Pointer to the software VMCS.
field	The VMCS field index as used on VMX hardware.
val	The value that shall be written to the given field.

Definition at line 500 of file __vm-vmx.h.

Referenced by l4_vm_vmx_write().

Here is the caller graph for this function:

13.115.3.15 l4_vm_vmx_write_32()

```
void l4_vm_vmx_write_32 (
            void * vmcs,
            unsigned field,
            l4_uint32_t val )  [inline]
```

Write to a 32bit VMCS field.

Parameters

vmcs	Pointer to the software VMCS.
field	The VMCS field index as used on VMX hardware.
val	The value that shall be written to the given field.

Definition at line 505 of file __vm-vmx.h.

Referenced by l4_vm_vmx_write().

Here is the caller graph for this function:

13.115.3.16 l4_vm_vmx_write_64()

```
void l4_vm_vmx_write_64 (
            void * vmcs,
            unsigned field,
            l4_uint64_t val )  [inline]
```

Write to a 64bit VMCS field.

Parameters

vmcs	Pointer to the software VMCS.
field	The VMCS field index as used on VMX hardware.
val	The value that shall be written to the given field.

Definition at line 510 of file __vm-vmx.h.

Referenced by l4_vm_vmx_write().

Here is the caller graph for this function:

13.115.3.17 l4_vm_vmx_write_nat()

```
void l4_vm_vmx_write_nat (
            void * vmcs,
            unsigned field,
            l4_umword_t val )  [inline]
```

Write to a natural width VMCS field.

Parameters

vmcs	Pointer to the software VMCS.
field	The VMCS field index as used on VMX hardware.
val	The value that shall be written to the given field.

Definition at line 495 of file __vm-vmx.h.

Referenced by l4_vm_vmx_write().

Here is the caller graph for this function:

13.116 Vbus API

C++ interface of the Vbus API.

Collaboration diagram for Vbus API:

Data Structures

- class L4vbus::Pm< DEC >

  Power-management API mixin.
- class L4vbus::Device

  Device on a L4vbus::Vbus.
- class L4vbus::Icu

  Vbus Interrupt controller API.
- class L4vbus::Vbus

  The virtual bus (Vbus) interface.
- class L4vbus::Gpio_pin

  A GPIO pin.
- class L4vbus::Gpio_module

  A Gpio_module groups multiple GPIO pins together.
- class L4vbus::Pci_host_bridge

  A Pci host bridge.
- class L4vbus::Pci_dev

  A PCI device.

13.116.1 Detailed Description

C++ interface of the Vbus API.

The virtual bus (Vbus) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an Icu (Interrupt controller) for interrupt handling.

The Vbus interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

Refer to L4 Vbus functions for the C API.

Include File

```
#include <l4/vbus/vbus>
```

Include File

```
#include <l4/vbus/vbus_gpio>
```

Include File

```
#include <l4/vbus/vbus_pci>
```

13.117 Video API

Collaboration diagram for Video API:

Data Structures

- struct l4re_video_color_component_t

  Color component structure.
- struct l4re_video_pixel_info_t

  Pixel_info structure.
- struct l4re_video_goos_info_t

  Goos information structure.
- struct l4re_video_view_info_t

  View information structure.
- struct l4re_video_view_t

  C representation of a goos view.

Typedefs

- typedef struct l4re_video_color_component_t l4re_video_color_component_t

  Color component structure.
- typedef struct l4re_video_pixel_info_t l4re_video_pixel_info_t

  Pixel_info structure.
- typedef struct l4re_video_view_info_t l4re_video_view_info_t

  View information structure.
- typedef struct l4re_video_view_t l4re_video_view_t

  C representation of a goos view.

Enumerations

- enum l4re_video_goos_info_flags_t { F_l4re_video_goos_auto_refresh = 0x01 , F_l4re_video_goos_pointer = 0x02 , F_l4re_video_goos_dynamic_views = 0x04 , F_l4re_video_goos_dynamic_buffers = 0x08 }

  Flags of information on the goos.
- enum l4re_video_view_info_flags_t { F_l4re_video_view_none = 0x00 , F_l4re_video_view_set_buffer = 0x01 , F_l4re_video_view_set_buffer_offset = 0x02 , F_l4re_video_view_set_bytes_per_line = 0x04 , F_l4re_video_view_set_pixel = 0x08 , F_l4re_video_view_set_position = 0x10 , F_l4re_video_view_dyn_allocated = 0x20 , F_l4re_video_view_set_background = 0x40 , F_l4re_video_view_set_flags = 0x80 , **F_l4re_video_view_fully_dynamic** , F_l4re_video_view_above = 0x01000 , F_l4re_video_view_flags_mask = 0xff000 }

  Flags of information on a view.

Functions

- int l4re_video_goos_info (l4re_video_goos_t goos, l4re_video_goos_info_t *ginfo) L4_NOTHROW

  Get information on a goos.
- int l4re_video_goos_refresh (l4re_video_goos_t goos, int x, int y, int w, int h) L4_NOTHROW

  Flush a rectangle of pixels of the goos screen.
- int l4re_video_goos_create_buffer (l4re_video_goos_t goos, unsigned long size, l4_cap_idx_t buffer) L4_NOTHROW

  Create a new buffer (memory buffer) for pixel data.
- int l4re_video_goos_delete_buffer (l4re_video_goos_t goos, unsigned idx) L4_NOTHROW

  Delete a pixel buffer.
- int l4re_video_goos_get_static_buffer (l4re_video_goos_t goos, unsigned idx, l4_cap_idx_t buffer) L4_NOTHROW

  Get the data-space capability of the static pixel buffer.
- int l4re_video_goos_create_view (l4re_video_goos_t goos, l4re_video_view_t *view) L4_NOTHROW

  Create a new view (.
- int l4re_video_goos_delete_view (l4re_video_goos_t goos, l4re_video_view_t *view) L4_NOTHROW

  Delete a view.
- int l4re_video_goos_get_view (l4re_video_goos_t goos, unsigned idx, l4re_video_view_t *view) L4_NOTHROW

  Get a view for the given index.
- int l4re_video_view_refresh (l4re_video_view_t *view, int x, int y, int w, int h) L4_NOTHROW

  Flush the given rectangle of pixels of the given view.
- int l4re_video_view_get_info (l4re_video_view_t *view, l4re_video_view_info_t *info) L4_NOTHROW

  Retrieve information about the given view.
- int l4re_video_view_set_info (l4re_video_view_t *view, l4re_video_view_info_t *info) L4_NOTHROW

  Set properties of the view.
- int l4re_video_view_set_viewport (l4re_video_view_t *view, int x, int y, int w, int h, unsigned long bofs) L4_NOTHROW

  Set the viewport parameters of a view.
- int l4re_video_view_stack (l4re_video_view_t *view, l4re_video_view_t *pivot, int behind) L4_NOTHROW

  Change the stacking order in the stack of visible views.

13.117.1 Detailed Description

13.117.2 Typedef Documentation

13.117.2.1 l4re_video_view_t

```
typedef struct l4re_video_view_t l4re_video_view_t
```

C representation of a goos view.

A view is a visible rectangle that provides a view to the contents of a buffer (frame buffer) memory object and is placed on a real screen.

13.117.3 Enumeration Type Documentation

13.117.3.1 l4re_video_goos_info_flags_t

enum l4re_video_goos_info_flags_t

Flags of information on the goos.

Enumerator

F_l4re_video_goos_auto_refresh	The graphics display is automatically refreshed.
F_l4re_video_goos_pointer	We have a mouse pointer.
F_l4re_video_goos_dynamic_views	Supports dynamically allocated views.
F_l4re_video_goos_dynamic_buffers	Supports dynamically allocated buffers.

Definition at line 39 of file goos.h.

13.117.3.2 l4re_video_view_info_flags_t

enum l4re_video_view_info_flags_t

Flags of information on a view.

Enumerator

F_l4re_video_view_none	everything for this view is static (the VESA-FB case)
F_l4re_video_view_set_buffer	buffer object for this view can be changed
F_l4re_video_view_set_buffer_offset	buffer offset can be set
F_l4re_video_view_set_bytes_per_line	bytes per line can be set
F_l4re_video_view_set_pixel	pixel type can be set
F_l4re_video_view_set_position	position on screen can be set
F_l4re_video_view_dyn_allocated	View is dynamically allocated.
F_l4re_video_view_set_background	Set view as background for session.
F_l4re_video_view_set_flags	Set view property flags.
F_l4re_video_view_above	Flag the view as stay on top.
F_l4re_video_view_flags_mask	Mask containing all possible property flags.

Definition at line 33 of file view.h.

13.117.4 Function Documentation

13.117.4.1 l4re_video_goos_create_buffer()

```
int l4re_video_goos_create_buffer (
            l4re_video_goos_t goos,
            unsigned long size,
            l4_cap_idx_t buffer )
```

Create a new buffer (memory buffer) for pixel data.

Parameters

goos	the target object for the operation.
size	the size in bytes for the pixel buffer.
buffer	a capability index to receive the data-space capability for the buffer.

Returns

>=0: The index of the created buffer (used to assign views and for deletion). < 0: on error

13.117.4.2 l4re_video_goos_create_view()

```
int l4re_video_goos_create_view (
            l4re_video_goos_t goos,
            l4re_video_view_t * view )
```

Create a new view (.

See also

l4re_video_view_t)

Parameters

goos	the goos session to use.

Return values

view	the structure will be initialized for the new view.

13.117.4.3 l4re_video_goos_delete_buffer()

```
int l4re_video_goos_delete_buffer (
            l4re_video_goos_t goos,
            unsigned idx )
```

Delete a pixel buffer.

Parameters

goos	the target goos object.
idx	the buffer index of the buffer to delete (the return value of l4re_video_goos_create_buffer())

13.117.4.4 l4re_video_goos_delete_view()

```
int l4re_video_goos_delete_view (
            l4re_video_goos_t goos,
            l4re_video_view_t * view )
```

Delete a view.

Parameters

goos	the goos session to use.
view	the view to delete, the given data-structure is invalid afterwards.

13.117.4.5 l4re_video_goos_get_static_buffer()

```
int l4re_video_goos_get_static_buffer (
            l4re_video_goos_t goos,
            unsigned idx,
            l4_cap_idx_t buffer )
```

Get the data-space capability of the static pixel buffer.

Parameters

goos	The target goos object.
idx	Index of the static buffer.
buffer	A capability index to receive the data-space capability.

This function allows access to static, preexisting pixel buffers. Such static buffers exist for static configurations, such as the VESA framebuffer.

13.117.4.6 l4re_video_goos_get_view()

```
int l4re_video_goos_get_view (
            l4re_video_goos_t goos,
            unsigned idx,
            l4re_video_view_t * view )
```

Get a view for the given index.

Parameters

goos	the target goos session.
idx	the index of the view to retrieve.

Return values

view	the structure will be initialized to the view with the given index.

This function allows to access static views as provided by the VESA framebuffer (the monitor). However, it also allows to access dynamic views created with l4re_video_goos_create_view().

13.117.4.7 l4re_video_goos_info()

```
int l4re_video_goos_info (
            l4re_video_goos_t goos,
            l4re_video_goos_info_t * ginfo )
```

Get information on a goos.

Parameters

goos	Goos object

Return values

ginfo	Pointer to goos information structure.

Returns

0 for success, <0 on error

- -L4_ENODEV
- IPC errors

13.117.4.8 l4re_video_goos_refresh()

```
int l4re_video_goos_refresh (
            l4re_video_goos_t goos,
            int x,
            int y,
            int w,
            int h )
```

Flush a rectangle of pixels of the goos screen.

Parameters

goos	the target object of the operation.
x	the x-coordinate of the upper left corner of the rectangle
y	the y-coordinate of the upper left corner of the rectangle
w	the width of the rectangle to be flushed
h	the height of the rectangle

13.117.4.9 l4re_video_view_get_info()

```
int l4re_video_view_get_info (
            l4re_video_view_t * view,
            l4re_video_view_info_t * info )
```

Retrieve information about the given *view*.

Parameters

view	the target view for the operation.

Return values

info	a buffer receiving the information about the view.

13.117.4.10 l4re_video_view_refresh()

```
int l4re_video_view_refresh (
            l4re_video_view_t * view,
            int x,
            int y,
            int w,
            int h )
```

Flush the given rectangle of pixels of the given *view*.

Parameters

view	the target view of the operation.
x	x-coordinate of the upper left corner
y	y-coordinate of the upper left corner
w	the width of the rectangle
h	the height of the rectangle

13.117.4.11 l4re_video_view_set_info()

```
int l4re_video_view_set_info (
            l4re_video_view_t * view,
            l4re_video_view_info_t * info )
```

Set properties of the view.

Parameters

view	the target view of the operation.
info	the parameters to be set on the view.

Which parameters can be manipulated on a given view can be figured out with l4re_video_view_get_info() and this depends on the concrete instance the view object.

13.117.4.12 l4re_video_view_set_viewport()

```
int l4re_video_view_set_viewport (
            l4re_video_view_t * view,
            int x,
            int y,
            int w,
            int h,
            unsigned long bofs )
```

Set the viewport parameters of a view.

Parameters

view	the target view of the operation.
x	the x-coordinate of the upper left corner on the screen.
y	the y-coordinate of the upper left corner on the screen.
w	the width of the view.
h	the height of the view.
bofs	the offset (in bytes) of the upper left pixel in the memory buffer

This function is a convenience wrapper for l4re_video_view_set_info(), just setting the often changed parameters of a dynamic view. With this function a view can be placed on the real screen and at the same time on its backing buffer.

13.117.4.13 l4re_video_view_stack()

```
int l4re_video_view_stack (
            l4re_video_view_t * view,
            l4re_video_view_t * pivot,
            int behind )
```

Change the stacking order in the stack of visible views.

Parameters

view	the target view for the operation.
pivot	the neighbor view, relative to which *view* shall be stacked. a NULL value allows top (*behind* = 1) and bottom (*behind* = 0) placement of the view.
behind	describes the placement of the view relative to the *pivot* view.

13.118 Virtual Console

C Virtual console interface for simple character based input and output, see L4::Vcon for the C++ interface.

Collaboration diagram for Virtual Console:

Data Structures

- struct l4_vcon_attr_t

  Vcon attribute structure.

Typedefs

- typedef struct l4_vcon_attr_t l4_vcon_attr_t

  Vcon attribute structure.

Enumerations

- enum L4_vcon_size_consts { L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) ∗ sizeof(l4_umword_t) , L4_VCON_READ_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 1) ∗ sizeof(l4_umword_t) }

  Size constants.
- enum L4_vcon_i_flags { L4_VCON_INLCR = 000100 , L4_VCON_IGNCR = 000200 , L4_VCON_ICRNL = 000400 }

  Input flags.
- enum L4_vcon_o_flags { L4_VCON_ONLCR = 000004 , L4_VCON_OCRNL = 000010 , L4_VCON_ONLRET = 000040 }

  Output flags.
- enum L4_vcon_l_flags { L4_VCON_ICANON = 000002 , L4_VCON_ECHO = 000010 }

  Local flags.

Functions

- l4_msgtag_t l4_vcon_send (l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW

  Send data to virtual console.
- l4_msgtag_t l4_vcon_send_u (l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW

  Send data to `this` virtual console.
- long l4_vcon_write (l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW

  Write data to virtual console.
- long l4_vcon_write_u (l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW

  Write data to `this` virtual console.
- int l4_vcon_read (l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW

  Read data from virtual console.
- int l4_vcon_read_u (l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW

  Read data from `this` virtual console.
- int l4_vcon_read_with_flags (l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW

  Read data from virtual console, extended version including flags.
- l4_msgtag_t l4_vcon_set_attr (l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW

  Set attributes of a Vcon.
- l4_msgtag_t l4_vcon_set_attr_u (l4_cap_idx_t vcon, l4_vcon_attr_t const *attr, l4_utcb_t *utcb) L4_NOTHROW

  Set the attributes of `this` virtual console.
- l4_msgtag_t l4_vcon_get_attr (l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW

  Get attributes of a Vcon.
- l4_msgtag_t l4_vcon_get_attr_u (l4_cap_idx_t vcon, l4_vcon_attr_t *attr, l4_utcb_t *utcb) L4_NOTHROW

  Get attributes of `this` virtual console.
- void l4_vcon_set_attr_raw (l4_vcon_attr_t *attr) L4_NOTHROW

  Set terminal attributes to disable all special processing.

13.118.1 Detailed Description

C Virtual console interface for simple character based input and output, see L4::Vcon for the C++ interface.

The interrupt for read events is provided by the virtual key interrupt which, in contrast to hardware IRQs, implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

A server implementing the virtual console protocol has a queue for input events. When the first input event is added to the empty queue, the virtual key interrupt is triggered. Further events are added to the queue without generating further interrupts. The queue is emptied when a client reads all queued input events.

Include File

```
#include <l4/sys/vcon.h>
```

See L4::Vcon for the C++ interface.

13.118.2 Enumeration Type Documentation

13.118.2.1 L4_vcon_i_flags

`enum L4_vcon_i_flags`

Input flags.

Enumerator

L4_VCON_INLCR	Translate NL to CR.
L4_VCON_IGNCR	Ignore CR.
L4_VCON_ICRNL	Translate CR to NL if L4_VCON_IGNCR is not set.

Definition at line 210 of file vcon.h.

13.118.2.2 L4_vcon_l_flags

`enum L4_vcon_l_flags`

Local flags.

Enumerator

L4_VCON_ICANON	Canonical mode.
L4_VCON_ECHO	Echo input.

Definition at line 232 of file vcon.h.

13.118.2.3 L4_vcon_o_flags

`enum L4_vcon_o_flags`

Output flags.

Enumerator

L4_VCON_ONLCR	Translate NL to CR-NL.
L4_VCON_OCRNL	Translate CR to NL.
L4_VCON_ONLRET	Do not output CR.

Definition at line 221 of file vcon.h.

13.118.2.4 L4_vcon_size_consts

```
enum L4_vcon_size_consts
```

Size constants.

Enumerator

L4_VCON_WRITE_SIZE	Maximum size that can be written with one l4_vcon_write call.
L4_VCON_READ_SIZE	Maximum size that can be read with one l4_vcon_read∗ call.

Definition at line 106 of file vcon.h.

13.118.3 Function Documentation

13.118.3.1 l4_vcon_get_attr()

```
l4_msgtag_t l4_vcon_get_attr (
            l4_cap_idx_t vcon,
            l4_vcon_attr_t * attr )  [inline]
```

Get attributes of a Vcon.

Parameters

	vcon	Vcon object.
out	*attr*	Attribute structure.

Returns

Syscall return tag

Definition at line 437 of file vcon.h.

References l4_utcb(), and l4_vcon_get_attr_u().

Here is the call graph for this function:

13.118.3.2 l4_vcon_get_attr_u()

```
l4_msgtag_t l4_vcon_get_attr_u (
            l4_cap_idx_t vcon,
            l4_vcon_attr_t * attr,
            l4_utcb_t * utcb )  [inline]
```

Get attributes of `this` virtual console.

Parameters

	vcon	Capability index of the vcon object.
out	*attr*	Attribute structure. Contains the attributes after a successful call of this function.
	utcb	UTCB pointer of the calling thread.

Returns

Syscall return tag.

Definition at line 419 of file vcon.h.

Referenced by L4::Vcon::get_attr(), and l4_vcon_get_attr().

Here is the caller graph for this function:

13.118.3.3 l4_vcon_read()

```
int l4_vcon_read (
            l4_cap_idx_t vcon,
            char * buf,
            unsigned size )  [inline]
```

Read data from virtual console.

Parameters

	vcon	Vcon object.
out	*buf*	Pointer to data buffer.
	size	Size of buffer in bytes.

Return values

<0	Error code.
>size	If more bytes are to read, `size` bytes are in the buffer `buf`.
<=size	Number of bytes read.

Note

Size must not exceed L4_VCON_READ_SIZE.

Definition at line 393 of file vcon.h.

References l4_utcb(), and l4_vcon_read_u().

Here is the call graph for this function:

13.118.3.4 l4_vcon_read_u()

```
int l4_vcon_read_u (
            l4_cap_idx_t vcon,
            char * buf,
            unsigned size,
            l4_utcb_t * utcb )  [inline]
```

Read data from `this` virtual console.

Parameters

	vcon	Capability index of the vcon object.
out	*buf*	Pointer to data buffer.
	size	Size of the data buffer in bytes.
	utcb	UTCB pointer of the calling thread.

Return values

<0	Error code.
>size	More bytes to read, `size` bytes are in the buffer `buf`.
<=size	Number of bytes read.

Note

Size must not exceed L4_VCON_READ_SIZE.

Definition at line 383 of file vcon.h.

Referenced by l4_vcon_read(), and L4::Vcon::read().

Here is the caller graph for this function:

13.118.3.5 l4_vcon_read_with_flags()

```
int l4_vcon_read_with_flags (
            l4_cap_idx_t vcon,
            char * buf,
            unsigned size )  [inline]
```

Read data from virtual console, extended version including flags.

Parameters

	vcon	Vcon object.
out	*buf*	Pointer to data buffer.
	size	Size of buffer in bytes.

If this function returns a positive value the caller can check the L4_VCON_READ_STAT_BREAK flag bit for a break condition. The bytes read can be obtained by masking the return value with L4_VCON_READ_SIZE_MASK.

If a break condition is signaled, it is always the first event in the transmitted content, i.e. all characters supplied by this read call follow the break condition.

`buf` might be a `NULL`, in this case the input data will be dropped.

Note

Size must not exceed L4_VCON_READ_SIZE.

Return values

<0	Error code.
>size	More bytes to read, `size` bytes are in the buffer `buf`.
<=size	Number of bytes read.

Definition at line 377 of file vcon.h.

13.118.3.6 l4_vcon_send()

```
l4_msgtag_t l4_vcon_send (
            l4_cap_idx_t vcon,
            char const * buf,
            unsigned size )  [inline]
```

Send data to virtual console.

Parameters

vcon	Vcon object.
buf	Pointer to data buffer.
size	Size of buffer in bytes.

Returns

Syscall return tag

Note

Size must not exceed L4_VCON_WRITE_SIZE, a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use l4_ipc_error() to check for send errors, do not use l4_error(), as l4_error() will always return an error.

Examples

examples/sys/utcb-ipc/main.c.

Definition at line 317 of file vcon.h.

References l4_utcb(), and l4_vcon_send_u().

Here is the call graph for this function:

13.118.3.7 l4_vcon_send_u()

```
l4_msgtag_t l4_vcon_send_u (
            l4_cap_idx_t vcon,
            char const * buf,
            unsigned size,
            l4_utcb_t * utcb )  [inline]
```

Send data to `this` virtual console.

Parameters

vcon	Capability index of the Vcon object.
buf	Pointer to the data buffer.
size	Size of the data buffer in bytes.
utcb	UTBC pointer of the calling thread.

Returns

Syscall return tag

Note

Size must not exceed L4_VCON_WRITE_SIZE, a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use l4_ipc_error() to check for send errors, do not use l4_error(), as l4_error() will always return an error.

Definition at line 304 of file vcon.h.

Referenced by l4_vcon_send(), l4_vcon_write_u(), and L4::Vcon::send().

Here is the caller graph for this function:

13.118.3.8 l4_vcon_set_attr()

```
l4_msgtag_t l4_vcon_set_attr (
            l4_cap_idx_t vcon,
            l4_vcon_attr_t const * attr )  [inline]
```

Set attributes of a Vcon.

Parameters

vcon	Vcon object.
attr	Attribute structure.

Returns

Syscall return tag

Definition at line 413 of file vcon.h.

References l4_utcb(), and l4_vcon_set_attr_u().

Here is the call graph for this function:

13.118.3.9 l4_vcon_set_attr_raw()

```
void l4_vcon_set_attr_raw (
            l4_vcon_attr_t * attr )  [inline]
```

Set terminal attributes to disable all special processing.

Removes all flags that would mangle the read or written characters. Also disables echoing and any special processing of characters.

Parameters

in,out	*attr*	Attribute structure to update.

Definition at line 443 of file vcon.h.

Referenced by l4_vcon_attr_t::set_raw().

Here is the caller graph for this function:

13.118.3.10 l4_vcon_set_attr_u()

```
l4_msgtag_t l4_vcon_set_attr_u (
            l4_cap_idx_t vcon,
            l4_vcon_attr_t const * attr,
            l4_utcb_t * utcb )  [inline]
```

Set the attributes of `this` virtual console.

Parameters

vcon	Capability index of the vcon object.
attr	Attribute structure with the attributes for the virtual console.
utcb	UTCB pointer of the calling thread.

Returns

Syscall return tag.

Definition at line 399 of file vcon.h.

Referenced by l4_vcon_set_attr(), and L4::Vcon::set_attr().

Here is the caller graph for this function:

13.118.3.11 l4_vcon_write()

```
long l4_vcon_write (
            l4_cap_idx_t vcon,
            char const * buf,
            unsigned size )  [inline]
```

Write data to virtual console.

Parameters

vcon	Vcon object.
buf	Pointer to data buffer.
size	Size of buffer in bytes.

Return values

<0	Error.
>=0	Number of bytes written to the virtual console

Definition at line 338 of file vcon.h.

References l4_utcb(), and l4_vcon_write_u().

Here is the call graph for this function:

13.118.3.12 l4_vcon_write_u()

```
long l4_vcon_write_u (
            l4_cap_idx_t vcon,
            char const * buf,
            unsigned size,
            l4_utcb_t * utcb )  [inline]
```

Write data to `this` virtual console.

Parameters

vcon	Capability index of the vcon object.
buf	Pointer to the data buffer.
size	Size of the data buffer in bytes.
utcb	UTCB pointer of the calling thread.

Return values

<0	Error.
>=0	Number of bytes written to the virtual console.

Definition at line 323 of file vcon.h.

References l4_error(), l4_msgtag_has_error(), l4_vcon_send_u(), and L4_VCON_WRITE_SIZE.

Referenced by l4_vcon_write(), and L4::Vcon::write().

Here is the call graph for this function:

Here is the caller graph for this function:

13.119 Virtual Machines

Virtual Machine API.

Collaboration diagram for Virtual Machines:

Modules

- VM API for SVM

  Virtual machine API for SVM.
- VM API for TZ

  Virtual Machine API for ARM TrustZone.
- VM API for VMX

  Virtual machine API for VMX.

13.119.1 Detailed Description

Virtual Machine API.

13.120 Virtual Registers (UTCBs)

L4 Virtual Registers (UTCB).

Collaboration diagram for Virtual Registers (UTCBs):

Modules

- ARM Virtual Registers (UTCB)
- Buffer Registers (BRs)
- Message Registers (MRs)
- Thread Control Registers (TCRs)
- amd64 Virtual Registers (UTCB)
- x86 Virtual Registers (UTCB)

Files

- file utcb.h

  UTCB definitions for ARM.
- file utcb.h

  UTCB definitions for amd64.
- file utcb.h

  UTCB definitions for X86.

Typedefs

- typedef struct l4_utcb_t l4_utcb_t

  Opaque type for the UTCB.

Functions

- l4_utcb_t ∗ l4_utcb (void) L4_NOTHROW L4_PURE

  Get the UTCB address.
- l4_msg_regs_t ∗ l4_utcb_mr (void) L4_NOTHROW L4_PURE

  Get the message-register block of a UTCB.
- l4_buf_regs_t ∗ l4_utcb_br (void) L4_NOTHROW L4_PURE

  Get the buffer-register block of a UTCB.
- l4_thread_regs_t ∗ l4_utcb_tcr (void) L4_NOTHROW L4_PURE

  Get the thread-control-register block of a UTCB.

13.120.1 Detailed Description

L4 Virtual Registers (UTCB).

Include File

```
#include <l4/sys/utcb.h>
```

The virtual registers are part of the micro-kernel API and are located in the user-level thread control block (UTCB). The UTCB is a data structure defined by the micro kernel and located on kernel-provided memory. Each L4 thread gets a unique UTCB assigned when it is bound to a task (see Thread Control , l4_thread_control_bind() for more information).

The UTCB is arranged in three blocks of virtual registers.

- Thread Control Registers (TCRs)
- Message Registers (MRs)
- Buffer Registers (BRs)

To access the contents of the virtual registers the l4_utcb_mr(), l4_utcb_tcr(), and l4_utcb_br() functions must be used.

13.120.2 Typedef Documentation

13.120.2.1 l4_utcb_t

```
typedef struct l4_utcb_t l4_utcb_t
```

Opaque type for the UTCB.

To access the contents of the virtual registers the l4_utcb_mr(), l4_utcb_tcr(), and l4_utcb_br() functions must be used.

Definition at line 1 of file utcb.h.

13.120.3 Function Documentation

13.120.3.1 l4_utcb_br()

```
l4_buf_regs_t * l4_utcb_br (
            void  )  [inline]
```

Get the buffer-register block of a UTCB.

Returns

A pointer to the buffer-register block of `u`.

Definition at line 355 of file utcb.h.

Referenced by l4util_ioport_map().

Here is the caller graph for this function:

13.120.3.2 l4_utcb_mr()

```
l4_msg_regs_t * l4_utcb_mr (
            void ) [inline]
```

Get the message-register block of a UTCB.

Returns

A pointer to the message-register block of `u`.

Examples

examples/sys/aliens/main.c, examples/sys/ipc/ipc_example.c, examples/sys/singlestep/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 352 of file utcb.h.

Referenced by l4util_ioport_map().

Here is the caller graph for this function:

13.120.3.3 l4_utcb_tcr()

```
l4_thread_regs_t * l4_utcb_tcr (
            void ) [inline]
```

Get the thread-control-register block of a UTCB.

Returns

A pointer to the thread-control-register block of `u`.

Definition at line 358 of file utcb.h.

13.121 amd64 Virtual Registers (UTCB)

Collaboration diagram for amd64 Virtual Registers (UTCB):

Data Structures

- struct l4_exc_regs_t

  UTCB structure for exceptions.

Typedefs

- typedef struct l4_exc_regs_t l4_exc_regs_t

  UTCB structure for exceptions.

Enumerations

- enum L4_utcb_consts_amd64

  UTCB constants for AMD64.

13.121.1 Detailed Description

13.122 vCPU API

vCPU API.

Collaboration diagram for vCPU API:

Data Structures

- struct l4_vcpu_state_t

  State of a vCPU.
- struct l4_vcpu_regs_t

  vCPU registers.
- struct l4_vcpu_ipc_regs_t

  vCPU message registers.

Typedefs

- typedef struct l4_vcpu_state_t l4_vcpu_state_t

  State of a vCPU.
- typedef struct l4_vcpu_regs_t l4_vcpu_regs_t

  vCPU registers.
- typedef struct l4_vcpu_ipc_regs_t l4_vcpu_ipc_regs_t

  vCPU message registers.
- typedef struct l4_vcpu_regs_t l4_vcpu_regs_t

  vCPU registers.
- typedef struct l4_vcpu_ipc_regs_t l4_vcpu_ipc_regs_t

  vCPU message registers.
- typedef struct l4_vcpu_regs_t l4_vcpu_regs_t

  vCPU registers.
- typedef struct l4_vcpu_ipc_regs_t l4_vcpu_ipc_regs_t

  vCPU message registers.

Enumerations

- enum L4_vcpu_state_flags {
  L4_VCPU_F_IRQ = 0x01 , L4_VCPU_F_PAGE_FAULTS = 0x02 , L4_VCPU_F_EXCEPTIONS = 0x04 , L4_VCPU_F_USER_MODE = 0x20 ,
  L4_VCPU_F_FPU_ENABLED = 0x80 }

  State flags of a vCPU.
- enum L4_vcpu_sticky_flags { L4_VCPU_SF_IRQ_PENDING = 0x01 }

  Sticky flags of a vCPU.
- enum L4_vcpu_state_offset { L4_VCPU_OFFSET_EXT_STATE = 0x400 , L4_VCPU_OFFSET_EXT_INFOS = 0x200 }

  Offsets for vCPU state layouts.

13.122.1 Detailed Description

vCPU API.

The vCPU API in L4Re implements virtual processors (vCPUs) on top of L4::Thread. This API can be used for user level threading, operating system rehosting (see L4Linux) and virtualization.

You switch a thread into `vCPU` operation with L4::Thread::vcpu_control.

In vCPU mode, incoming IPC can be redirected to a handler function. If an IPC is sent to the vCPU, the thread's normal execution is interrupted and the handler called. Which kind of IPC is redirected is specified by the L4_vcpu_state_flags set in the l4_vcpu_state_t::state field of `vcpu_state`. All events enabled

in the `vcpu_state` field are redirected to the handler. The handler is set via l4_vcpu_state_t::entry_ip and l4_vcpu_state_t::entry_sp. IPC redirection works independent of "kernel" and "user" mode, but see l4_vcpu_↩state::entry_sp. When the entry handler is called, the UTCB contains the result of the IPC and content normally found in CPU register is in l4_vcpu_state_t::i.

Furthermore, the thread can execute in the context of different tasks, called the "kernel" and the "user" mode. The kernel task is the one to which the thread was originally bound via L4::Thread::control(). Execution starts in the kernel task and it is always switched to when the asynchronous IPC handler is invoked. When returning from the handler via L4::Thread::vcpu_resume_start() and L4::Thread::vcpu_resume_commit(), a different user task can be specified by setting l4_vcpu_state_t::user_task and enabling the L4_VCPU_F_USER_MODE flag in l4_vcpu_state_t::state.

If the L4_VCPU_F_USER_MODE flag is enabled, the following flags will be automatically enabled in l4_vcpu_state_t::state on L4::Thread::vcpu_resume_commit():

- L4_VCPU_F_IRQ
- L4_VCPU_F_PAGE_FAULTS
- L4_VCPU_F_EXCEPTIONS

When the kernel mode is entered, the following flags will be automatically disabled in l4_vcpu_state_t::state:

- L4_VCPU_F_IRQ
- L4_VCPU_F_PAGE_FAULTS
- L4_VCPU_F_USER_MODE

Extended vCPU operation is used for hardware CPU virtualization. It can be enabled with L4::Thread::vcpu_control_ext().

vCPU Support Library defines a convenience API for working with vCPUs.

See also

vCPU Support Library

13.122.2 Enumeration Type Documentation

13.122.2.1 L4_vcpu_state_flags

```
enum L4_vcpu_state_flags
```

State flags of a vCPU.

Enumerator

L4_VCPU_F_IRQ	Receiving of IRQs and IPC enabled. While this flag is not set, the corresponding vCPU thread will not receive any IPC and threads attempting to send an IPC to this thread will block (according to the selected send timeout). **Note** On L4::Thread::vcpu_resume_commit() this flag is automatically enabled in l4_vcpu_state_t::state if L4_VCPU_F_USER_MODE is enabled. When the kernel mode is entered, this flags is automatically disabled in l4_vcpu_state_t::state.
L4_VCPU_F_PAGE_FAULTS	Page faults enabled. If this flag is set, a page fault switches to kernel mode (potentially causing a VM exit) and calls the entry handler. If this flag is not set, a page fault generates a page fault IPC to the pager of the vCPU thread. **Note** IPC redirection for page faults controlled by this flag works independent of "kernel" and "user" mode. On L4::Thread::vcpu_resume_commit() this flag is automatically enabled in l4_vcpu_state_t::state if L4_VCPU_F_USER_MODE is enabled. When the kernel mode is entered, this flags is automatically disabled in l4_vcpu_state_t::state.
L4_VCPU_F_EXCEPTIONS	Exceptions enabled. If this flag is set, then, on the event of an exception, the vCPU switches to kernel mode (potentially causing a VM exit) and calls the entry handler. If this flag is not set, an exception generates an exception IPC to the exception handler of the vCPU thread. **Note** IPC redirection for exceptions controlled by this flag works independent of "kernel" and "user" mode. On L4::Thread::vcpu_resume_commit() this flag is automatically enabled in l4_vcpu_state_t::state if L4_VCPU_F_USER_MODE is enabled.
L4_VCPU_F_USER_MODE	User task will be used. If set, the vCPU switches to user mode on next L4::Thread::vcpu_resume_commit(). If clear, the vCPU stays in "kernel" mode. **Note** When the kernel mode is entered, this flags is automatically disabled in l4_vcpu_state_t::state.
L4_VCPU_F_FPU_ENABLED	FPU enabled. This flag is only relevant if L4_VCPU_F_USER_MODE is set. Setting this flag allows code in vCPU mode to use the FPU. IF this flag is not set, any FPU operation will trigger a corresponding exception (FPU fault).

Definition at line 104 of file vcpu.h.

13.122.2.2 L4_vcpu_state_offset

```
enum L4_vcpu_state_offset
```

Offsets for vCPU state layouts.

Enumerator

L4_VCPU_OFFSET_EXT_STATE	Offset where extended state begins.
L4_VCPU_OFFSET_EXT_INFOS	Offset where extended infos begin.

Definition at line 181 of file vcpu.h.

13.122.2.3 L4_vcpu_sticky_flags

```
enum L4_vcpu_sticky_flags
```

Sticky flags of a vCPU.

Enumerator

L4_VCPU_SF_IRQ_PENDING	An event is pending: Either an IRQ or another thread attempts to send an IPC to this vCPU thread.

Definition at line 170 of file vcpu.h.

13.123 vCPU Support Library

vCPU handling functionality.

Collaboration diagram for vCPU Support Library:

Modules

- Extended vCPU support

  extended vCPU handling functionality.

Data Structures

- class L4vcpu::State

  C++ implementation of state word in the vCPU area.
- class L4vcpu::Vcpu

  C++ implementation of the vCPU save state area.

Functions

- void l4vcpu_irq_disable (l4_vcpu_state_t ∗vcpu) L4_NOTHROW

  Disable a vCPU for event delivery.
- unsigned l4vcpu_irq_disable_save (l4_vcpu_state_t ∗vcpu) L4_NOTHROW

  Disable a vCPU for event delivery and return previous state.
- void l4vcpu_irq_enable (l4_vcpu_state_t ∗vcpu, l4_utcb_t ∗utcb, l4vcpu_event_hndl_t do_event_work_cb, l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW

  Enable a vCPU for event delivery.
- void l4vcpu_irq_restore (l4_vcpu_state_t ∗vcpu, unsigned s, l4_utcb_t ∗utcb, l4vcpu_event_hndl_t do_event_work_cb, l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW

  Restore a previously saved IRQ/event state.
- void l4vcpu_wait_for_event (l4_vcpu_state_t ∗vcpu, l4_utcb_t ∗utcb, l4vcpu_event_hndl_t do_event_work_cb, l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW

  Wait for event.
- void l4vcpu_print_state (const l4_vcpu_state_t ∗vcpu, const char ∗prefix) L4_NOTHROW

  Print the state of a vCPU.
- int l4vcpu_is_irq_entry (l4_vcpu_state_t const ∗vcpu) L4_NOTHROW

  Return whether the entry reason was an IRQ/IPC message.
- int l4vcpu_is_page_fault_entry (l4_vcpu_state_t const ∗vcpu) L4_NOTHROW

  Return whether the entry reason was a page fault.

13.123.1 Detailed Description

vCPU handling functionality.

This library provides convenience functionality on top of the l4sys vCPU interface to ease programming. It wraps commonly used code and abstracts architecture depends parts as far as reasonable.

13.123.2 Function Documentation

13.123.2.1 l4vcpu_irq_disable()

```
void l4vcpu_irq_disable (
            l4_vcpu_state_t * vcpu )  [inline]
```

Disable a vCPU for event delivery.

Parameters

vcpu	Pointer to vCPU area.

Definition at line 208 of file vcpu.h.

References l4_barrier(), and L4_VCPU_F_IRQ.

Referenced by L4vcpu::Vcpu::irq_disable(), l4vcpu_irq_disable_save(), and l4vcpu_irq_restore().

Here is the call graph for this function:

Here is the caller graph for this function:

13.123.2.2 l4vcpu_irq_disable_save()

```
unsigned l4vcpu_irq_disable_save (
            l4_vcpu_state_t * vcpu )  [inline]
```

Disable a vCPU for event delivery and return previous state.

Parameters

vcpu	Pointer to vCPU area.

Returns

IRQ state before disabling IRQs.

Definition at line 216 of file vcpu.h.

References l4vcpu_irq_disable().

Referenced by L4vcpu::Vcpu::irq_disable_save().

Here is the call graph for this function:

Here is the caller graph for this function:

13.123.2.3 l4vcpu_irq_enable()

```
void l4vcpu_irq_enable (
            l4_vcpu_state_t * vcpu,
            l4_utcb_t * utcb,
            l4vcpu_event_hndl_t do_event_work_cb,
            l4vcpu_setup_ipc_t setup_ipc )  [inline]
```

Enable a vCPU for event delivery.

Parameters

vcpu	Pointer to vCPU area.
utcb	Utcb pointer of the calling vCPU.
do_event_work_cb	Call-back function that is called in case an event (such as an interrupt) is pending.
setup_ipc	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 239 of file vcpu.h.

References l4_barrier(), L4_IPC_BOTH_TIMEOUT_0, L4_LIKELY, L4_VCPU_F_IRQ, and L4_VCPU_SF_IRQ_PENDING.

Referenced by L4vcpu::Vcpu::irq_enable(), and l4vcpu_irq_restore().

Here is the call graph for this function:

Here is the caller graph for this function:

13.123.2.4 l4vcpu_irq_restore()

```
void l4vcpu_irq_restore (
            l4_vcpu_state_t * vcpu,
            unsigned s,
            l4_utcb_t * utcb,
            l4vcpu_event_hndl_t do_event_work_cb,
            l4vcpu_setup_ipc_t setup_ipc )  [inline]
```

Restore a previously saved IRQ/event state.

Parameters

vcpu	Pointer to vCPU area.
s	IRQ state to be restored.
utcb	Utcb pointer of the calling vCPU.
do_event_work_cb	Call-back function that is called in case an event (such as an interrupt) is pending after enabling.
setup_ipc	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 264 of file vcpu.h.

References L4_VCPU_F_IRQ, l4vcpu_irq_disable(), and l4vcpu_irq_enable().

Referenced by L4vcpu::Vcpu::irq_restore().

Here is the call graph for this function:

Here is the caller graph for this function:

13.123.2.5 l4vcpu_is_irq_entry()

```
int l4vcpu_is_irq_entry (
            l4_vcpu_state_t const * vcpu )  [inline]
```

Return whether the entry reason was an IRQ/IPC message.

Parameters

vcpu	Pointer to vCPU area.

return 0 if not, !=0 otherwise.

Referenced by L4vcpu::Vcpu::is_irq_entry().

Here is the caller graph for this function:

13.123.2.6 l4vcpu_is_page_fault_entry()

```
int l4vcpu_is_page_fault_entry (
            l4_vcpu_state_t const * vcpu )  [inline]
```

Return whether the entry reason was a page fault.

Parameters

vcpu	Pointer to vCPU area.

return 0 if not, !=0 otherwise.

Referenced by L4vcpu::Vcpu::is_page_fault_entry().

Here is the caller graph for this function:

13.123.2.7 l4vcpu_print_state()

```
void l4vcpu_print_state (
            const l4_vcpu_state_t * vcpu,
            const char * prefix )
```

Print the state of a vCPU.

Parameters

vcpu	Pointer to vCPU area.
prefix	A prefix for each line printed.

Referenced by L4vcpu::Vcpu::print_state().

Here is the caller graph for this function:

13.123.2.8 l4vcpu_wait_for_event()

```
void l4vcpu_wait_for_event (
            l4_vcpu_state_t * vcpu,
            l4_utcb_t * utcb,
            l4vcpu_event_hndl_t do_event_work_cb,
            l4vcpu_setup_ipc_t setup_ipc )  [inline]
```

Wait for event.

Parameters

vcpu	Pointer to vCPU area.
utcb	Utcb pointer of the calling vCPU.
do_event_work_cb	Call-back function that is called when the vCPU awakes and needs to handle an event/IRQ.
setup_ipc	Function call-back that is called right before any IPC operation.

Note that event delivery remains disabled after this function returns.

Definition at line 277 of file vcpu.h.

References L4_IPC_NEVER.

Referenced by L4vcpu::Vcpu::wait_for_event().

Here is the caller graph for this function:

13.124 x86 Virtual Registers (UTCB)

Collaboration diagram for x86 Virtual Registers (UTCB):

Data Structures

- struct l4_exc_regs_t

  UTCB structure for exceptions.

Typedefs

- typedef struct l4_exc_regs_t l4_exc_regs_t

  UTCB structure for exceptions.

Enumerations

- enum L4_utcb_consts_x86 {
  L4_UTCB_EXCEPTION_REGS_SIZE = 19 , L4_UTCB_GENERIC_DATA_SIZE = 63 , L4_UTCB_GENERIC_BUFFERS_SIZE = 58 , L4_UTCB_MSG_REGS_OFFSET = 0 ,
  L4_UTCB_BUF_REGS_OFFSET = 64 ∗ sizeof(l4_umword_t) , L4_UTCB_THREAD_REGS_OFFSET = 123 ∗ sizeof(l4_umword_t) , L4_UTCB_INHERIT_FPU = 1UL << 24 , L4_UTCB_OFFSET = 512 }

  UTCB constants for x86.

13.124.1 Detailed Description

13.124.2 Enumeration Type Documentation

13.124.2.1 L4_utcb_consts_x86

```
enum L4_utcb_consts_x86
```

UTCB constants for x86.

Enumerator

L4_UTCB_EXCEPTION_REGS_SIZE	Number if message registers used for exception IPC.
L4_UTCB_GENERIC_DATA_SIZE	Total number of message register (MRs) available.
L4_UTCB_GENERIC_BUFFERS_SIZE	Total number of buffer registers (BRs) available.
L4_UTCB_MSG_REGS_OFFSET	Offset of MR[0] relative to the UTCB pointer.
L4_UTCB_BUF_REGS_OFFSET	Offset of BR[0] relative to the UTCB pointer.
L4_UTCB_THREAD_REGS_OFFSET	Offset of TCR[0] relative to the UTCB pointer.
L4_UTCB_INHERIT_FPU	BDR flag to accept reception of FPU state.
L4_UTCB_OFFSET	Offset of two consecutive UTCBs.

Definition at line 41 of file utcb.h.

Chapter 14

Namespace Documentation

14.1 cxx Namespace Reference

Our C++ library.

Namespaces

- Bits

  Internal helpers for the cxx package.

Data Structures

- class Auto_ptr

  Smart pointer with automatic deletion.
- class Avl_map

  AVL tree based associative container.
- class Avl_set

  AVL set for simple compareable items.
- class Avl_tree_node

  Node of an AVL tree.
- class Avl_tree

  A generic AVL tree.
- class Bitfield

  Definition for a member (part) of a bit field.
- class Bitmap_base

  Basic bitmap abstraction.
- class Bitmap

  A static bit map.
- class H_list_item_t

  Basic element type for a double-linked H_list.
- class H_list

  General double-linked list of unspecified cxx::H_list_item elements.
- struct H_list_t

  Double-linked list of typed H_list_item_t elements.

- class List_item

  Basic list item.
- class List

  Doubly linked list, with internal allocation.
- class List_alloc

  Standard list-based allocator.
- struct Pair

  Pair of two values.
- class Pair_first_compare

  Comparison functor for Pair.
- class Ref_ptr

  A reference-counting pointer with automatic cleanup.
- struct Ref_obj_list_item

  Item for list linked via cxx::Ref_ptr with default refence counting.
- class Base_slab

  Basic slab allocator.
- class Slab

  Slab allocator for object of type `Type`.
- class Base_slab_static

  Merged slab allocator (allocators for objects of the same size are merged together).
- class Slab_static

  Merged slab allocator (allocators for objects of the same size are merged together).
- class S_list

  Simple single-linked list.
- class static_vector

  Simple encapsulation for a dynamically allocated array.
- class Nothrow

  Helper type to distinguish the `oeprator new` *version that does not throw exceptions.*
- class New_allocator

  Standard allocator based on `operator new ()` .
- struct Lt_functor

  Generic comparator class that defaults to the less-than operator.
- class String

  Allocation free string class with explicit length field.
- class Weak_ref_base

  Generic (base) weak reference to some object.
- class Weak_ref

  Typed weak reference to an object of type `T`.

Typedefs

- typedef H_list_item_t< void > H_list_item

  Untyped list item.
- template<typename T >
  using Ref_ptr_list_item = Bits::Smart_ptr_list_item< T, cxx::Ref_ptr< T > >

  Item for list linked with cxx::Ref_ptr.
- template<typename T >
  using Ref_ptr_list = Bits::Smart_ptr_list< Ref_ptr_list_item< T > >

  Single-linked list where elements are connected via a cxx::Ref_ptr.

- template<typename T >
  using Unique_ptr_list_item = Bits::Smart_ptr_list_item< T, cxx::unique_ptr< T > >

  Item for list linked with cxx::unique_ptr.
- template<typename T >
  using Unique_ptr_list = Bits::Smart_ptr_list< Unique_ptr_list_item< T > >

  Single-linked list where elements are connected with a cxx::unique_ptr.

Functions

- template<typename T1 >
  T1 min (T1 a, T1 b)

  Get the minimum of a and b.
- template<typename T1 >
  T1 max (T1 a, T1 b)

  Get the maximum of a and b.

14.1.1 Detailed Description

Our C++ library.

Small Low-Level C++ Library.

Strings.

Various kinds of C++ utilities.

14.2 cxx::Bits Namespace Reference

Internal helpers for the cxx package.

Data Structures

- struct Avl_map_get_key

  Key-getter for Avl_map.
- struct Avl_set_get_key

  Internal, key-getter for Avl_set nodes.
- class Base_avl_set

  Internal: AVL set with internally managed nodes.
- class Bst

  Basic binary search tree (BST).
- struct Direction

  The direction to go in a binary search tree.
- class Bst_node

  Basic type of a node in a binary search tree (BST).
- class Basic_list

  Internal: Common functions for all head-based list implementations.
- class Smart_ptr_list_item

  List item for an arbitrary item in a Smart_ptr_list.
- class Smart_ptr_list

  List of smart-pointer-managed objects.

14.2.1 Detailed Description

Internal helpers for the cxx package.

14.3 L4 Namespace Reference

L4 low-level kernel interface.

Namespaces

- Ipc

  IPC related functionality.
- Ipc_svr

  Helper classes for L4::Server instantiation.
- Typeid

  Definition of interface data-type helpers.
- Types

  L4 basic type helpers for C++.

Data Structures

- struct Type_info

  Dynamic Type Information for L4Re Interfaces.
- struct Kobject_typeid

  Meta object for handling access to type information of Kobjects.
- struct Kobject_typeid< void >

  Minimalistic ID for `void` interface.
- class Kobject_t

  Helper class to create an L4Re interface class that is derived from a single base class.
- class Kobject_2t

  Helper class to create an L4Re interface class that is derived from two base classes (see `L4::Kobject_t`).
- struct Kobject_3t

  Helper class to create an L4Re interface class that is derived from three base classes (see `L4::Kobject_t`).
- struct Kobject_demand

  Get the combined server-side resource requirements for all type T...
- struct Proto_t

  Data type for defining protocol numbers.
- struct Kobject_x

  Generic Kobject inheritance template.
- class Vm

  Virtual machine host address space.
- class Arm_smccc

  Wrapper for function calls that follow the ARM SMC/HVC calling convention.
- class Cap

  C++ interface for capabilities.
- class Cap_base

  Base class for all kinds of capabilities.

- struct Epiface

  Base class for interface implementations.
- struct Epiface_t0

  Epiface mixin for generic Kobject-based interfaces.
- struct Irqep_t

  Epiface implementation for interrupt handlers.
- class Registry_iface

  Abstract interface for object registries.
- struct Epiface_t

  Epiface implementation for Kobject-based interface implementations.
- class Basic_registry

  This registry returns the corresponding server object based on the label of an Ipc_gate.
- class Server

  Basic server loop for handling client requests.
- class Debugger

  C++ kernel debugger API.
- class Exception

  Exception interface.
- class Factory

  C++ Factory interface, see Factory for the C interface.
- class Iommu

  Interface for IO-MMUs used for DMA remapping.
- class Ipc_gate

  The C++ IPC gate interface, see IPC-Gate API for the C interface.
- class Irq_eoi

  Interface for sending an unmask message to an object.
- struct Triggerable

  Interface that allows an object to be triggered by some source.
- class Irq

  C++ Irq interface, see IRQs for the C interface.
- struct Irq_mux

  IRQ multiplexer for shared IRQs.
- class Icu

  C++ Icu interface, see Interrupt controller for the C interface.
- class Kobject

  Base class for all kinds of kernel objects and remote objects, referenced by capabilities.
- class Meta

  Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects.
- class Io_pager

  Io_pager interface.
- class Pager

  Pager interface including the Io_pager interface.
- class Platform_control

  L4 C++ interface for controlling platform-wide properties, see Platform Control C API for the C interface.
- class Rcv_endpoint

  Interface for kernel objects that allow to receive IPC from them.
- class Scheduler

  C++ interface of the Scheduler kernel object, see Scheduler for the C interface.
- struct Semaphore

  C++ Kernel-provided semaphore interface, see Kernel-provided semaphore for the C interface.

- class Smart_cap

  Smart capability class.
- class Task

  C++ interface of the Task kernel object, see Task for the C interface.
- class Thread

  C++ L4 kernel thread interface, see Thread for the C interface.
- class Vcon

  C++ L4 Vcon interface, see Virtual Console for the C interface.
- class Poll_timeout_kipclock

  A polling timeout based on the L4Re clock.
- class Alloc_list

  A simple list-based allocator.
- class IOModifier

  Modifier class for the IO stream.
- class Exception_tracer

  Back-trace support for exceptions.
- class Base_exception

  Base class for all exceptions, thrown by the L4Re framework.
- class Runtime_error

  Exception for an abstract runtime error.
- class Out_of_memory

  Exception signalling insufficient memory.
- class Element_already_exists

  Exception for duplicate element insertions.
- class Unknown_error

  Exception for an unknown condition.
- class Element_not_found

  Exception for a failed lookup (element not found).
- class Invalid_capability

  Indicates that an invalid object was invoked.
- class Com_error

  Error conditions during IPC.
- class Bounds_error

  Access out of bounds.
- class Server_object

  Abstract server object to be used with L4::Server and L4::Basic_registry.
- struct Server_object_t

  Base class (template) for server implementing server objects.
- struct Server_object_x

  Helper class to implement p_dispatch based server objects.
- struct Irq_handler_object

  Server object base class for handling IRQ messages.
- class String

  A null-terminated string container class.

Typedefs

- typedef int Opcode

  Data type for RPC opcodes.

Enumerations

- enum { PROTO_ANY = 0 , PROTO_EMPTY = -19 }

Functions

- template<typename T >
  Type_info const * kobject_typeid () noexcept

  Get the L4::Type_info for the L4Re interface given in `T`.
- template<typename T , typename F >
  Cap< T > cap_dynamic_cast (Cap< F > const &c) noexcept

  `dynamic_cast` for capabilities.
- template<typename T , typename F >
  Cap< T > cap_cast (Cap< F > const &c) noexcept

  static_cast for capabilities.
- template<typename T , typename F >
  Cap< T > cap_reinterpret_cast (Cap< F > const &c) noexcept

  reinterpret_cast for capabilities.
- template<typename T >
  constexpr T trunc_order (T val, unsigned char order)

  Round a value down so the given number of lsb is zero.
- template<typename T >
  constexpr T round_order (T val, unsigned char order)

  Round a value up so the given number of lsb is zero.
- template<typename T , typename F , typename SMART >
  Smart_cap< T, SMART > cap_cast (Smart_cap< F, SMART > const &c) noexcept

  `static_cast` for (smart) capabilities.
- template<typename T , typename F , typename SMART >
  Smart_cap< T, SMART > cap_reinterpret_cast (Smart_cap< F, SMART > const &c) noexcept

  `reinterpret_cast` for (smart) capabilities.
- void throw_ipc_exception (L4::Cap< void > const &o, l4_msgtag_t const &err, l4_utcb_t *utcb)

  Throw an L4 IPC error as exception.
- void throw_ipc_exception (void const *o, l4_msgtag_t const &err, l4_utcb_t *utcb)

  Throw an L4 IPC error as exception.

Variables

- IOModifier const hex

  Modifies the stream to print numbers as hexadecimal values.
- IOModifier const dec

  Modifies the stream to print numbers as decimal values.
- BasicOStream cout

  Standard output stream.
- BasicOStream cerr

  Standard error stream.

14.3.1 Detailed Description

L4 low-level kernel interface.

14.3.2 Enumeration Type Documentation

14.3.2.1 anonymous enum

```
anonymous enum
```

Enumerator

PROTO_ANY	Default protocol used by Kobject_t and Kobject_x.
PROTO_EMPTY	Empty protocol for empty APIs.

Definition at line 55 of file __typeinfo.h.

14.3.3 Function Documentation

14.3.3.1 cap_cast() [1/2]

```
template<typename T , typename F >
Cap<T> L4::cap_cast (
            Cap< F > const & c )  [inline], [noexcept]
```

static_cast for capabilities.

Template Parameters

T	The target type of the capability
F	The source type (and is usually implicitly set)

Parameters

c	The source capability that shall be casted

Returns

A capability typed to the interface `T`.

The use of this cast operator is similar to the `static_cast<>()` for C++ pointers. It does the same type checking and adjustments like C++ does on pointers.

Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_cast<L4::Icu>(obj);
```

Definition at line 380 of file capability.h.

14.3.3.2 cap_cast() [2/2]

```
template<typename T , typename F , typename SMART >
Smart_cap<T, SMART> L4::cap_cast (
            Smart_cap< F, SMART > const & c )  [inline], [noexcept]
```

`static_cast` for (smart) capabilities.

Template Parameters

T	Type to cast the capability to.
F	(implicit) Type of the passed capability.
SMART	(implicit) Class implementing the Smart_cap interface.

Parameters

c	Capability to be casted.

Returns

A smart capability with new type `T`.

Definition at line 203 of file smart_capability.

14.3.3.3 cap_dynamic_cast()

```
template<typename T , typename F >
Cap<T> L4::cap_dynamic_cast (
            Cap< F > const & c )  [inline], [noexcept]
```

`dynamic_cast` for capabilities.

Template Parameters

T	The target type of the capability.
F	The source type (is usually implicitly set).

Parameters

c	The source capability that shall be casted.

Return values

Cap<T>	Capability of target interface `T`.
L4_INVALID_CAP	`c` does not support the target interface `T` or the L4::Meta interface.

The use of this cast operator is similar to the `dynamic_cast<>()` for C++ pointers. It also induces overhead, because it uses the meta interface (L4::Meta) to do runtime type checking.

Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_dynamic_cast<L4::Icu>(obj);
```

Definition at line 125 of file capability.

References l4_error().

Here is the call graph for this function:

14.3.3.4 cap_reinterpret_cast() [1/2]

```
template<typename T , typename F >
Cap<T> L4::cap_reinterpret_cast (
            Cap< F > const & c )  [inline], [noexcept]
```

reinterpret_cast for capabilities.

Template Parameters

T	The target type of the capability
F	The source type (and is usually implicitly set)

Parameters

c	The source capability that shall be casted

Returns

A capability typed to the interface `T`.

The use of this cast operator is similar to the `reinterpret_cast<>()` for C++ pointers. It does not do any type checking or type adjustment.

Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_reinterpret_cast<L4::Icu>(obj);
```

Definition at line 411 of file capability.h.

14.3.3.5 cap_reinterpret_cast() [2/2]

```
template<typename T , typename F , typename SMART >
Smart_cap<T, SMART> L4::cap_reinterpret_cast (
            Smart_cap< F, SMART > const & c )  [inline], [noexcept]
```

`reinterpret_cast` for (smart) capabilities.

Template Parameters

T	Type to cast the capability to.
F	(implicit) Type of the passed capability.
SMART	(implicit) Class implementing the Smart_cap interface.

Parameters

c	Capability to be casted.

Returns

A smart capability with new type `T`.

Definition at line 222 of file smart_capability.

14.3.3.6 round_order()

```
template<typename T >
constexpr T L4::round_order (
            T val,
            unsigned char order )  [constexpr]
```

Round a value up so the given number of lsb is zero.

Template Parameters

T	The type of the value (shall be some integral type.

Parameters

val	The value to rund up to the next multiple of 2^order.

Parameters

order	order (2^order) to round up to.

Returns

val rounded up to the next 2^order.

Definition at line 32 of file consts.

Referenced by L4Re::Dataspace::map_region().

Here is the caller graph for this function:

14.3.3.7 throw_ipc_exception() [1/2]

```
void L4::throw_ipc_exception (
            L4::Cap< void > const & o,
            l4_msgtag_t const & err,
            l4_utcb_t * utcb )  [inline]
```

Throw an L4 IPC error as exception.

Parameters

o	The client side object, for which the IPC was invoked.
err	The IPC result code (error code).
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Definition at line 41 of file ipc_helper.

References l4_msgtag_t::has_error().

Referenced by throw_ipc_exception().

Here is the call graph for this function:

Here is the caller graph for this function:

14.3.3.8 throw_ipc_exception() [2/2]

```
void L4::throw_ipc_exception (
            void const * o,
            l4_msgtag_t const & err,
            l4_utcb_t * utcb )  [inline]
```

Throw an L4 IPC error as exception.

Parameters

o	The client side object, for which the IPC was invoked.
err	The IPC result code (error code).
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Definition at line 58 of file ipc_helper.

References throw_ipc_exception().

Here is the call graph for this function:

14.3.3.9 trunc_order()

```
template<typename T >
constexpr T L4::trunc_order (
            T val,
            unsigned char order )  [constexpr]
```

Round a value down so the given number of lsb is zero.

Template Parameters

T	The type of the value (shall be some integral type.

Parameters

val	The value where the given lsb shall be masked.
order	the number of least significant bits (lsb) to mask.

Returns

val with order lsb masked to zero.

Definition at line 18 of file consts.

Referenced by L4Re::Dataspace::map_region().

Here is the caller graph for this function:

14.4 L4::Ipc Namespace Reference

IPC related functionality.

Namespaces

- Msg

  IPC Message related functionality.

Data Structures

- struct Array_ref

  Array reference data type for arrays located in the message.
- struct Array

  Array data type for dynamically sized arrays in RPCs.
- struct Array_in_buf

  Server-side copy in buffer for Array.
- struct Call

  RPC attribute for a standard RPC call.
- struct Call_zero_send_timeout

  RPC attribute for an RPC call, with zero send timeout.
- struct Call_t

  RPC attribute for an RPC call with required rights.
- struct Send_only

  RPC attribute for a send-only RPC.
- struct Ret_array

  Dynamically sized output array of type T.
- struct Out

  Mark an argument as a output value in an RPC signature.
- struct In_out

  Mark an argument as in-out argument.
- struct As_value

  Pass the argument as plain data value.
- struct Opt

  Attribute for defining an optional RPC argument.
- class Small_buf

  A receive item for receiving a single capability.
- class Snd_item

  RPC wrapper for a send item.
- class Buf_item

  RPC warpper for a receive item.
- class Gen_fpage

  Generic RPC wrapper for L4 flex-pages.
- class Cap

  Capability type for RPC interfaces (see `L4::Cap<T>`).
- class Varg

  Variably sized RPC argument.
- class Varg_list

  Self-contained list of variable-sized RPC parameters.
- class Varg_list_ref

  List of variable-sized RPC parameters as received by the server.
- class Str_cp_in

  Abstraction for extracting a zero-terminated string from an Ipc::Istream.
- class Msg_ptr

  Pointer to an element of type T in an Ipc::Istream.
- class Istream

  Input stream for IPC unmarshalling.
- class Ostream

  Output stream for IPC marshalling.
- class Iostream

  Input/Output stream for IPC [un]marshalling.

Typedefs

- typedef unsigned short Array_len_default

  Default type for passing length of an array.
- typedef Gen_fpage< Snd_item > Snd_fpage

  Send flex-page.
- typedef Gen_fpage< Buf_item > Rcv_fpage

  Rcv flex-page.

Functions

- template<typename T >
  Cap< T > make_cap (L4::Cap< T > cap, unsigned rights) noexcept

  Make an L4::Ipc::Cap<T> for the given capability and rights.
- template<typename T >
  Cap< T > make_cap_rw (L4::Cap< T > cap) noexcept

  Make an L4::Ipc::Cap<T> for the given capability with L4_CAP_FPAGE_RW rights.
- template<typename T >
  Cap< T > make_cap_rws (L4::Cap< T > cap) noexcept

  Make an L4::Ipc::Cap<T> for the given capability with L4_CAP_FPAGE_RWS rights.
- template<typename T >
  Cap< T > make_cap_full (L4::Cap< T > cap) noexcept

  Make an L4::IPC::Cap<T> for the given capability with full fpage and object-specific rights.
- template<typename T >
  Internal::Buf_cp_out< T > buf_cp_out (T const *v, unsigned long size)

  Insert an array into an Ipc::Ostream.
- template<typename T >
  Internal::Buf_cp_in< T > buf_cp_in (T *v, unsigned long &size)

  Extract an array from an Ipc::Istream.
- template<typename T >
  Str_cp_in< T > str_cp_in (T *v, unsigned long &size)

  Create a Str_cp_in for the given values.
- template<typename T >
  Msg_ptr< T > msg_ptr (T *&p)

  Create an Msg_ptr to adjust the given pointer.
- template<typename T >
  Internal::Buf_in< T > buf_in (T *&v, unsigned long &size)

  Return a pointer to stream array data.
- template<typename T >
  T read (Istream &s)

  Read a value out of a stream.

14.4.1 Detailed Description

IPC related functionality.

14.4.2 Function Documentation

14.4.2.1 buf_cp_in()

```
template<typename T >
Internal::Buf_cp_in<T> L4::Ipc::buf_cp_in (
            T * v,
            unsigned long & size )
```

Extract an array from an Ipc::Istream.

Parameters

	v	Pointer to the array that shall receive the values from the Ipc::Istream.
in,out	*size*	Input: the number of elements the array can take at most Output: the number of elements found in the stream.

buf_cp_in() can be used to extract an array from an Ipc::Istream. This is the counterpart buf_cp_out(). The data from the received message is thereby copied to the given buffer and size is set to the number of elements found in the stream. To avoid the copy operation buf_in() may be used instead.

See also

buf_in() and buf_cp_out().

Definition at line 170 of file ipc_stream.

14.4.2.2 buf_cp_out()

```
template<typename T >
Internal::Buf_cp_out<T> L4::Ipc::buf_cp_out (
            T const * v,
            unsigned long size )
```

Insert an array into an Ipc::Ostream.

Parameters

v	Pointer to the array that shall be inserted into an Ipc::Ostream.
size	Number of elements in the array.

This function inserts an array (e.g. a string) into an Ipc::Ostream. The data is copied to the stream. On insertion into the Ipc::Ostream exactly the given number of elements of type T are copied to the message buffer, this means the source buffer is no longer referenced after insertion into the stream.

See also

The counterpart is either buf_cp_in() or buf_in().

Definition at line 111 of file ipc_stream.

14.4.2.3 buf_in()

```
template<typename T >
Internal::Buf_in<T> L4::Ipc::buf_in (
            T *& v,
            unsigned long & size )
```

Return a pointer to stream array data.

Parameters

out	*v*	Pointer to the array within the Ipc::Istream.
out	*size*	The number of elements found in the stream.

This routine provdes a possibility to extract an array from an Ipc::Istream, without extra copy overhead. In contrast to buf_cp_in() the data is not copied to a buffer, but a pointer to the array is returned. The user must make sure the UTCB is not used for other purposes while the returned pointer is still in use.

The mechanism is comparable to that of Msg_ptr, however it handles arrays inserted with buf_cp_out().

See also

buf_cp_in() and buf_cp_out().

Definition at line 321 of file ipc_stream.

14.4.2.4 make_cap()

```
template<typename T >
Cap<T> L4::Ipc::make_cap (
            L4::Cap< T > cap,
            unsigned rights )  [noexcept]
```

Make an L4::Ipc::Cap<T> for the given capability and rights.

Template Parameters

T	(IMPLICIT) type of the referenced interface

Parameters

cap	source capability (L4::Cap<T>)
rights	rights mask that shall be applied on transfer.

Definition at line 624 of file ipc_types.

14.4.2.5 make_cap_full()

```
template<typename T >
Cap<T> L4::Ipc::make_cap_full (
            L4::Cap< T > cap )  [noexcept]
```

Make an L4::IPC::Cap<T> for the given capability with full fpage and object-specific rights.

Template Parameters

T	(implicit) type of the referenced interface

Parameters

cap	source capability (L4::Cap<T>)

See also

L4_cap_fpage_rights

L4_obj_fpage_ctl

Note

Full rights (including object-specific rights) are required when mapping an IPC gate where the receiver should become the server, i.e. where the receiver wants to call L4::Ipc_gate::bind_thread().

Definition at line 662 of file ipc_types.

References L4_CAP_FPAGE_RWSD, and L4_FPAGE_C_OBJ_RIGHTS.

14.4.2.6 make_cap_rw()

```
template<typename T >
Cap<T> L4::Ipc::make_cap_rw (
            L4::Cap< T > cap )  [noexcept]
```

Make an L4::Ipc::Cap<T> for the given capability with L4_CAP_FPAGE_RW rights.

Template Parameters

T	(IMPLICIT) type of the referenced interface

Parameters

cap	source capability (L4::Cap<T>)

Examples

examples/libs/l4re/c++/mem_alloc/ma+rm.cc, examples/libs/l4re/c++/shared_ds/ds_clnt.cc, and examples/libs/l4re/c++/shared_

Definition at line 634 of file ipc_types.

References L4_CAP_FPAGE_RW.

Referenced by L4Re::Util::Event_buffer_t< PAYLOAD >::attach(), L4virtio::Driver::Device::driver_connect(), L4Re::Util::Event_t< PAYLOAD >::init(), L4Re::Util::Event_t< PAYLOAD >::init_poll(), L4virtio::Driver::Device::register_ds(), and L4virtio::Driver::Block_device::setup_device().

Here is the caller graph for this function:

14.4.2.7 make_cap_rws()

```
template<typename T >
Cap<T> L4::Ipc::make_cap_rws (
            L4::Cap< T > cap )  [noexcept]
```

Make an L4::Ipc::Cap<T> for the given capability with L4_CAP_FPAGE_RWS rights.

Template Parameters

T	(IMPLICIT) type of the referenced interface

Parameters

cap	source capability (L4::Cap<T>)

Definition at line 644 of file ipc_types.

References L4_CAP_FPAGE_RWS.

14.4.2.8 msg_ptr()

```
template<typename T >
Msg_ptr<T> L4::Ipc::msg_ptr (
            T *& p )
```

Create an Msg_ptr to adjust the given pointer.

This function makes it more convenient to extract pointers to data in the message buffer itself from an Ipc::Istream. This may be used to avoid copy out of large data structures. (See Msg_ptr.)

Definition at line 263 of file ipc_stream.

14.4.2.9 read()

```
template<typename T >
T L4::Ipc::read (
            Istream & s )  [inline]
```

Read a value out of a stream.

Parameters

s	An Istream.

Returns

The value of type `T`.

The stream position is progressed accordingly.

Definition at line 1294 of file ipc_stream.

14.4.2.10 str_cp_in()

```
template<typename T >
Str_cp_in<T> L4::Ipc::str_cp_in (
            T * v,
            unsigned long & size )
```

Create a Str_cp_in for the given values.

Parameters

	v	Pointer to the array that shall receive the values from the Ipc::Istream.
in,out	*size*	Input: the number of elements the array can take at most Output: the number of elements found in the stream.

This function makes it more convenient to extract arrays from an Ipc::Istream (

See also

Str_cp_in.)

Definition at line 224 of file ipc_stream.

14.5 L4::Ipc::Msg Namespace Reference

IPC Message related functionality.

Data Structures

- struct Elem< Array< A, LEN > >

  Array as input arguments.
- struct Elem< Array< A, LEN > & >

  Array as output argument.
- struct Elem< Array_ref< A, LEN > & >

  Array_ref as output argument.
- struct Dir_in

  Marker type for input values.
- struct Dir_out

  Marker type for output values.
- struct Cls_data

  Marker type for data values.
- struct Cls_item

  Marker type for item values.
- struct Cls_buffer

  Marker type for receive buffer values.
- struct Do_in_data

  Marker for Input data.
- struct Do_out_data

  Marker for Output data.
- struct Do_in_items

  Marker for Input items.
- struct Do_out_items

  Marker for Output items.
- struct Do_rcv_buffers

  Marker for receive buffers.
- struct Clnt_val_ops

  Defines client-side handling of 'MTYPE' as RPC argument.
- struct Svr_val_ops

  Defines server-side handling for `MTYPE` server arguments.
- struct Is_valid_rpc_type

  Type trait defining a valid RPC parameter type.
- struct Svr_arg_pack

  Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function.

Enumerations

- enum {
  Word_bytes = sizeof(l4_umword_t) , Item_words = 2 , Item_bytes = Word_bytes ∗ Item_words , Mr_words = L4_UTCB_GENERIC_DATA_SIZE ,
  Mr_bytes = Word_bytes ∗ Mr_words , Br_bytes = Word_bytes ∗ L4_UTCB_GENERIC_BUFFERS_SIZE }

Functions

- constexpr unsigned long align_to (unsigned long bytes, unsigned long align) noexcept

  Pad bytes to the given alignment align (in bytes)
- template<typename T >
  constexpr unsigned long align_to (unsigned long bytes) noexcept

  Pad bytes to the alignment of the type T.
- template<typename T >
  constexpr bool check_size (unsigned offset, unsigned limit) noexcept

  Check if there is enough space for T from offset to limit.
- template<typename T , typename CTYPE >
  bool check_size (unsigned offset, unsigned limit, CTYPE cnt) noexcept

  Check if there is enough space for an array of T from offset to limit.
- template<typename T >
  int msg_add (char ∗msg, unsigned offs, unsigned limit, T v) noexcept

  Add some data to a message at offs.
- template<typename T >
  int msg_get (char ∗msg, unsigned offs, unsigned limit, T &v) noexcept

  Get some data from a message at offs.

14.5.1 Detailed Description

IPC Message related functionality.

14.5.2 Enumeration Type Documentation

14.5.2.1 anonymous enum

`anonymous enum`

Enumerator

Word_bytes	number of bytes for one message word
Item_words	number of message words for one message item
Item_bytes	number of bytes for one message item
Mr_words	number of message words available in the UTCB
Mr_bytes	number of bytes available in the UTCB message registers
Br_bytes	number of bytes available in the UTCB buffer registers

Definition at line 96 of file ipc_basics.

14.5.3 Function Documentation

14.5.3.1 align_to() [1/2]

```
template<typename T >
constexpr unsigned long L4::Ipc::Msg::align_to (
            unsigned long bytes )  [constexpr], [noexcept]
```

Pad *bytes* to the alignment of the type *T*.

Template Parameters

T	The data type used for the alignment

Parameters

bytes	The value to add the padding to

Returns

bytes padded to achieve the alignment of *T*.

Definition at line 51 of file ipc_basics.

References align_to().

Here is the call graph for this function:

14.5.3.2 align_to() [2/2]

```
constexpr unsigned long L4::Ipc::Msg::align_to (
            unsigned long bytes,
            unsigned long align )  [constexpr], [noexcept]
```

Pad bytes to the given alignment *align* (in bytes)

Parameters

bytes	The input value in bytes
align	The alignment value in bytes

Returns

the result after padding *bytes* to *align*.

Definition at line 41 of file ipc_basics.

Referenced by align_to().

Here is the caller graph for this function:

14.5.3.3 check_size() [1/2]

```
template<typename T >
constexpr bool L4::Ipc::Msg::check_size (
            unsigned offset,
            unsigned limit )  [constexpr], [noexcept]
```

Check if there is enough space for T from offset to limit.

Template Parameters

T	The data type that shall be fitted at *offset*

Parameters

offset	The current offset in bytes (must already be padded if desired).
limit	The limit in bytes that must not be exceeded after adding the size of *T*.

Returns

true if the limit will not be exceeded, false else.

Definition at line 64 of file ipc_basics.

14.5.3.4 check_size() [2/2]

```
template<typename T , typename CTYPE >
bool L4::Ipc::Msg::check_size (
            unsigned offset,
            unsigned limit,
            CTYPE cnt )  [inline], [noexcept]
```

Check if there is enough space for an array of T from offset to limit.

Template Parameters

T	The data type that shall be fitted at *offset*
CTYPE	Type of the *cnt* parameter

Parameters

offset	The current offset in bytes (must already be padded if desired).
limit	The limit in bytes that must not be exceeded after adding *cnt* times the size of *T*.
cnt	The number of elements of type *T* that shall be put at *offset*.

Returns

true if the limit will not be exceeded, false else.

Definition at line 82 of file ipc_basics.

References L4_UNLIKELY.

14.5.3.5 msg_add()

```
template<typename T >
int L4::Ipc::Msg::msg_add (
            char * msg,
            unsigned offs,
            unsigned limit,
            T v )  [inline], [noexcept]
```

Add some data to a message at offs.

Template Parameters

T	The type of the data to add

Parameters

msg	pointer to the start of the message
offs	The current offset within the message, this shall be padded to the alignment of *T* if *v* is added.
limit	The size limit in bytes that offset must not exceed.
v	The value to add to the message

Returns

The new offset when successful, a negative value if the given limit will be exceeded.

Definition at line 125 of file ipc_basics.

References L4_EMSGTOOLONG, and L4_UNLIKELY.

14.5.3.6 msg_get()

```
template<typename T >
int L4::Ipc::Msg::msg_get (
            char * msg,
            unsigned offs,
            unsigned limit,
            T & v )  [inline], [noexcept]
```

Get some data from a message at offs.

Template Parameters

T	The type of the data to read

Parameters

msg	Pointer to the start of the message
offs	The current offset within the message, this shall be padded to the alignment of *T* if a *v* can be read.
limit	The size limit in bytes that offset must not exceed.
v	A reference to receive the value from the message

Returns

The new offset when successful, a negative value if the given limit will be exceeded.

Definition at line 146 of file ipc_basics.

References L4_EMSGTOOSHORT, and L4_UNLIKELY.

14.6 L4::Ipc_svr Namespace Reference

Helper classes for L4::Server instantiation.

Data Structures

- class Server_iface

  Interface for server-loop related functions.
- struct Ignore_errors

  Mix in for LOOP_HOOKS to ignore IPC errors.
- struct Default_timeout

  Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.
- struct Compound_reply

  Mix in for LOOP_HOOKS to always use compound reply and wait.
- struct Default_setup_wait

  Mix in for LOOP_HOOKS for setup_wait no op.
- struct Direct_dispatch

  Direct dispatch helper, for forwarding dispatch calls to a registry R.
- struct Direct_dispatch< R * >

  Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry R.
- struct Exc_dispatch

  Dispatch helper wrapping try {} catch {} around the dispatch call.
- class Br_manager_no_buffers

  Empty implementation of Server_iface.
- struct Default_loop_hooks

  Default LOOP_HOOKS.
- class Timeout

  Callback interface for Timeout_queue.
- class Timeout_queue

  Timeout queue to be used in l4re server loop.
- class Timeout_queue_hooks

  Loop hooks mixin for integrating a timeout queue into the server loop.

Enumerations

- enum Reply_mode { Reply_compound , Reply_separate }

  Reply mode for server loop.

14.6.1 Detailed Description

Helper classes for L4::Server instantiation.

14.7 L4::Typeid Namespace Reference

Definition of interface data-type helpers.

Data Structures

- struct P_dispatch

  Use for protocol based dispatch stage.
- struct Raw_ipc

  RPCs list for passing raw incoming IPC to the server object.
- struct Rpcs

  Standard list of RPCs of an interface.
- struct Rpcs_code

  List of RPCs of an interface using a special opcode type.
- struct Rpc_nocode

  List of RPCs of an interface using a single operation without an opcode.
- struct Rpcs_sys

  List of RPCs typically used for kernel interfaces.

14.7.1 Detailed Description

Definition of interface data-type helpers.

Note

These type helpers are intended for internal use, if you look for standard C++ type traits use the `<type_traits>` header for the standard C++ library or use `<l4/cxx/type_traits>`.

14.8 L4::Types Namespace Reference

L4 basic type helpers for C++.

Data Structures

- class Flags

  Template for defining typical Flags bitmaps.
- struct Int_for_size

  Metafunction to get an unsigned integral type for the given size.
- struct Int_for_type

  Metafunction to get an integral type of the same size as `T`.
- struct Flags_ops_t

  Mixin class to define a set of friend bitwise operators on `DT`.
- struct Flags_t

  Template type to define a flags type with bitwise operations.
- struct Bool

  Boolean meta type.
- struct False

  False meta value.
- struct True

  True meta value.
- struct Same

  Compare two data types for equality.

14.8.1 Detailed Description

L4 basic type helpers for C++.

14.9 L4Re Namespace Reference

L4Re C++ Interfaces.

Namespaces

- Util

  Documentation of the L4 Runtime Environment utility functionality in C++.
- Vfs

  Virtual file system for interfaces POSIX libc.

Data Structures

- class Cap_alloc

  Capability allocator interface.
- class Smart_cap_auto

  Helper for Unique_cap and Unique_del_cap.
- class Smart_count_cap

  Helper for Ref_cap and Ref_del_cap.
- class Console

  Console class.
- class Dataspace

  Interface for memory-like objects.
- class Debug_obj

  Debug interface.
- class Dma_space

  DMA Address Space.
- class Env

  C++ interface of the initial environment that is provided to an L4 task.
- class Event

  Event class.
- class Event_buffer_t

  Event buffer class.
- class Inhibitor

  Set of inhibitor locks, which inhibit specific actions when held.
- class Log

  Log interface class.
- class Mem_alloc

  Memory allocation interface.
- struct Mmio_space

  Interface for memory-like address space accessible via IPC.
- class Namespace

  Name-space interface.
- class Parent

  Parent interface.
- struct Random

  Low-bandwidth interface for random number generators.
- class Rm

  Region map.

Typedefs

- template<typename T >
  using Shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >

  Shared capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >

  Shared capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using Shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ > >

  Shared capability that implements automatic free and unmap+delete of the capability selector.
- template<typename T >
  using shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ > >

  Shared capability that implements automatic free and unmap+delete of the capability selector.
- template<typename T >
  using Unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >

  Unique capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >

  Unique capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using Unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >

  Unique capability that implements automatic free and unmap+delete of the capability selector.
- template<typename T >
  using unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >

  Unique capability that implements automatic free and unmap+delete of the capability selector.

Functions

- void throw_error (long err, char const *extra="")

  Generate C++ exception.
- long chksys (long err, char const *extra="", long ret=0)

  Generate C++ exception on error.
- long chksys (l4_msgtag_t const &t, char const *extra="", l4_utcb_t *utcb=l4_utcb(), long ret=0)

  Generate C++ exception on error.
- long chksys (l4_msgtag_t const &t, l4_utcb_t *utcb, char const *extra="")

  Generate C++ exception on error.
- template<typename T >
  T chkcap (T &&cap, char const *extra="", long err=-L4_ENOMEM)

  Check for valid capability or raise C++ exception.
- l4_msgtag_t chkipc (l4_msgtag_t tag, char const *extra="", l4_utcb_t *utcb=l4_utcb())

  Test a message tag for IPC errors.
- template<typename T >
  Shared_cap< T > make_shared_cap (L4Re::Cap_alloc *ca)

  Allocate a capability slot and wrap it in a Shared_cap.
- template<typename T >
  Shared_del_cap< T > make_shared_del_cap (L4Re::Cap_alloc *ca)

  Allocate a capability slot and wrap it in a Shared_del_cap.
- template<typename T >
  Unique_cap< T > make_unique_cap (L4Re::Cap_alloc *ca)

  Allocate a capability slot and wrap it in an Unique_cap.
- template<typename T >
  Unique_del_cap< T > make_unique_del_cap (L4Re::Cap_alloc *ca)

  Allocate a capability slot and wrap it in an Unique_del_cap.

14.9.1 Detailed Description

L4Re C++ Interfaces.

L4 Runtime Environment.

14.9.2 Typedef Documentation

14.9.2.1 Shared_cap

```
template<typename T >
using L4Re::Shared_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>
>
```

Shared capability that implements automatic free and unmap of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Note

This type is intended for users who implement a custom capability allocator; otherwise use L4Re::Util::Shared_cap.

Definition at line 44 of file shared_cap.

14.9.2.2 shared_cap

```
template<typename T >
using L4Re::shared_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>
>
```

Shared capability that implements automatic free and unmap of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Note

This type is intended for users who implement a custom capability allocator; otherwise use L4Re::Util::Shared_cap.

Definition at line 47 of file shared_cap.

14.9.2.3 Shared_del_cap

```
template<typename T >
using L4Re::Shared_del_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>
>
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to Shared_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Note

This type is intended for users who implement a custom capability allocator; otherwise use L4Re::Util::Shared_del_cap.

Definition at line 80 of file shared_cap.

14.9.2.4 shared_del_cap

```
template<typename T >
using L4Re::shared_del_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>
>
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to Shared_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Note

This type is intended for users who implement a custom capability allocator; otherwise use L4Re::Util::Shared_del_cap.

Definition at line 83 of file shared_cap.

14.9.2.5 Unique_cap

```
template<typename T >
using L4Re::Unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>
>
```

Unique capability that implements automatic free and unmap of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

The ownership of the capability is managed in the same way as unique_ptr.

Note

This type is intended for users who implement a custom capability allocator; otherwise use L4Re::Util::Unique_cap.

Definition at line 42 of file unique_cap.

14.9.2.6 unique_cap

```
template<typename T >
using L4Re::unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>
>
```

Unique capability that implements automatic free and unmap of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

The ownership of the capability is managed in the same way as unique_ptr.

Note

This type is intended for users who implement a custom capability allocator; otherwise use L4Re::Util::Unique_cap.

Definition at line 45 of file unique_cap.

14.9.2.7 Unique_del_cap

```
template<typename T >
using L4Re::Unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

The main difference to Unique_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Note

This type is intended for users who implement a custom capability allocator; otherwise use L4Re::Util::Unique_del_cap.

Definition at line 75 of file unique_cap.

14.9.2.8 unique_del_cap

```
template<typename T >
using L4Re::unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

The main difference to Unique_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Note

This type is intended for users who implement a custom capability allocator; otherwise use L4Re::Util::Unique_del_cap.

Definition at line 78 of file unique_cap.

14.9.3 Function Documentation

14.9.3.1 chkcap()

```
template<typename T >
T L4Re::chkcap (
            T && cap,
            char const * extra = "",
            long err = -L4_ENOMEM )  [inline]
```

Check for valid capability or raise C++ exception.

Template Parameters

T	Type of object to check, must be capability-like (L4::Cap, L4Re::Util::Unique_cap etc.)

Parameters

cap	Capability value to check.
extra	Optional text for exception.
err	Error value for exception or 0 if the capability value should be used.

This function checks whether the capability is valid. If the capability is invalid an C++ exception is generated, using err if err is not zero, otherwise the capability value is used. A valid capability will just be returned.

Definition at line 145 of file error_helper.

References L4_UNLIKELY, and throw_error().

Referenced by L4virtio::Svr::Dev_config::Dev_config(), L4virtio::Driver::Device::driver_connect(), and L4virtio::Driver::Block_device::se

Here is the call graph for this function:

Here is the caller graph for this function:

14.9.3.2 chkipc()

```
l4_msgtag_t L4Re::chkipc (
            l4_msgtag_t tag,
            char const * extra = "",
            l4_utcb_t * utcb = l4_utcb() )  [inline]
```

Test a message tag for IPC errors.

Parameters

tag	Message tag returned by the IPC.
extra	Exception message in case of error.
utcb	The UTCB used in the IPC operation.

Returns

On IPC error an exception is thrown, otherwise `tag` is returned.

Exceptions

L4::Runtime_exception	with the translated IPC error code

This function does not check the message tag's label value.

Note

This must be called on a message tag before the UTCB is changed.

Definition at line 176 of file error_helper.

References chksys(), l4_msgtag_t::has_error(), and L4_UNLIKELY.

Here is the call graph for this function:

14.9.3.3 chksys() [1/3]

```
long L4Re::chksys (
            l4_msgtag_t const & t,
            char const * extra = "",
            l4_utcb_t * utcb = l4_utcb(),
            long ret = 0 )  [inline]
```

Generate C++ exception on error.

Parameters

t	Message tag.
extra	Optional text for exception (default "")
utcb	Option UTCB
ret	Optional value for exception, default is error value (err)

This function throws an exception if the message tag contains an error or the label in the message tag is negative. Otherwise the label in the message tag is returned.

Definition at line 89 of file error_helper.

References l4_msgtag_t::has_error(), L4_UNLIKELY, and throw_error().

Here is the call graph for this function:

14.9.3.4 chksys() [2/3]

```
long L4Re::chksys (
            l4_msgtag_t const & t,
            l4_utcb_t * utcb,
            char const * extra = "" )  [inline]
```

Generate C++ exception on error.

Parameters

t	Message tag.
utcb	UTCB.
extra	Optional text for exception (default "")

This function throws an exception if the message tag contains an error or the label in the message tag is negative. Otherwise the label in the message tag is returned.

Definition at line 112 of file error_helper.

References chksys().

Here is the call graph for this function:

14.9.3.5 chksys() [3/3]

```
long L4Re::chksys (
            long err,
            char const * extra = "",
            long ret = 0 )  [inline]
```

Generate C++ exception on error.

Parameters

err	Error value, if negative exception will be thrown
extra	Optional text for exception (default "")
ret	Optional value for exception, default is error value (err)

This function throws an exception if the err is negative and otherwise returns err.

Definition at line 68 of file error_helper.

References L4_UNLIKELY, and throw_error().

Referenced by L4virtio::Svr::Driver_mem_list_t< DATA >::add(), chkipc(), chksys(), L4virtio::Svr::Dev_config::Dev_config(), L4virtio::Svr::Device_t< DATA >::device_notify_irq(), L4virtio::Driver::Device::driver_acknowledge(), L4virtio::Driver::Device::driver_c L4virtio::Svr::Driver_mem_region_t< DATA >::Driver_mem_region_t(), L4virtio::Driver::Block_device::process_used_queue(), L4virtio::Svr::Device_t< DATA >::register_driver_irq(), L4virtio::Svr::Device_t< DATA >::register_single_driver_irq(), L4virtio::Svr::Driver_mem_list_t< DATA >::remove(), and L4virtio::Driver::Block_device::setup_device().

Here is the call graph for this function:

Here is the caller graph for this function:

14.9.3.6 make_shared_cap()

```
template<typename T >
Shared_cap<T> L4Re::make_shared_cap (
            L4Re::Cap_alloc * ca )
```

Allocate a capability slot and wrap it in a Shared_cap.

Template Parameters

T	Type of the object the capability refers to.

Parameters

ca	Capability allocator to use.

Note

This function is intended for users who implement a custom capability allocator; otherwise use L4Re::Util::make_shared_cap<T>().

Definition at line 60 of file shared_cap.

References L4Re::Cap_alloc::alloc().

Here is the call graph for this function:

14.9.3.7 make_shared_del_cap()

```
template<typename T >
Shared_del_cap<T> L4Re::make_shared_del_cap (
            L4Re::Cap_alloc * ca )
```

Allocate a capability slot and wrap it in a Shared_del_cap.

Template Parameters

T	Type of the object the capability refers to.

Parameters

ca	Capability allocator to use.

Note

This function is intended for users who implement a custom capability allocator; otherwise use L4Re::Util::make_shared_del_cap<T>().

Definition at line 96 of file shared_cap.

References L4Re::Cap_alloc::alloc().

Here is the call graph for this function:

14.9.3.8 make_unique_cap()

```
template<typename T >
Unique_cap<T> L4Re::make_unique_cap (
            L4Re::Cap_alloc * ca )
```

Allocate a capability slot and wrap it in an Unique_cap.

Template Parameters

T	Type of the object the capability refers to.

Parameters

ca	Capability allocator to use.

Note

This function is intended for users who implement a custom capability allocator; otherwise use L4Re::Util::make_unique_cap<T>().

Definition at line 58 of file unique_cap.

References L4Re::Cap_alloc::alloc().

Here is the call graph for this function:

14.9.3.9 make_unique_del_cap()

```
template<typename T >
Unique_del_cap<T> L4Re::make_unique_del_cap (
            L4Re::Cap_alloc * ca )
```

Allocate a capability slot and wrap it in an Unique_del_cap.

Template Parameters

T	Type of the object the capability refers to.

Parameters

ca	Capability allocator to use.

Note

This function is intended for users who implement a custom capability allocator; otherwise use L4Re::Util::make_unique_del_cap<T>().

Definition at line 91 of file unique_cap.

References L4Re::Cap_alloc::alloc().

Here is the call graph for this function:

14.9.3.10 throw_error()

```
void L4Re::throw_error (
            long err,
            char const * extra = "" )  [inline]
```

Generate C++ exception.

Parameters

err	Error value
extra	Optional text for exception (default "")

This function throws an L4 exception. The exact exception type depends on the error value (err). This function does never return.

Definition at line 45 of file error_helper.

References L4_EEXIST, L4_ENOENT, L4_ENOMEM, and L4_ERANGE.

Referenced by chkcap(), and chksys().

Here is the caller graph for this function:

14.10 L4Re::Util Namespace Reference

Documentation of the L4 Runtime Environment utility functionality in C++.

Data Structures

- class Cap_alloc_base

  Capability allocator.
- class Br_manager

  Buffer-register (BR) manager for L4::Server.

- struct Br_manager_hooks

  Predefined server-loop hooks for a server loop using the Br_manager.
- struct Br_manager_timeout_hooks

  Predefined server-loop hooks for a server with using the Br_manager and a timeout queue.
- class Smart_cap_auto

  Helper for Unique_cap and Unique_del_cap.
- class Smart_count_cap

  Helper for Ref_cap and Ref_del_cap.
- struct Ref_cap

  Automatic capability that implements automatic free and unmap of the capability selector.
- struct Ref_del_cap

  Automatic capability that implements automatic free and unmap+delete of the capability selector.
- struct Counter

  Counter for Counting_cap_alloc with variable data width.
- class Counting_cap_alloc

  Internal reference-counting cap allocator.
- class Dataspace_svr

  Dataspace server class.
- class Event_t

  Convenience wrapper for getting access to an event object.
- class Event_buffer_t

  Event_buffer utility class.
- class Event_buffer_consumer_t

  An event buffer consumer.
- class Event_svr

  Convenience wrapper for implementing an event server.
- class Item_alloc_base

  Item allocator.
- class Object_registry

  A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.
- class Registry_server

  A server loop object which has a Object_registry included.
- class Vcon_svr

  Console server template class.

Typedefs

- template<typename T >
  using Shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >

  Shared capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >

  Shared capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using Shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ > >

  Shared capability that implements automatic free and unmap+delete of the capability selector.
- template<typename T >
  using shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ > >

  Shared capability that implements automatic free and unmap+delete of the capability selector.
- template<typename T >
  using Unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >

Unique capability that implements automatic free and unmap of the capability selector.

- template<typename T >
  using unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >

  Unique capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using Unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >

  Unique capability that implements automatic free and unmap+delete of the capability selector.
- template<typename T >
  using unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >

  Unique capability that implements automatic free and unmap+delete of the capability selector.

Functions

- template<typename T >
  Ref_cap< T >::Cap make_ref_cap ()

  Allocate a capability slot and wrap it in a Ref_cap.
- template<typename T >
  Ref_del_cap< T >::Cap make_ref_del_cap ()

  Allocate a capability slot and wrap it in a Ref_del_cap.
- int kumem_alloc (l4_addr_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) noexcept

  Allocate state area.
- template<typename T >
  Shared_cap< T > make_shared_cap ()

  Allocate a capability slot and wrap it in a Shared_cap.
- template<typename T >
  Shared_del_cap< T > make_shared_del_cap ()

  Allocate a capability slot and wrap it in a Shared_del_cap.
- template<typename T >
  Unique_cap< T > make_unique_cap ()

  Allocate a capability slot and wrap it in an Unique_cap.
- template<typename T >
  Unique_del_cap< T > make_unique_del_cap ()

  Allocate a capability slot and wrap it in an Unique_del_cap.

Variables

- _Cap_alloc & cap_alloc

  Capability allocator.

14.10.1 Detailed Description

Documentation of the L4 Runtime Environment utility functionality in C++.

14.10.2 Typedef Documentation

14.10.2.1 Shared_cap

```
template<typename T >
using L4Re::Util::Shared_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>
>
```

Shared capability that implements automatic free and unmap of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Usage:

```
L4Re::Util::Shared_cap<L4Re::Dataspace> global_ds_cap;

{
  L4Re::Util::Shared_cap<L4Re::Dataspace>
    ds_cap = make_shared_cap<L4Re::Dataspace>();
  // reference count for the allocated cap selector is now 1

  // use the dataspace cap
  L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

  global_ds_cap = ds_cap;
  // reference count is now 2
  ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 59 of file shared_cap.

14.10.2.2 shared_cap

```
template<typename T >
using L4Re::Util::shared_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>
>
```

Shared capability that implements automatic free and unmap of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Usage:

```
L4Re::Util::Shared_cap<L4Re::Dataspace> global_ds_cap;

{
  L4Re::Util::Shared_cap<L4Re::Dataspace>
    ds_cap = make_shared_cap<L4Re::Dataspace>();
  // reference count for the allocated cap selector is now 1

  // use the dataspace cap
  L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

  global_ds_cap = ds_cap;
```

```
  // reference count is now 2
  ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 62 of file shared_cap.

14.10.2.3 Shared_del_cap

```
template<typename T >
using L4Re::Util::Shared_del_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>
>
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to Shared_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
L4Re::Util::Shared_del_cap<L4Re::Dataspace> global_ds_cap;

{
  L4Re::Util::Shared_del_cap<L4Re::Dataspace>
    ds_cap = make_shared_del_cap<L4Re::Dataspace>();
  // reference count for the allocated cap selector is now 1

  // use the dataspace cap
  L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

  global_ds_cap = ds_cap;
  // reference count is now 2
  ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).
```

Definition at line 109 of file shared_cap.

14.10.2.4 shared_del_cap

```
template<typename T >
using L4Re::Util::shared_del_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>
>
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to Shared_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
L4Re::Util::Shared_del_cap<L4Re::Dataspace> global_ds_cap;

{
  L4Re::Util::Shared_del_cap<L4Re::Dataspace>
    ds_cap = make_shared_del_cap<L4Re::Dataspace>();
  // reference count for the allocated cap selector is now 1

  // use the dataspace cap
  L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

  global_ds_cap = ds_cap;
  // reference count is now 2
  ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).
```

Definition at line 112 of file shared_cap.

14.10.2.5 Unique_cap

```
template<typename T >
using L4Re::Util::Unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>
>
```

Unique capability that implements automatic free and unmap of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

The ownership of the capability is managed in the same way as unique_ptr.

Usage:

```
{
  L4Re::Util::Unique_cap<L4Re::Dataspace>
    ds_cap = L4Re::Util::make_unique_cap<L4Re::Dataspace>();

  // use the dataspace cap
  L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));
```

```
  ...

  // At the end of the scope ds_cap is unmapped and the capability
  // selector is freed.
}
```

Definition at line 54 of file unique_cap.

14.10.2.6 unique_cap

```
template<typename T >
using L4Re::Util::unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>
>
```

Unique capability that implements automatic free and unmap of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

The ownership of the capability is managed in the same way as unique_ptr.

Usage:

```
{
  L4Re::Util::Unique_cap<L4Re::Dataspace>
    ds_cap = L4Re::Util::make_unique_cap<L4Re::Dataspace>();

  // use the dataspace cap
  L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

  ...

  // At the end of the scope ds_cap is unmapped and the capability
  // selector is freed.
}
```

Definition at line 57 of file unique_cap.

14.10.2.7 Unique_del_cap

```
template<typename T >
using L4Re::Util::Unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

The main difference to Unique_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
{
  L4Re::Util::Unique_del_cap<L4Re::Dataspace>
    ds_cap = make_unique_del_cap<L4Re::Dataspace>());

  // use the dataspace cap
  L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

  ...

  // At the end of the scope ds_cap is unmapped and the capability
  // selector is freed. Because the deletion flag is set the data space
  // shall also be deleted (even if there are other references to this
  // data space).
}
```

Definition at line 97 of file unique_cap.

14.10.2.8 unique_del_cap

```
template<typename T >
using L4Re::Util::unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

T	Type of the object the capability refers to.

The main difference to Unique_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
{
  L4Re::Util::Unique_del_cap<L4Re::Dataspace>
    ds_cap = make_unique_del_cap<L4Re::Dataspace>());

  // use the dataspace cap
  L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

  ...

  // At the end of the scope ds_cap is unmapped and the capability
  // selector is freed. Because the deletion flag is set the data space
  // shall also be deleted (even if there are other references to this
  // data space).
}
```

Definition at line 100 of file unique_cap.

14.10.3 Function Documentation

14.10.3.1 make_shared_cap()

```
template<typename T >
Shared_cap<T> L4Re::Util::make_shared_cap ( )
```

Allocate a capability slot and wrap it in a Shared_cap.

Template Parameters

T	Type of the object the capability refers to.

Definition at line 71 of file shared_cap.

References L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc(), and cap_alloc.

Here is the call graph for this function:

14.10.3.2 make_shared_del_cap()

```
template<typename T >
Shared_del_cap<T> L4Re::Util::make_shared_del_cap ( )
```

Allocate a capability slot and wrap it in a Shared_del_cap.

Template Parameters

T	Type of the object the capability refers to.

Definition at line 121 of file shared_cap.

References L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc(), and cap_alloc.

Here is the call graph for this function:

14.10.3.3 make_unique_cap()

```
template<typename T >
Unique_cap<T> L4Re::Util::make_unique_cap ( )
```

Allocate a capability slot and wrap it in an Unique_cap.

Template Parameters

T	Type of the object the capability refers to.

Definition at line 66 of file unique_cap.

References L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc(), and cap_alloc.

Here is the call graph for this function:

14.10.3.4 make_unique_del_cap()

```
template<typename T >
Unique_del_cap<T> L4Re::Util::make_unique_del_cap ( )
```

Allocate a capability slot and wrap it in an Unique_del_cap.

Template Parameters

T	Type of the object the capability refers to.

Definition at line 109 of file unique_cap.

References L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc(), and cap_alloc.

Here is the call graph for this function:

14.11 L4Re::Vfs Namespace Reference

Virtual file system for interfaces POSIX libc.

Data Structures

- class Be_file

  Boiler plate class for implementing an open file for L4Re::Vfs.
- class Be_file_system

  Boilerplate class for implementing a L4Re::Vfs::File_system.
- class Generic_file

  The common interface for an open POSIX file.
- class Directory

  Interface for a POSIX file that is a directory.
- class Regular_file

  Interface for a POSIX file that provides regular file semantics.
- class Special_file

  Interface for a POSIX file that provides special file semantics.
- class File

  The basic interface for an open POSIX file.
- class Mman

  Interface for the POSIX memory management.
- class File_system

  Basic interface for an L4Re::Vfs file system.
- class Fs

  POSIX File-system related functionality.
- class Ops

  Interface for the POSIX backends for an application.

Functions

- L4Re::Vfs::Ops *vfs_ops asm ("l4re_env_posix_vfs_ops")

  Reference to the applications L4Re::Vfs::Ops singleton.

14.11.1 Detailed Description

Virtual file system for interfaces POSIX libc.

14.12 L4vbus Namespace Reference

C++ interface of the Vbus API.

Data Structures

- class Pm

  Power-management API mixin.
- class Device

  Device on a L4vbus::Vbus.
- class Icu

  Vbus Interrupt controller API.
- class Vbus

  The virtual bus (Vbus) interface.
- class Gpio_pin

  A GPIO pin.
- class Gpio_module

  A Gpio_module groups multiple GPIO pins together.
- class Pci_host_bridge

  A Pci host bridge.
- class Pci_dev

  A PCI device.

14.12.1 Detailed Description

C++ interface of the Vbus API.

The virtual bus (Vbus) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an Icu (Interrupt controller) for interrupt handling.

The Vbus interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

Refer to L4 Vbus functions for the C API.

Include File

```
#include <l4/vbus/vbus>
```

Include File

```
#include <l4/vbus/vbus_gpio>
```

Include File

```
#include <l4/vbus/vbus_pci>
```

14.13 L4virtio Namespace Reference

L4-VIRTIO Transport C++ API.

Data Structures

- class Device

  IPC interface for virtio over L4 IPC.
- class Ptr

  Pointer used in virtio descriptors.
- class Virtqueue

  Low-level Virtqueue.

14.13.1 Detailed Description

L4-VIRTIO Transport C++ API.

Chapter 15

Data Structure Documentation

15.1 cxx::Auto_ptr< T > Class Template Reference

Smart pointer with automatic deletion.

Collaboration diagram for cxx::Auto_ptr< T >:

Public Types

- typedef T Ref_type

  The referenced type.

Public Member Functions

- Auto_ptr (T *p=0) throw ()

  Construction by assignment of a normal pointer.
- Auto_ptr (Auto_ptr const &o) throw ()

  Copy construction, releases the original pointer.
- Auto_ptr & operator= (Auto_ptr const &o) throw ()

  Assignment from another smart pointer.
- ~Auto_ptr () throw ()

  Destruction, shall delete the object.
- T & operator* () const throw ()

  Dereference the pointer.
- T * operator-> () const throw ()

  Member access for the object.
- T * get () const throw ()

  Get the normal pointer.
- T * release () throw ()

  Release the object and get the normal pointer back.
- void reset (T *p=0) throw ()

  Delete the object and reset the smart pointer to NULL.
- operator Priv_type * () const throw ()

  Operator for `if (!ptr) ...`.

15.1.1 Detailed Description

template<typename T>
class cxx::Auto_ptr< T >

Smart pointer with automatic deletion.

Template Parameters

T	The type of the referenced object.

This smart pointer calls the delete operator when the destructor is called. This has the effect that the object the pointer points to will be deleted when the pointer goes out of scope, or a new value gets assigned. The smart pointer provides a release() method to get a normal pointer to the object and set the smart pointer to NULL.

Definition at line 36 of file auto_ptr.

15.1.2 Constructor & Destructor Documentation

15.1.2.1 Auto_ptr() [1/2]

```
template<typename T >
cxx::Auto_ptr< T >::Auto_ptr (
            T * p = 0 ) throw ( )   [inline], [explicit]
```

Construction by assignment of a normal pointer.

Parameters

p	The pointer to the object

Definition at line 51 of file auto_ptr.

15.1.2.2 Auto_ptr() [2/2]

```
template<typename T >
cxx::Auto_ptr< T >::Auto_ptr (
            Auto_ptr< T > const & o ) throw ( )   [inline]
```

Copy construction, releases the original pointer.

Parameters

o	The smart pointer, which shall be copied and released.

Definition at line 57 of file auto_ptr.

15.1.3 Member Function Documentation

15.1.3.1 get()

```
template<typename T >
T* cxx::Auto_ptr< T >::get ( ) const throw ( )   [inline]
```

Get the normal pointer.

Attention

This function will not release the object, the object will be deleted by the smart pointer.

Definition at line 90 of file auto_ptr.

15.1.3.2 operator=()

```
template<typename T >
Auto_ptr& cxx::Auto_ptr< T >::operator= (
            Auto_ptr< T > const & o ) throw ( )   [inline]
```

Assignment from another smart pointer.

Parameters

o	The source for the assignment (will be released).

Definition at line 65 of file auto_ptr.

References cxx::Auto_ptr< T >::release().

Here is the call graph for this function:

15.1.3.3 release()

```
template<typename T >
T* cxx::Auto_ptr< T >::release ( ) throw ( )   [inline]
```

Release the object and get the normal pointer back.

After calling this function the smart pointer will point to NULL and the object will not be deleted by the pointer anymore.

Definition at line 98 of file auto_ptr.

Referenced by cxx::Auto_ptr< T >::operator=().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/auto_ptr

15.2 cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > Class Template Reference

AVL tree based associative container.

Inheritance diagram for cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >:

Collaboration diagram for cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >:

Public Types

- typedef COMPARE< KEY_TYPE > Key_compare

  Type of the comparison functor.
- typedef KEY_TYPE Key_type

  Type of the key values.
- typedef DATA_TYPE Data_type

Type of the data values.

- typedef Base_type::Node Node

  Return type for find.

- typedef Base_type::Node_allocator Node_allocator

  Type of the allocator.

Public Member Functions

- Avl_map (Node_allocator const &alloc=Node_allocator())

  Create an empty AVL-tree based map.

- cxx::Pair< Iterator, int > insert (Key_type const &key, Data_type const &data)

  Insert a <key, data> pair into the map.

- Data_type const & operator[ ] (Key_type const &key) const

  Get the data for the given key.

- Data_type & operator[ ] (Key_type const &key)

  Get or insert data for the given key.

15.2.1 Detailed Description

template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_functor, template< typename B > class ALLOC = New_allocator>
class cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >

AVL tree based associative container.

Template Parameters

KEY_TYPE	Type of the key values.
DATA_TYPE	Type of the data values.
COMPARE	Type comparison functor for the key values.
ALLOC	Type of the allocator used for the nodes.

Definition at line 56 of file avl_map.

15.2.2 Constructor & Destructor Documentation

15.2.2.1 Avl_map()

```
template<typename KEY_TYPE , typename DATA_TYPE , template< typename A > class COMPARE = Lt↩
_functor, template< typename B > class ALLOC = New_allocator>
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::Avl_map (
            Node_allocator const & alloc = Node_allocator() )  [inline]
```

Create an empty AVL-tree based map.

Parameters

alloc	The node allocator.

Definition at line 91 of file avl_map.

15.2.3 Member Function Documentation

15.2.3.1 insert()

```
template<typename KEY_TYPE , typename DATA_TYPE , template< typename A > class COMPARE = Lt↩
_functor, template< typename B > class ALLOC = New_allocator>
cxx::Pair<Iterator, int> cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::insert (
            Key_type const & key,
            Data_type const & data )  [inline]
```

Insert a <key, data> pair into the map.

Parameters

key	The key value.
data	The data value to insert.

Returns

A pair of iterator (`first`) and return value (`second`). `second` will be 0 if the element was inserted into the set and `-#E_exist` if the key was already in the set and the set was therefore not updated. In both cases, `first` contains an iterator that points to the element. `second` may also be `-#E_nomem` when memory for the new node could not be allocated. `first` is then invalid.

Definition at line 110 of file avl_map.

References cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::insert().

Referenced by cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator[ ]().

Here is the call graph for this function:

Here is the caller graph for this function:

15.2.3.2 operator[]() [1/2]

```
template<typename KEY_TYPE , typename DATA_TYPE , template< typename A > class COMPARE = Lt↩
_functor, template< typename B > class ALLOC = New_allocator>
Data_type& cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator[] (
            Key_type const & key )  [inline]
```

Get or insert data for the given key.

Parameters

key	The key value to use for lookup.

Returns

If the item already exists, a reference to the data item. Otherwise a new data item is default-constructed and inserted under the given key before a reference is returned.

Definition at line 130 of file avl_map.

References cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, Lt_functor< KEY_TYPE >, New_allocator, Bits::Avl_map_ cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::insert(), and cxx::Pair< First, Second >::second.

Here is the call graph for this function:

15.2.3.3 operator[]() [2/2]

```
template<typename KEY_TYPE , typename DATA_TYPE , template< typename A > class COMPARE = Lt↩
_functor, template< typename B > class ALLOC = New_allocator>
Data_type const& cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator[] (
            Key_type const & key ) const  [inline]
```

Get the data for the given key.

Parameters

key	The key value to use for lookup.

Precondition

A <key, data> pair for the given key value must exist.

Definition at line 118 of file avl_map.

References cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, Lt_functor< KEY_TYPE >, New_allocator, Bits::Avl_map_ and cxx::Pair< First, Second >::second.

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/avl_map

15.3 cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > Class Template Reference

AVL set for simple compareable items.

Inheritance diagram for cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >:

Collaboration diagram for cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >:

Additional Inherited Members

15.3.1 Detailed Description

template<typename ITEM_TYPE, class COMPARE = Lt_functor<ITEM_TYPE>, template< typename A > class ALLOC = New↩
_allocator>
class cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >

AVL set for simple compareable items.

The AVL set can store any kind of items where a partial order is defined. The default relation is defined by the '<' operator.

Template Parameters

ITEM_TYPE	The type of the items to be stored in the set.
COMPARE	The relation to define the partial order, default is to use operator '<'.
ALLOC	The allocator to use for the nodes of the AVL set.

Definition at line 447 of file avl_set.

The documentation for this class was generated from the following file:

- l4/cxx/avl_set

15.4 cxx::Avl_tree< Node, Get_key, Compare > Class Template Reference

A generic AVL tree.

Inheritance diagram for cxx::Avl_tree< Node, Get_key, Compare >:

Collaboration diagram for cxx::Avl_tree< Node, Get_key, Compare >:

Public Types

- typedef Bst::Iterator Iterator

- typedef Bst::Const_iterator Const_iterator

  Constant forward iterator for the tree.
- typedef Bst::Rev_iterator Rev_iterator

  Backward iterator for the tree.
- typedef Bst::Const_rev_iterator Const_rev_iterator

  Constant backward iterator for the tree.

Public Member Functions

- Pair< Node *, bool > insert (Node *new_node)

  Insert a new node into this AVL tree.
- Node * remove (Key_param_type key)

  Remove the node with key from the tree.
- Node * erase (Key_param_type key)

  An alias for remove().
- Avl_tree ()=default

  Create an empty AVL tree.
- ~Avl_tree () noexcept

  Destroy the tree.

Additional Inherited Members

15.4.1 Detailed Description

template<typename Node, typename Get_key, typename Compare = Lt_functor<typename Get_key::Key_type>>
class cxx::Avl_tree< Node, Get_key, Compare >

A generic AVL tree.

Template Parameters

Node	The data type of the nodes (must inherit from Avl_tree_node).
Get_key	The meta function to get the key value from a node. The implementation uses `Get_key::key_of(ptr_to_node)`. The type of the key values must be defined in `Get_key::Key_type`.
Compare	Binary relation to establish a total order for the nodes of the tree. `Compare()(l, r)` must return true if the key *l* is smaller than the key *r*.

This implementation does not provide any memory management. It is the responsibility of the caller to allocate nodes before inserting them and to free them when they are removed or when the tree is destroyed. Conversely, the caller must also ensure that nodes are removed from the tree before they are destroyed.

Examples

tmpfs/lib/src/fs.cc.

Definition at line 107 of file avl_tree.

15.4.2 Member Typedef Documentation

15.4.2.1 Iterator

```
template<typename Node , typename Get_key , typename Compare = Lt_functor<typename Get_key::↩
Key_type>>
typedef Bst::Iterator cxx::Avl_tree< Node, Get_key, Compare >::Iterator
```

Forward iterator for the tree.

Definition at line 137 of file avl_tree.

15.4.3 Member Function Documentation

15.4.3.1 insert()

```
template<typename Node , typename Get_key , class Compare >
Pair< Node *, bool > cxx::Avl_tree< Node, Get_key, Compare >::insert (
            Node * new_node )
```

Insert a new node into this AVL tree.

Parameters

new_node	A pointer to the new node.

Returns

A pair, with *second* set to `true` and *first* pointing to *new_node*, on success. If there is already a node with the same key then *first* points to this node and *second* is 'false'.

Definition at line 227 of file avl_tree.

Referenced by cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::insert().

Here is the caller graph for this function:

15.4.3.2 remove()

```
template<typename Node , typename Get_key , class Compare >
Node * cxx::Avl_tree< Node, Get_key, Compare >::remove (
            Key_param_type key )  [inline]
```

Remove the node with *key* from the tree.

Parameters

key	The key to the node to remove.

Returns

The pointer to the removed node on success, or 0 if no node with the *key* exists.

Definition at line 289 of file avl_tree.

Referenced by cxx::Avl_tree< Node, Get_key, Compare >::erase(), and cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC,

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/avl_tree

15.5 cxx::Avl_tree_node Class Reference

Node of an AVL tree.

Inheritance diagram for cxx::Avl_tree_node:

Collaboration diagram for cxx::Avl_tree_node:

Protected Member Functions

- Avl_tree_node ()=default

  Create an uninitialized node, this is what you should do.

Additional Inherited Members

15.5.1 Detailed Description

Node of an AVL tree.

Examples

tmpfs/lib/src/fs.cc.

Definition at line 38 of file avl_tree.

The documentation for this class was generated from the following file:

- l4/cxx/avl_tree

15.6 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference

Basic slab allocator.

Inheritance diagram for cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >:

Collaboration diagram for cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >:

Data Structures

- struct Slab_i

  Type of a slab.

Public Types

- enum { object_size = Obj_size , slab_size = Slab_size , objects_per_slab = (Slab_size - sizeof(Slab_head)) / object_size , max_free_slabs = Max_free }
- typedef Alloc< Slab_i > Slab_alloc

  Type of the backend allocator.

Public Member Functions

- void * alloc () throw ()

  Allocate a new object.
- void free (void *_o) throw ()

  Free the given object (_o).
- unsigned total_objects () const throw ()

  Get the total number of objects managed by the slab allocator.
- unsigned free_objects () const throw ()

  Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.

15.6.1 Detailed Description

template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator> class cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >

Basic slab allocator.

Template Parameters

Obj_size	The size of the objects managed by the allocator (in bytes).
Slab_size	The size of a slab (in bytes).
Max_free	The maximum number of free slabs. When this limit is reached slabs are freed, provided that the backend allocator supports allocated memory to be freed.
Alloc	The backend allocator used to allocate slabs.

Definition at line 42 of file slab_alloc.

15.6.2 Member Enumeration Documentation

15.6.2.1 anonymous enum

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
anonymous enum
```

Enumerator

object_size	Size of an object.
slab_size	Size of a slab.
objects_per_slab	Objects per slab.
max_free_slabs	Maximum number of free slabs.

Definition at line 76 of file slab_alloc.

15.6.3 Member Function Documentation

15.6.3.1 alloc()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void* cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::alloc ( ) throw ( )   [inline]
```

Allocate a new object.

Returns

A pointer to the new object if the allocation succeeds, or 0 on failure to acquire memory from the backend allocator when the slab cache memory is already exhausted.

Note

The user is responsible for initializing the object.

Definition at line 218 of file slab_alloc.

References cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free(), cxx::H_list< T, POLICY >::push_front(), and cxx::H_list< T, POLICY >::remove().

Referenced by cxx::Slab< Type, Slab_size, Max_free, Alloc >::alloc(), and cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Al

Here is the call graph for this function:

Here is the caller graph for this function:

15.6.3.2 free()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free (
            void * _o ) throw ( )   [inline]
```

Free the given object (_o).

Precondition

The object must have been allocated with this allocator.

Definition at line 257 of file slab_alloc.

References cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::max_free_slabs, cxx::Base_slab< Obj_size, Slab_size, Max_f
cxx::H_list< T, POLICY >::push_front(), cxx::H_list< T, POLICY >::remove(), and cxx::Base_slab< Obj_size, Slab_size, Max_free, A

Referenced by cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::alloc(), and cxx::Base_slab_static< Obj_size, Slab_size, Ma

Here is the call graph for this function:

Here is the caller graph for this function:

15.6.3.3 free_objects()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free_objects ( ) const throw
( )   [inline]
```

Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.

Returns

The number of free objects in the slab allocator.

Definition at line 319 of file slab_alloc.

References cxx::Bits::Basic_list< POLICY >::begin(), cxx::Bits::Basic_list< POLICY >::end(), and cxx::Base_slab< Obj_size, Slab_

Referenced by cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::free_objects().

Here is the call graph for this function:

Here is the caller graph for this function:

15.6.3.4 total_objects()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::total_objects ( ) const throw
( )   [inline]
```

Get the total number of objects managed by the slab allocator.

Returns

The number of objects managed by the allocator (including the free objects).

Definition at line 310 of file slab_alloc.

References cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::objects_per_slab.

Referenced by cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::total_objects().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/slab_alloc

15.7 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i Struct Reference

Type of a slab.

Inherits cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_store, and cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_head.

Collaboration diagram for cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i:

15.7.1 Detailed Description

template< int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator> struct cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i

Type of a slab.

Definition at line 97 of file slab_alloc.

The documentation for this struct was generated from the following file:

- l4/cxx/slab_alloc

15.8 cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference

Merged slab allocator (allocators for objects of the same size are merged together).

Inheritance diagram for cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >:

Collaboration diagram for cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >:

Public Types

- enum { object_size = Obj_size , slab_size = Slab_size , objects_per_slab = _A::objects_per_slab , max_free_slabs = Max_free }

Public Member Functions

- void * alloc () throw ()

  Allocate an object.
- void free (void *p) throw ()

  Free the given object (p).
- unsigned total_objects () const throw ()

  Get the total number of objects managed by the slab allocator.
- unsigned free_objects () const throw ()

  Get the number of free objects in the slab allocator.

15.8.1 Detailed Description

template< int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator> class cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >

Merged slab allocator (allocators for objects of the same size are merged together).

Template Parameters

Obj_size	The size of an object managed by the slab allocator.
Slab_size	The size of a slab.
Max_free	The maximum number of free slabs.
Alloc	The allocator for the slabs.

This slab allocator class is useful for merging slab allocators with the same parameters (equal `Obj_size`, `Slab_size`, `Max_free`, and `Alloc` parameters) together and share the overhead for the slab caches among all equal-sized objects.

Definition at line 399 of file slab_alloc.

15.8.2 Member Enumeration Documentation

15.8.2.1 anonymous enum

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
anonymous enum
```

Enumerator

object_size	Size of an object.
slab_size	Size of a slab.
objects_per_slab	Number of objects per slab.
max_free_slabs	Maximum number of free slabs.

Definition at line 406 of file slab_alloc.

15.8.3 Member Function Documentation

15.8.3.1 alloc()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void* cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::alloc ( ) throw ( )   [inline]
```

Allocate an object.

Note

The user is responsible for initializing the object.

Definition at line 423 of file slab_alloc.

References cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::alloc().

Here is the call graph for this function:

15.8.3.2 free()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::free (
            void * p ) throw ( )   [inline]
```

Free the given object (p).

Parameters

p	The pointer to the object to free.

Precondition

p must be a pointer to an object allocated by this allocator.

Definition at line 431 of file slab_alloc.

References cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free().

Here is the call graph for this function:

15.8.3.3 free_objects()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::free_objects ( ) const
throw ( )   [inline]
```

Get the number of free objects in the slab allocator.

Returns

The number of free objects in all free and partially used slabs managed by this allocator.

Note

The value is the merged value for all equal parameterized Base_slab_static instances.

Definition at line 451 of file slab_alloc.

References cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free_objects().

Here is the call graph for this function:

15.8.3.4 total_objects()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::total_objects ( )
const throw ( )   [inline]
```

Get the total number of objects managed by the slab allocator.

Returns

The number of objects managed by the allocator (including the free objects).

Note

The value is the merged value for all equal parameterized Base_slab_static instances.

Definition at line 441 of file slab_alloc.

References cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::total_objects().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/slab_alloc

15.9 cxx::Bitfield< T, LSB, MSB > Class Template Reference

Definition for a member (part) of a bit field.

Collaboration diagram for cxx::Bitfield< T, LSB, MSB >:

Data Structures

- class Value

  Internal helper type.
- class Value_base

  Internal helper type.
- class Value_unshifted

  Internal helper type.

Public Types

- enum { Bits = MSB + 1 - LSB , Lsb = LSB , Msb = MSB }
- enum Masks : T { Low_mask = ((T)~0ULL) >> (sizeof(T)*8 - Bits) , Mask = Low_mask << Lsb }
- typedef Best_type< Bits >::Type Bits_type

  Type to hold at least Bits bits.
- typedef Best_type< Bits+Lsb >::Type Shift_type

  Type to hold at least Bits + Lsb bits.
- typedef Value< T & > Ref

  Reference type to access the bits inside a raw bit field.
- typedef Value< T const > Val

  Value type to access the bits inside a raw bit field.
- typedef Value_unshifted< T & > Ref_unshifted

  Reference type to access the bits inside a raw bit field (in place).
- typedef Value_unshifted< T const > Val_unshifted

  Value type to access the bits inside a raw bit field (in place).

Static Public Member Functions

- static Bits_type get (Shift_type val)

  Get the bits out of `val`.
- static T get_unshifted (Shift_type val)

  Get the bits in place out of `val`.
- static T set_dirty (T dest, Shift_type val)

  Set the bits corresponding to `val`.
- static T set_unshifted_dirty (T dest, Shift_type val)

  Set the bits corresponding to `val`.
- static T set (T dest, Bits_type val)

  Set the bits corresponding to `val`.
- static T set_unshifted (T dest, Shift_type val)

  Set the bits corresponding to `val`.
- static T val_dirty (Shift_type val)

  Get the shifted bits for `val`.
- static T val (Bits_type val)

  Get the shifted bits for `val`.
- static T val_unshifted (Shift_type val)

  Get the shifted bits for `val`.

15.9.1 Detailed Description

template<typename T, unsigned LSB, unsigned MSB>
class cxx::Bitfield< T, LSB, MSB >

Definition for a member (part) of a bit field.

Parameters

T	The underlying type of the bit field.
LSB	The least significant bit of our bits.
MSB	The most significant bit of our bits.

Definition at line 35 of file bitfield.

15.9.2 Member Typedef Documentation

15.9.2.1 Bits_type

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Best_type<Bits>::Type cxx::Bitfield< T, LSB, MSB >::Bits_type
```

Type to hold at least Bits bits.

This type can handle all values that can be stored in this part of the bit field.

Definition at line 82 of file bitfield.

15.9.2.2 Shift_type

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Best_type<Bits + Lsb>::Type cxx::Bitfield< T, LSB, MSB >::Shift_type
```

Type to hold at least Bits + Lsb bits.

This type can handle all values that can be stored in this part of the bit field when they are at the target location (Lsb bits shifted to the left).

Definition at line 90 of file bitfield.

15.9.3 Member Enumeration Documentation

15.9.3.1 anonymous enum

```
template<typename T , unsigned LSB, unsigned MSB>
anonymous enum
```

Enumerator

Bits	Number of bits.
Lsb	index of the LSB
Msb	index of the MSB

Definition at line 61 of file bitfield.

15.9.3.2 Masks

```
template<typename T , unsigned LSB, unsigned MSB>
enum cxx::Bitfield::Masks : T
```

Enumerator

Low_mask	Mask value to get Bits bits.
Mask	Mask value to the bits out of a `T`.

Definition at line 68 of file bitfield.

15.9.4 Member Function Documentation

15.9.4.1 get()

```
template<typename T , unsigned LSB, unsigned MSB>
static Bits_type cxx::Bitfield< T, LSB, MSB >::get (
            Shift_type val )  [inline], [static]
```

Get the bits out of `val`.

Parameters

val	The raw value of the whole bit field.

Returns

The bits form Lsb to Msb shifted to the right.

Definition at line 107 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Low_mask, cxx::Bitfield< T, LSB, MSB >::Lsb, and cxx::Bitfield< T, LSB, MSB >::val().

Here is the call graph for this function:

15.9.4.2 get_unshifted()

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::get_unshifted (
            Shift_type val )  [inline], [static]
```

Get the bits in place out of `val`.

Parameters

val	The raw value of the whole bit field.

Returns

The bits from Lsb to Msb (unshifted).

This means other bits are masked out, however the result is not shifted to the right.

Definition at line 120 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Mask, and cxx::Bitfield< T, LSB, MSB >::val().

Here is the call graph for this function:

15.9.4.3 set()

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set (
            T dest,
            Bits_type val )  [inline], [static]
```

Set the bits corresponding to `val`.

Parameters

dest	The current value of the whole bit field.
val	The value to set into the bits.

Returns

The new value of the whole bit field.

Definition at line 169 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Low_mask, cxx::Bitfield< T, LSB, MSB >::set_dirty(), and cxx::Bitfield< T, LSB, MSB >::v

Here is the call graph for this function:

15.9.4.4 set_dirty()

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set_dirty (
            T dest,
            Shift_type val )  [inline], [static]
```

Set the bits corresponding to `val`.

Parameters

dest	The current value of the whole bit field.
val	The value to set into the bits.

Returns

The new value of the whole bit field.

Precondition

`val` must not contain more than Bits bits.

Note

This function does not mask `val` to the right number of bits.

Definition at line 135 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Lsb, cxx::Bitfield< T, LSB, MSB >::Mask, and cxx::Bitfield< T, LSB, MSB >::val().

Referenced by cxx::Bitfield< T, LSB, MSB >::set().

Here is the call graph for this function:

Here is the caller graph for this function:

15.9.4.5 set_unshifted()

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set_unshifted (
            T dest,
            Shift_type val )  [inline], [static]
```

Set the bits corresponding to `val`.

Parameters

dest	The current value of the whole bit field.
val	The value shifted Lsb bits to the left that shall be set into the bit field.

Returns

the new value of the whole bit field.

Definition at line 181 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Mask, cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty(), and cxx::Bitfield< T, LSB, MSB >::val().

Here is the call graph for this function:

15.9.4.6 set_unshifted_dirty()

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty (
            T dest,
            Shift_type val )  [inline], [static]
```

Set the bits corresponding to `val`.

Parameters

dest	The current value of the whole bit field.
val	The value shifted Lsb bits to the left that shall be set into the bits.

Returns

The new value of the whole bit field.

Precondition

`val` must not contain more than Bits bits shifted Lsb bits to the left.

Note

This function does not mask `val` to the right number of bits.

Definition at line 155 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Mask, and cxx::Bitfield< T, LSB, MSB >::val().

Referenced by cxx::Bitfield< T, LSB, MSB >::set_unshifted().

Here is the call graph for this function:

Here is the caller graph for this function:

15.9.4.7 val()

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::val (
            Bits_type val )  [inline], [static]
```

Get the shifted bits for `val`.

Parameters

val	The value to set into the bits.

Returns

The raw bit field value.

Definition at line 204 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Low_mask, and cxx::Bitfield< T, LSB, MSB >::val_dirty().

Referenced by cxx::Bitfield< T, LSB, MSB >::get(), cxx::Bitfield< T, LSB, MSB >::get_unshifted(), cxx::Bitfield< T, LSB, MSB >::set(cxx::Bitfield< T, LSB, MSB >::set_dirty(), cxx::Bitfield< T, LSB, MSB >::set_unshifted(), cxx::Bitfield< T, LSB, MSB >::set_unshiftec cxx::Bitfield< T, LSB, MSB >::val_dirty(), and cxx::Bitfield< T, LSB, MSB >::val_unshifted().

Here is the call graph for this function:

Here is the caller graph for this function:

15.9.4.8 val_dirty()

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::val_dirty (
            Shift_type val )  [inline], [static]
```

Get the shifted bits for `val`.

Parameters

val	The value to set into the bits.

Returns

The raw bit field value.

Precondition

`val` must not contain more than Bits bits.

Note

This function does not mask `val` to the right number of bits.

Definition at line 195 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Lsb, and cxx::Bitfield< T, LSB, MSB >::val().

Referenced by cxx::Bitfield< T, LSB, MSB >::val().

Here is the call graph for this function:

Here is the caller graph for this function:

15.9.4.9 val_unshifted()

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::val_unshifted (
            Shift_type val )  [inline], [static]
```

Get the shifted bits for `val`.

Parameters

val	The value shifted Lsb bits to the left that shall be set into the bits.

Returns

The raw bit field value.

Definition at line 214 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Mask, and cxx::Bitfield< T, LSB, MSB >::val().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/bitfield

15.10 cxx::Bitfield< T, LSB, MSB >::Value< TT > Class Template Reference

Internal helper type.

Inheritance diagram for cxx::Bitfield< T, LSB, MSB >::Value< TT >:

Collaboration diagram for cxx::Bitfield< T, LSB, MSB >::Value< TT >:

15.10.1 Detailed Description

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value< TT >

Internal helper type.

Definition at line 236 of file bitfield.

The documentation for this class was generated from the following file:

- l4/cxx/bitfield

15.11 cxx::Bitfield< T, LSB, MSB >::Value_base< TT > Class Template Reference

Internal helper type.

Inheritance diagram for cxx::Bitfield< T, LSB, MSB >::Value_base< TT >:

Collaboration diagram for cxx::Bitfield< T, LSB, MSB >::Value_base< TT >:

15.11.1 Detailed Description

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value_base< TT >

Internal helper type.

Definition at line 218 of file bitfield.

The documentation for this class was generated from the following file:

- l4/cxx/bitfield

15.12 cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Class Template Reference

Internal helper type.

Inheritance diagram for cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >:

Collaboration diagram for cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >:

15.12.1 Detailed Description

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >

Internal helper type.

Definition at line 247 of file bitfield.

The documentation for this class was generated from the following file:

- l4/cxx/bitfield

15.13 cxx::Bitmap< BITS > Class Template Reference

A static bit map.

Inheritance diagram for cxx::Bitmap< BITS >:

Collaboration diagram for cxx::Bitmap< BITS >:

Public Member Functions

- Bitmap () throw ()

  Create a bitmap with BITS bits.
- long scan_zero (long start_bit=0) const throw ()

  Scan for the first zero bit.

Additional Inherited Members

15.13.1 Detailed Description

template<int BITS>
class cxx::Bitmap< BITS >

A static bit map.

Parameters

BITS	the number of bits that shall be in the bitmap.

Definition at line 180 of file bitmap.

15.13.2 Member Function Documentation

15.13.2.1 scan_zero()

```
template<int BITS>
long cxx::Bitmap< BITS >::scan_zero (
            long start_bit = 0 ) const throw ( )   [inline]
```

Scan for the first zero bit.

Parameters

start_bit	Hint at the number of the first bit to look at. Zero bits below `start_bit` may or may not be taken into account by the implementation.

Return values

>=	0 Number of first zero bit found.
-1	All bits at `start_bit` or higher are set.

Compared to Bitmap_base::scan_zero(), the upper bound is set to BITS.

Definition at line 285 of file bitmap.

References cxx::Bitmap_base::scan_zero().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/bitmap

15.14 cxx::Bitmap_base Class Reference

Basic bitmap abstraction.

Inheritance diagram for cxx::Bitmap_base:

Collaboration diagram for cxx::Bitmap_base:

Data Structures

- class Bit

  A writeable bit in a bitmap.
- class Char

  Helper abstraction for a byte contained in the bitmap.
- class Word

  Helper abstraction for a word contained in the bitmap.

Public Member Functions

- void bit (long bit, bool on) throw ()

  Set the value of bit bit to on.
- void clear_bit (long bit) throw ()

  Clear bit bit.
- void set_bit (long bit) throw ()

  Set bit bit.
- word_type bit (long bit) const throw ()

  Get the truth value of a bit.
- word_type operator[ ] (long bit) const throw ()

  Get the bit at index bit.
- Bit operator[ ] (long bit) throw ()

  Get the lvalue for the bit at index bit.
- long scan_zero (long max_bit, long start_bit=0) const throw ()

  Scan for the first zero bit.

Static Public Member Functions

- static long words (long bits) throw ()

  Get the number of Words that are used for the bitmap.
- static long chars (long bits) throw ()

  Get the number of chars that are used for the bitmap.

Protected Types

- enum { W_bits = sizeof(word_type) ∗ 8 , C_bits = 8 }
- typedef unsigned long word_type

  Data type for each element of the bit buffer.

Static Protected Member Functions

- static unsigned word_index (unsigned bit)

  Get the word index for the given bit.
- static unsigned bit_index (unsigned bit)

  Get the bit index within word_type for the given bit.

Protected Attributes

- word_type ∗ _bits

  Pointer to the buffer storing the bits.

15.14.1 Detailed Description

Basic bitmap abstraction.

This abstraction keeps a pointer to a memory area that is used as bitmap.

Definition at line 30 of file bitmap.

15.14.2 Member Enumeration Documentation

15.14.2.1 anonymous enum

```
anonymous enum [protected]
```

Enumerator

W_bits	number of bits in word_type
C_bits	number of bits in char

Definition at line 38 of file bitmap.

15.14.3 Member Function Documentation

15.14.3.1 bit() [1/2]

```
Bitmap_base::word_type cxx::Bitmap_base::bit (
            long bit ) const throw ( )   [inline]
```

Get the truth value of a bit.

Parameters

bit	the number of the bit to read.

Returns

0 if *bit* is not set, != 0 if *bit* is set.

Definition at line 238 of file bitmap.

15.14.3.2 bit() [2/2]

```
void cxx::Bitmap_base::bit (
            long bit,
            bool on ) throw ( )   [inline]
```

Set the value of bit *bit* to *on*.

Parameters

bit	the number of the bit
on	the boolean value that shall be assigned to the bit.

Definition at line 211 of file bitmap.

Referenced by bit_index(), operator[ ]() , and word_index().

Here is the caller graph for this function:

15.14.3.3 bit_index()

```
static unsigned cxx::Bitmap_base::bit_index (
            unsigned bit )  [inline], [static], [protected]
```

Get the bit index within word_type for the given bit.

Parameters

bit	the bit index in the bitmap.

Returns

the bit index within word_type (bit % W_bits).

Definition at line 61 of file bitmap.

References bit(), and W_bits.

Here is the call graph for this function:

15.14.3.4 clear_bit()

```
void cxx::Bitmap_base::clear_bit (
            long bit ) throw ( )   [inline]
```

Clear bit *bit*.

Parameters

bit	the number of the bit to clear.

Definition at line 220 of file bitmap.

15.14.3.5 operator[]() [1/2]

```
Bit cxx::Bitmap_base::operator[] (
            long bit ) throw ( )   [inline]
```

Get the lvalue for the bit at index *bit*.

Parameters

bit	the number.

Returns

lvalue for *bit*

Definition at line 151 of file bitmap.

References bit().

Here is the call graph for this function:

15.14.3.6 operator[]() [2/2]

```
word_type cxx::Bitmap_base::operator[] (
            long bit ) const throw ( )   [inline]
```

Get the bit at index *bit*.

Parameters

bit	the number of the bit to read.

Returns

0 if *bit* is not set, != 0 if *bit* is set.

Definition at line 143 of file bitmap.

References bit().

Here is the call graph for this function:

15.14.3.7 scan_zero()

```
long cxx::Bitmap_base::scan_zero (
            long max_bit,
            long start_bit = 0 ) const throw ( )   [inline]
```

Scan for the first zero bit.

Parameters

max_bit	Upper bound (exclusive) for the scanning operation.
start_bit	Hint at the number of the first bit to look at. Zero bits below `start_bit` may or may not be taken into account by the implementation.

Return values

>=	0 Number of first zero bit found.
-1	All bits between `start_bit` and `max_bit` are set.

Definition at line 259 of file bitmap.

Referenced by cxx::Bitmap< BITS >::scan_zero().

Here is the caller graph for this function:

15.14.3.8 set_bit()

```
void cxx::Bitmap_base::set_bit (
            long bit ) throw ( )   [inline]
```

Set bit *bit*.

Parameters

bit	the number of the bit to set,

Definition at line 229 of file bitmap.

15.14.3.9 word_index()

```
static unsigned cxx::Bitmap_base::word_index (
            unsigned bit )  [inline], [static], [protected]
```

Get the word index for the given bit.

Parameters

bit	the index of the bit in question.

Returns

the index in Bitmap_base::_bits for the given bit (bit / W_bits).

Definition at line 54 of file bitmap.

References bit(), and W_bits.

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/bitmap

15.15 cxx::Bitmap_base::Bit Class Reference

A writeable bit in a bitmap.

Collaboration diagram for cxx::Bitmap_base::Bit:

15.15.1 Detailed Description

A writeable bit in a bitmap.

Definition at line 66 of file bitmap.

The documentation for this class was generated from the following file:

- l4/cxx/bitmap

15.16 cxx::Bitmap_base::Char< BITS > Class Template Reference

Helper abstraction for a byte contained in the bitmap.

Collaboration diagram for cxx::Bitmap_base::Char< BITS >:

15.16.1 Detailed Description

template< long BITS >
class cxx::Bitmap_base::Char< BITS >

Helper abstraction for a byte contained in the bitmap.

Definition at line 103 of file bitmap.

The documentation for this class was generated from the following file:

- l4/cxx/bitmap

15.17 cxx::Bitmap_base::Word< BITS > Class Template Reference

Helper abstraction for a word contained in the bitmap.

Inheritance diagram for cxx::Bitmap_base::Word< BITS >:

Collaboration diagram for cxx::Bitmap_base::Word< BITS >:

15.17.1 Detailed Description

template<long BITS>
class cxx::Bitmap_base::Word< BITS >

Helper abstraction for a word contained in the bitmap.

Definition at line 87 of file bitmap.

The documentation for this class was generated from the following file:

- l4/cxx/bitmap

15.18 cxx::Bits::Avl_map_get_key< KEY_TYPE > Struct Template Reference

Key-getter for Avl_map.

Collaboration diagram for cxx::Bits::Avl_map_get_key< KEY_TYPE >:

15.18.1 Detailed Description

template<typename KEY_TYPE>
struct cxx::Bits::Avl_map_get_key< KEY_TYPE >

Key-getter for Avl_map.

Definition at line 36 of file avl_map.

The documentation for this struct was generated from the following file:

- l4/cxx/avl_map

15.19 cxx::Bits::Avl_set_get_key< KEY_TYPE > Struct Template Reference

Internal, key-getter for Avl_set nodes.

Collaboration diagram for cxx::Bits::Avl_set_get_key< KEY_TYPE >:

15.19.1 Detailed Description

template<typename KEY_TYPE>
struct cxx::Bits::Avl_set_get_key< KEY_TYPE >

Internal, key-getter for Avl_set nodes.

Definition at line 107 of file avl_set.

The documentation for this struct was generated from the following file:

- l4/cxx/avl_set

15.20 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > Class Template Reference

Internal: AVL set with internally managed nodes.

Inheritance diagram for cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >:

Collaboration diagram for cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >:

Data Structures

- class Node

  A smart pointer to a tree item.

Public Types

- enum { E_noent = 2 , E_exist = 17 , E_nomem = 12 , E_inval = 22 }

  Return status constants.
- typedef ITEM_TYPE Item_type

  Type for the items store in the set.
- typedef GET_KEY Get_key

  Key-getter type to derive the sort key of an internal node.
- typedef GET_KEY::Key_type Key_type

  Type of the sort key used for the items.
- typedef Type_traits< Item_type >::Const_type Const_item_type

  Type used for const items within the set.
- typedef COMPARE Item_compare

  Type for the comparison functor.
- typedef ALLOC< _Node > Node_allocator

  Type for the node allocator.
- typedef Avl_set_iter< _Node, Item_type, Fwd > Iterator

  Forward iterator for the set.
- typedef Avl_set_iter< _Node, Const_item_type, Fwd > Const_iterator

  Constant forward iterator for the set.
- typedef Avl_set_iter< _Node, Item_type, Rev > Rev_iterator

  Backward iterator for the set.
- typedef Avl_set_iter< _Node, Const_item_type, Rev > Const_rev_iterator

  Constant backward iterator for the set.

Public Member Functions

- Base_avl_set (Node_allocator const &alloc=Node_allocator())

  Create a AVL-tree based set.
- Base_avl_set (Base_avl_set const &o)

  Create a copy of an AVL-tree based set.
- cxx::Pair< Iterator, int > insert (Item_type const &item)

  Insert an item into the set.
- int remove (Key_type const &item)

  Remove an item from the set.
- int erase (Key_type const &item)

  Erase the item with the given key.
- Node find_node (Key_type const &item) const

  Lookup a node equal to `item`.
- Node lower_bound_node (Key_type const &key) const

  Find the first node greater or equal to `key`.
- Const_iterator begin () const

  Get the constant forward iterator for the first element in the set.
- Const_iterator end () const

  Get the end marker for the constant forward iterator.
- Iterator begin ()

  Get the mutable forward iterator for the first element of the set.
- Iterator end ()

  Get the end marker for the mutable forward iterator.
- Const_rev_iterator rbegin () const

  Get the constant backward iterator for the last element in the set.
- Const_rev_iterator rend () const

  Get the end marker for the constant backward iterator.
- Rev_iterator rbegin ()

  Get the mutable backward iterator for the last element of the set.
- Rev_iterator rend ()

  Get the end marker for the mutable backward iterator.

15.20.1 Detailed Description

template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GET_KEY>
class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >

Internal: AVL set with internally managed nodes.

Use Avl_set, Avl_map, or Avl_tree in applications.

Template Parameters

ITEM_TYPE	The type of the items to be stored in the set.
COMPARE	The relation to define the partial order, default is to use operator '<'.
ALLOC	The allocator to use for the nodes of the AVL set.
GET_KEY	Sort-key getter (must provide the `Key_type` and sort-key for an item (of `ITEM_TYPE`).

Definition at line 131 of file avl_set.

15.20.2 Member Enumeration Documentation

15.20.2.1 anonymous enum

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
anonymous enum
```

Return status constants.

These constants are compatible with the L4 error codes, see l4_error_code_t.

Enumerator

E_noent	Item does not exist.
E_exist	Item exists already.
E_nomem	Memory allocation failed.
E_inval	Internal error.

Definition at line 140 of file avl_set.

15.20.3 Constructor & Destructor Documentation

15.20.3.1 Base_avl_set() [1/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Base_avl_set (
            Node_allocator const & alloc = Node_allocator() )  [inline], [explicit]
```

Create a AVL-tree based set.

Parameters

alloc	Node allocator.

Create an empty set (AVL-tree based).

Definition at line 246 of file avl_set.

15.20.3.2 Base_avl_set() [2/2]

```
template<typename Item , class Compare , template< typename A > class Alloc, typename KEY_↩
TYPE >
cxx::Bits::Base_avl_set< Item, Compare, Alloc, KEY_TYPE >::Base_avl_set (
            Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > const & o )  [inline]
```

Create a copy of an AVL-tree based set.

Parameters

o	The set to copy.

Creates a deep copy of the set with all its items.

Definition at line 405 of file avl_set.

References cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin(), cxx::Bits::Base_avl_set< ITEM_TYPE and cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::insert().

Here is the call graph for this function:

15.20.4 Member Function Documentation

15.20.4.1 begin() [1/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin ( )  [inline]
```

Get the mutable forward iterator for the first element of the set.

Returns

The mutable forward iterator for the first element of the set.

Definition at line 359 of file avl_set.

References cxx::Bits::Bst< Node, Get_key, Compare >::begin().

Here is the call graph for this function:

15.20.4.2 begin() [2/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Const_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin ( ) const
[inline]
```

Get the constant forward iterator for the first element in the set.

Returns

Constant forward iterator for the first element in the set.

Definition at line 348 of file avl_set.

References cxx::Bits::Bst< Node, Get_key, Compare >::begin().

Referenced by cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Base_avl_set().

Here is the call graph for this function:

Here is the caller graph for this function:

15.20.4.3 end() [1/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::end ( )  [inline]
```

Get the end marker for the mutable forward iterator.

Returns

The end marker for mutable forward iterator.

Definition at line 364 of file avl_set.

References cxx::Bits::Bst< Node, Get_key, Compare >::end().

Here is the call graph for this function:

15.20.4.4 end() [2/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Const_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::end ( ) const
[inline]
```

Get the end marker for the constant forward iterator.

Returns

The end marker for the constant forward iterator.

Definition at line 353 of file avl_set.

References cxx::Bits::Bst< Node, Get_key, Compare >::end().

Referenced by cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Base_avl_set().

Here is the call graph for this function:

Here is the caller graph for this function:

15.20.4.5 erase()

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
int cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::erase (
            Key_type const & item )  [inline]
```

Erase the item with the given key.

Parameters

item	The key of the item to remove.

Definition at line 316 of file avl_set.

References cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::remove().

Here is the call graph for this function:

15.20.4.6 find_node()

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Node cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::find_node (
            Key_type const & item ) const  [inline]
```

Lookup a node equal to `item`.

Parameters

item	The value to search for.

Returns

A smart pointer to the element found. If no element was found the smart pointer will be invalid.

Definition at line 327 of file avl_set.

References cxx::Bits::Bst< Node, Get_key, Compare >::find_node().

Here is the call graph for this function:

15.20.4.7 insert()

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Pair< typename Base_avl_set< Item, Compare, Alloc, KEY_TYPE >::Iterator, int > cxx::Bits::Base_avl_set<
Item, Compare, Alloc, KEY_TYPE >::insert (
            Item_type const & item )
```

Insert an item into the set.

Parameters

item	The item to insert.

Returns

A pair of iterator (`first`) and return value (`second`). `second` will be 0 if the element was inserted into the set and `-#E_exist` if the element was already in the set and the set was therefore not updated. In both cases, `first` contains an iterator that points to the element. `second` may also be `-#E_nomem` when memory for the node could not be allocated. `first` is then invalid.

Insert a new item into the set, each item can only be once in the set.

Definition at line 415 of file avl_set.

References cxx::Pair< First, Second >::first, cxx::Avl_tree< Node, Get_key, Compare >::insert(), and cxx::Pair< First, Second >::se

Referenced by cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Base_avl_set(), and cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::insert().

Here is the call graph for this function:

Here is the caller graph for this function:

15.20.4.8 lower_bound_node()

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Node cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::lower_bound_node (
            Key_type const & key ) const  [inline]
```

Find the first node greater or equal to `key`.

Parameters

key	Minimum key to look for.

Returns

Smart pointer to the first node greater or equal to `key`. Will be invalid if no such element was found.

Definition at line 338 of file avl_set.

References cxx::Bits::Bst< Node, Get_key, Compare >::lower_bound_node().

Here is the call graph for this function:

15.20.4.9 rbegin() [1/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rbegin ( )  [inline]
```

Get the mutable backward iterator for the last element of the set.

Returns

The mutable backward iterator for the last element of the set.

Definition at line 381 of file avl_set.

References cxx::Bits::Bst< Node, Get_key, Compare >::rbegin().

Here is the call graph for this function:

15.20.4.10 rbegin() [2/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Const_rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rbegin ( )
const  [inline]
```

Get the constant backward iterator for the last element in the set.

Returns

The constant backward iterator for the last element in the set.

Definition at line 370 of file avl_set.

References cxx::Bits::Bst< Node, Get_key, Compare >::rbegin().

Here is the call graph for this function:

15.20.4.11 remove()

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
int cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::remove (
            Key_type const & item )  [inline]
```

Remove an item from the set.

Parameters

item	The item to remove.

Return values

0	Success
-E_noent	Item does not exist
-E_inval	Internal error

Definition at line 296 of file avl_set.

References cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::E_inval, cxx::Bits::Base_avl_set< ITEM_TYP and cxx::Avl_tree< Node, Get_key, Compare >::remove().

Referenced by cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::erase().

Here is the call graph for this function:

Here is the caller graph for this function:

15.20.4.12 rend() [1/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rend ( ) [inline]
```

Get the end marker for the mutable backward iterator.

Returns

The end marker for mutable backward iterator.

Definition at line 386 of file avl_set.

References cxx::Bits::Bst< Node, Get_key, Compare >::rend().

Here is the call graph for this function:

15.20.4.13 rend() [2/2]

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Const_rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rend ( )
const [inline]
```

Get the end marker for the constant backward iterator.

Returns

The end marker for the constant backward iterator.

Definition at line 375 of file avl_set.

References cxx::Bits::Bst< Node, Get_key, Compare >::rend().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/avl_set

15.21 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node Class Reference

A smart pointer to a tree item.

Collaboration diagram for cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node:

Public Member Functions

- Node ()

  Default construction for NIL pointer.
- Item_type const & operator∗ ()

  Dereference the pointer.
- Item_type const ∗ operator-> ()

  Dereferenced member access.
- bool valid () const

  Validity check.
- operator Item_type const ∗ ()

  Cast to a real item pointer.

15.21.1 Detailed Description

template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GET_KEY> class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node

A smart pointer to a tree item.

Definition at line 175 of file avl_set.

15.21.2 Member Function Documentation

15.21.2.1 operator∗()

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Item_type const& cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::operator*
( ) [inline]
```

Dereference the pointer.

Precondition

Node is valid.

Definition at line 192 of file avl_set.

15.21.2.2 operator->()

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
Item_type const* cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::operator->
( ) [inline]
```

Dereferenced member access.

Precondition

Node is valid.

Definition at line 198 of file avl_set.

15.21.2.3 valid()

```
template<typename ITEM_TYPE , class COMPARE , template< typename A > class ALLOC, typename
GET_KEY >
bool cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::valid ( ) const
[inline]
```

Validity check.

Returns

false if the pointer is NIL, true if valid.

Definition at line 204 of file avl_set.

The documentation for this class was generated from the following file:

- l4/cxx/avl_set

15.22 cxx::Bits::Basic_list< POLICY > Class Template Reference

Internal: Common functions for all head-based list implementations.

```
#include <list_basics.h>
```

Inheritance diagram for cxx::Bits::Basic_list< POLICY >:

Collaboration diagram for cxx::Bits::Basic_list< POLICY >:

Public Member Functions

- bool empty () const

  Check if the list is empty.
- Value_type front () const

  Return the first element in the list.
- void clear ()

  Remove all elements from the list.
- Iterator begin ()

  Return an iterator to the beginning of the list.
- Const_iterator begin () const

  Return a const iterator to the beginning of the list.
- Const_iterator end () const

  Return a const iterator to the end of the list.
- Iterator end ()

  Return an iterator to the end of the list.

Static Public Member Functions

- static Const_iterator iter (Const_value_type c)

  Return a const iterator that begins at the given element.

Protected Attributes

- POLICY::Head_type _f

  Pointer to front of the list.

15.22.1 Detailed Description

template<typename POLICY>
class cxx::Bits::Basic_list< POLICY >

Internal: Common functions for all head-based list implementations.

Definition at line 50 of file list_basics.h.

15.22.2 Member Function Documentation

15.22.2.1 clear()

```
template<typename POLICY >
void cxx::Bits::Basic_list< POLICY >::clear ( )  [inline]
```

Remove all elements from the list.

After the operation the state of the elements is undefined.

Definition at line 146 of file list_basics.h.

References cxx::Bits::Basic_list< POLICY >::_f.

15.22.2.2 iter()

```
template<typename POLICY >
static Const_iterator cxx::Bits::Basic_list< POLICY >::iter (
            Const_value_type c )  [inline], [static]
```

Return a const iterator that begins at the given element.

Parameters

c	Element where the iterator should start.

Precondition

The element c must already be in a list.

Definition at line 159 of file list_basics.h.

The documentation for this class was generated from the following file:

- l4/cxx/bits/list_basics.h

15.23 cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference

Basic binary search tree (BST).

```
#include <bst.h>
```

Inheritance diagram for cxx::Bits::Bst< Node, Get_key, Compare >:

Collaboration diagram for cxx::Bits::Bst< Node, Get_key, Compare >:

Public Types

- typedef Get_key::Key_type Key_type

  The type of key values used to generate the total order of the elements.
- typedef Type_traits< Key_type >::Param_type Key_param_type

  The type for key parameters.
- typedef Fwd Fwd_iter_ops

Helper for building forward iterators for different wrapper classes.

- typedef Rev Rev_iter_ops

  Helper for building reverse iterators for different wrapper classes.

Iterators

Forward iterator.

- typedef __Bst_iter< Node, Node, Fwd > **Iterator**
- typedef __Bst_iter< Node, Node const, Fwd > Const_iterator

  Constant forward iterator.
- typedef __Bst_iter< Node, Node, Rev > Rev_iterator

  Backward iterator.
- typedef __Bst_iter< Node, Node const, Rev > Const_rev_iterator

  Constant backward.

Public Member Functions

Get default iterators for the ordered tree.

- Const_iterator begin () const

  Get the constant forward iterator for the first element in the set.
- Const_iterator end () const

  Get the end marker for the constant forward iterator.
- Iterator begin ()

  Get the mutable forward iterator for the first element of the set.
- Iterator end ()

  Get the end marker for the mutable forward iterator.
- Const_rev_iterator rbegin () const

  Get the constant backward iterator for the last element in the set.
- Const_rev_iterator rend () const

  Get the end marker for the constant backward iterator.
- Rev_iterator rbegin ()

  Get the mutable backward iterator for the last element of the set.
- Rev_iterator rend ()

  Get the end marker for the mutable backward iterator.

Lookup functions.

- Node * find_node (Key_param_type key) const

  find the node with the given key.
- Node * lower_bound_node (Key_param_type key) const

  Find the first node with a key not less than the given `key`.
- Const_iterator find (Key_param_type key) const

  find the node with the given key.
- template<typename FUNC >
  void remove_all (FUNC &&callback)

  Clear the tree.

Static Protected Member Functions

- template<typename FUNC >
  static void remove_tree (Bst_node *head, FUNC &&callback)

  Remove all elements in the subtree of head.

Interior access for descendants.

As this class is an intended base class we provide protected access to our interior, use 'using' to make this private in concrete implementations.

- Bst_node * _head

  The head pointer of the tree.
- Bst ()

  Create an empty tree.
- Node * head () const

  Access the head node as object of type Node.
- static Key_type k (Bst_node const *n)

  Get the key value of n.
- static Dir dir (Key_param_type l, Key_param_type r)

  Get the direction to go from `l` to search for `r`.
- static Dir dir (Key_param_type l, Bst_node const *r)

  Get the direction to go from `l` to search for `r`.
- static bool greater (Key_param_type l, Key_param_type r)

  Is l greater than r.
- static bool greater (Key_param_type l, Bst_node const *r)

  Is l greater than r.
- static bool greater (Bst_node const *l, Bst_node const *r)

  Is l greater than r.

15.23.1 Detailed Description

template<typename Node, typename Get_key, typename Compare>
class cxx::Bits::Bst< Node, Get_key, Compare >

Basic binary search tree (BST).

This class is intended as a base class for concrete binary search trees, such as an AVL tree. This class already provides the basic lookup methods and iterator definitions for a BST.

Definition at line 40 of file bst.h.

15.23.2 Member Function Documentation

15.23.2.1 begin() [1/2]

```
template<typename Node , typename Get_key , typename Compare >
Iterator cxx::Bits::Bst< Node, Get_key, Compare >::begin ( ) [inline]
```

Get the mutable forward iterator for the first element of the set.

Returns

The mutable forward iterator for the first element of the set.

Definition at line 190 of file bst.h.

References cxx::Bits::Bst< Node, Get_key, Compare >::head().

Here is the call graph for this function:

15.23.2.2 begin() [2/2]

```
template<typename Node , typename Get_key , typename Compare >
Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::begin ( ) const [inline]
```

Get the constant forward iterator for the first element in the set.

Returns

Constant forward iterator for the first element in the set.

Definition at line 179 of file bst.h.

References cxx::Bits::Bst< Node, Get_key, Compare >::head().

Referenced by cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin().

Here is the call graph for this function:

Here is the caller graph for this function:

15.23.2.3 dir() [1/2]

```
template<typename Node , typename Get_key , typename Compare >
static Dir cxx::Bits::Bst< Node, Get_key, Compare >::dir (
            Key_param_type l,
            Bst_node const * r )  [inline], [static], [protected]
```

Get the direction to go from `l` to search for `r`.

Parameters

l	is the key to look for.
r	is the node at the current position.

Return values

Direction::L	For left.
Direction::R	For right.
Direction::N	If `l` is equal to `r`.

Definition at line 135 of file bst.h.

References cxx::Bits::Bst< Node, Get_key, Compare >::dir(), and cxx::Bits::Bst< Node, Get_key, Compare >::k().

Here is the call graph for this function:

15.23.2.4 dir() [2/2]

```
template<typename Node , typename Get_key , typename Compare >
static Dir cxx::Bits::Bst< Node, Get_key, Compare >::dir (
            Key_param_type l,
            Key_param_type r )  [inline], [static], [protected]
```

Get the direction to go from `l` to search for `r`.

Parameters

l	is the key to look for.
r	is the key at the current position.

Return values

Direction::L	for left
Direction::R	for right
Direction::N	if `l` is equal to `r`.

Definition at line 118 of file bst.h.

References cxx::Bits::Direction::L, and cxx::Bits::Direction::N.

Referenced by cxx::Bits::Bst< Node, Get_key, Compare >::dir().

Here is the caller graph for this function:

15.23.2.5 end() [1/2]

```
template<typename Node , typename Get_key , typename Compare >
Iterator cxx::Bits::Bst< Node, Get_key, Compare >::end ( )  [inline]
```

Get the end marker for the mutable forward iterator.

Returns

The end marker for mutable forward iterator.

Definition at line 195 of file bst.h.

15.23.2.6 end() [2/2]

```
template<typename Node , typename Get_key , typename Compare >
Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::end ( ) const  [inline]
```

Get the end marker for the constant forward iterator.

Returns

The end marker for the constant forward iterator.

Definition at line 184 of file bst.h.

Referenced by cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::end().

Here is the caller graph for this function:

15.23.2.7 find()

```
template<typename Node , typename Get_key , class Compare >
Bst< Node, Get_key, Compare >::Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::find
(
            Key_param_type key ) const  [inline]
```

find the node with the given *key*.

Parameters

key	The key value of the element to search.

Returns

A valid iterator for the node with the given *key*, or an invalid iterator if *key* was not found.

Definition at line 312 of file bst.h.

References cxx::Bits::Bst_node::next().

Here is the call graph for this function:

15.23.2.8 find_node()

```
template<typename Node , typename Get_key , class Compare >
Node * cxx::Bits::Bst< Node, Get_key, Compare >::find_node (
            Key_param_type key ) const  [inline]
```

find the node with the given *key*.

Parameters

key	The key value of the element to search.

Returns

A pointer to the node with the given *key*, or NULL if *key* was not found.

Definition at line 276 of file bst.h.

References cxx::Bits::Bst_node::next().

Referenced by cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::find_node().

Here is the call graph for this function:

Here is the caller graph for this function:

15.23.2.9 lower_bound_node()

```
template<typename Node , typename Get_key , class Compare >
Node * cxx::Bits::Bst< Node, Get_key, Compare >::lower_bound_node (
            Key_param_type key ) const  [inline]
```

Find the first node with a key not less than the given `key`.

Parameters

key	The key used for the search.

Returns

A pointer to the found node, or `NULL` if no node was found.

Definition at line 292 of file bst.h.

References cxx::Bits::Bst_node::next().

Referenced by cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::lower_bound_node().

Here is the call graph for this function:

Here is the caller graph for this function:

15.23.2.10 rbegin() [1/2]

```
template<typename Node , typename Get_key , typename Compare >
Rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rbegin ( )  [inline]
```

Get the mutable backward iterator for the last element of the set.

Returns

The mutable backward iterator for the last element of the set.

Definition at line 212 of file bst.h.

References cxx::Bits::Bst< Node, Get_key, Compare >::head().

Here is the call graph for this function:

15.23.2.11 rbegin() [2/2]

```
template<typename Node , typename Get_key , typename Compare >
Const_rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rbegin ( ) const  [inline]
```

Get the constant backward iterator for the last element in the set.

Returns

The constant backward iterator for the last element in the set.

Definition at line 201 of file bst.h.

References cxx::Bits::Bst< Node, Get_key, Compare >::head().

Referenced by cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rbegin().

Here is the call graph for this function:

Here is the caller graph for this function:

15.23.2.12 remove_all()

```
template<typename Node , typename Get_key , typename Compare >
template<typename FUNC >
void cxx::Bits::Bst< Node, Get_key, Compare >::remove_all (
            FUNC && callback )  [inline]
```

Clear the tree.

Parameters

callback	Optional function to be called on each removed element.

The callback may delete the elements. The function guarantees that the elements are no longer used after the callback has been called.

Definition at line 258 of file bst.h.

References cxx::Bits::Bst< Node, Get_key, Compare >::_head, cxx::Bits::Bst< Node, Get_key, Compare >::head(), and cxx::Bits::Bst< Node, Get_key, Compare >::remove_tree().

Here is the call graph for this function:

15.23.2.13 remove_tree()

```
template<typename Node , typename Get_key , typename Compare >
template<typename FUNC >
static void cxx::Bits::Bst< Node, Get_key, Compare >::remove_tree (
            Bst_node * head,
            FUNC && callback )  [inline], [static], [protected]
```

Remove all elements in the subtree of head.

Parameters

head	Head of the the subtree to remove
callback	Optional function called on each removed element.

Definition at line 158 of file bst.h.

References cxx::Bits::Bst< Node, Get_key, Compare >::head(), cxx::Bits::Direction::L, cxx::Bits::Bst_node::next(), and cxx::Bits::Direction::R.

Referenced by cxx::Bits::Bst< Node, Get_key, Compare >::remove_all().

Here is the call graph for this function:

Here is the caller graph for this function:

15.23.2.14 rend() [1/2]

```
template<typename Node , typename Get_key , typename Compare >
Rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rend ( )  [inline]
```

Get the end marker for the mutable backward iterator.

Returns

The end marker for mutable backward iterator.

Definition at line 217 of file bst.h.

15.23.2.15 rend() [2/2]

```
template<typename Node , typename Get_key , typename Compare >
Const_rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rend ( ) const  [inline]
```

Get the end marker for the constant backward iterator.

Returns

The end marker for the constant backward iterator.

Definition at line 206 of file bst.h.

Referenced by cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rend().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/bits/bst.h

15.24 cxx::Bits::Bst_node Class Reference

Basic type of a node in a binary search tree (BST).

```
#include <bst_base.h>
```

Inheritance diagram for cxx::Bits::Bst_node:

Collaboration diagram for cxx::Bits::Bst_node:

Protected Member Functions

- Bst_node ()

  Create uninitialized node.
- Bst_node (bool)

  Create initialized node.

Static Protected Member Functions

Access to BST linkage.

Provide access to the tree linkage to inherited classes Inherited nodes, such as AVL nodes should make these methods private via 'using'

- static Bst_node ∗ next (Bst_node const ∗p, Direction d)

  Get next node in direction d.
- static void next (Bst_node ∗p, Direction d, Bst_node ∗n)

  Set next node of p in direction d to n.
- static Bst_node ∗∗ next_p (Bst_node ∗p, Direction d)

  Get pointer to link in direction d.
- template<typename Node >
  static Node ∗ next (Bst_node const ∗p, Direction d)

  Get next node in direction d as type Node.
- static void rotate (Bst_node ∗∗t, Direction idir)

  Rotate subtree t in the opposite direction of idir.

15.24.1 Detailed Description

Basic type of a node in a binary search tree (BST).

Definition at line 77 of file bst_base.h.

The documentation for this class was generated from the following file:

- l4/cxx/bits/bst_base.h

15.25 cxx::Bits::Direction Struct Reference

The direction to go in a binary search tree.

```
#include <bst_base.h>
```

Collaboration diagram for cxx::Bits::Direction:

Public Types

- enum Direction_e { L = 0 , R = 1 , N = 2 }

  The literal direction values.

Public Member Functions

- Direction ()=default

  Uninitialized direction.
- Direction (Direction_e d)

  Convert a literal direction (L, R, N) to an object.
- Direction (bool b)

  Convert a boolean to a direction (false == L, true == R)
- Direction operator! () const

  Negate the direction.

Comparison operators (equality and inequality)

- bool **operator==** (Direction_e o) const
- bool **operator!=** (Direction_e o) const
- bool **operator==** (Direction o) const
- bool **operator!=** (Direction o) const

15.25.1 Detailed Description

The direction to go in a binary search tree.

Definition at line 39 of file bst_base.h.

15.25.2 Member Enumeration Documentation

15.25.2.1 Direction_e

```
enum cxx::Bits::Direction::Direction_e
```

The literal direction values.

Enumerator

L	Go to the left child.
R	Go to the right child.
N	Stop.

Definition at line 42 of file bst_base.h.

15.25.3 Member Function Documentation

15.25.3.1 operator"!()

```
Direction cxx::Bits::Direction::operator!  ( ) const  [inline]
```

Negate the direction.

Note

This is only defined for a current value of L or R

Definition at line 63 of file bst_base.h.

References Direction().

Here is the call graph for this function:

The documentation for this struct was generated from the following file:

- l4/cxx/bits/bst_base.h

15.26 cxx::Bits::Smart_ptr_list< ITEM > Class Template Reference

List of smart-pointer-managed objects.

```
#include <smart_ptr_list.h>
```

Collaboration diagram for cxx::Bits::Smart_ptr_list< ITEM >:

Public Member Functions

- void push_front (Next_type &&e)

  Add an element to the front of the list.
- void push_front (Next_type const &e)

  Add an element to the front of the list.
- void push_back (Next_type &&e)

  Add an element at the end of the list.
- void push_back (Next_type const &e)

  Add an element at the end of the list.
- Value_type * front () const

  Return a pointer to the first element in the list.
- Next_type pop_front ()

  Remove the element in front of the list and return it.
- bool empty () const

  Check if the list is empty.

15.26.1 Detailed Description

template<typename ITEM>
class cxx::Bits::Smart_ptr_list< ITEM >

List of smart-pointer-managed objects.

Template Parameters

ITEM	Type of the list items.

The list is implemented as a single-linked list connected via smart pointers, so that they are automatically cleaned up when they are removed from the list.

Definition at line 47 of file smart_ptr_list.h.

15.26.2 Member Function Documentation

15.26.2.1 pop_front()

```
template<typename ITEM >
Next_type cxx::Bits::Smart_ptr_list< ITEM >::pop_front ( ) [inline]
```

Remove the element in front of the list and return it.

Returns

The element that was previously in front of the list as a managed pointer or a nullptr-equivalent when the list was already empty.

Definition at line 150 of file smart_ptr_list.h.

The documentation for this class was generated from the following file:

- l4/cxx/bits/smart_ptr_list.h

15.27 cxx::Bits::Smart_ptr_list_item< T, STORE_T > Class Template Reference

List item for an arbitrary item in a Smart_ptr_list.

```
#include <smart_ptr_list.h>
```

Inheritance diagram for cxx::Bits::Smart_ptr_list_item< T, STORE_T >:

Collaboration diagram for cxx::Bits::Smart_ptr_list_item< T, STORE_T >:

15.27.1 Detailed Description

template<typename T, typename STORE_T>
class cxx::Bits::Smart_ptr_list_item< T, STORE_T >

List item for an arbitrary item in a Smart_ptr_list.

Template Parameters

T	Type of object to be stored in the list.
STORE↩ _T	Storage type for pointer to next item. The class must implement a get() function that returns a pointer to the stored object and destroy the stored object when the item goes out of scope.

Definition at line 28 of file smart_ptr_list.h.

The documentation for this class was generated from the following file:

- l4/cxx/bits/smart_ptr_list.h

15.28 cxx::H_list< T, POLICY > Class Template Reference

General double-linked list of unspecified cxx::H_list_item elements.

Inheritance diagram for cxx::H_list< T, POLICY >:

Collaboration diagram for cxx::H_list< T, POLICY >:

Public Member Functions

- void add (T *e)

  Add element to the front of the list.
- void push_front (T *e)

  Add element to the front of the list.
- T * pop_front ()

Remove and return the head element of the list.

- Iterator insert (T ∗e, Iterator const &pred)

  Insert an element at the iterator position.

Static Public Member Functions

- static Iterator iter (T ∗c)

  Return an iterator for an arbitrary list element.
- static bool in_list (T const ∗e)

  Check if the given element is currently part of a list.
- static Iterator insert_after (T ∗e, Iterator const &pred)

  Insert an element after the iterator position.
- static void insert_before (T ∗e, Iterator const &succ)

  Insert an element before the iterator position.
- static void replace (T ∗p, T ∗e)

  Replace an element in a list with a new element.
- static void remove (T ∗e)

  Remove the given element from its list.
- static Iterator erase (Iterator const &e)

  Remove the element at the given iterator position.

Additional Inherited Members

15.28.1 Detailed Description

template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
class cxx::H_list< T, POLICY >

General double-linked list of unspecified cxx::H_list_item elements.

Most of the time, you want to use H_list_t.

Definition at line 80 of file hlist.

15.28.2 Member Function Documentation

15.28.2.1 erase()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static Iterator cxx::H_list< T, POLICY >::erase (
            Iterator const & e )  [inline], [static]
```

Remove the element at the given iterator position.

Parameters

e	Iterator pointing to the element to be removed. Must not point to end().

Returns

New iterator pointing to the element after the removed one.

Note

The hlist implementation guarantees that the original iterator is still valid after the element has been removed. In fact, the iterator returned is the same as the one supplied in the e parameter.

Definition at line 247 of file hlist.

Referenced by L4::Ipc_svr::Timeout_queue::handle_expired_timeouts().

Here is the caller graph for this function:

15.28.2.2 insert()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
Iterator cxx::H_list< T, POLICY >::insert (
            T * e,
            Iterator const & pred )  [inline]
```

Insert an element at the iterator position.

Parameters

e	New Element to be inserted
pred	Iterator pointing to the element after which the element will be inserted. If end() is given, the element will be inserted at the beginning of the queue.

Returns

Iterator pointing to the newly inserted element.

Definition at line 144 of file hlist.

References cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item > >::_f, cxx::Bits::Basic_list< POLICY >::end(), and cxx::H_list< T, POLICY >::iter().

Here is the call graph for this function:

15.28.2.3 insert_after()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static Iterator cxx::H_list< T, POLICY >::insert_after (
            T * e,
            Iterator const & pred )  [inline], [static]
```

Insert an element after the iterator position.

Parameters

e	New element to be inserted.
pred	Iterator pointing to the element after which the element will be inserted. Must not be end().

Returns

Iterator pointing to the newly inserted element.

Precondition

The list must not be empty.

Definition at line 171 of file hlist.

References cxx::H_list< T, POLICY >::iter().

Here is the call graph for this function:

15.28.2.4 insert_before()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static void cxx::H_list< T, POLICY >::insert_before (
            T * e,
            Iterator const & succ )  [inline], [static]
```

Insert an element before the iterator position.

Parameters

e	New element to be inserted.
succ	Iterator pointing to the element before which the element will be inserted. Must not be end().

Definition at line 191 of file hlist.

Referenced by L4::Ipc_svr::Timeout_queue::add().

Here is the caller graph for this function:

15.28.2.5 iter()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static Iterator cxx::H_list< T, POLICY >::iter (
            T * c )  [inline], [static]
```

Return an iterator for an arbitrary list element.

Parameters

c	List element to start the iteration.

Returns

A mutable forward iterator.

Precondition

The element must be in a list.

Definition at line 104 of file hlist.

Referenced by cxx::H_list< T, POLICY >::insert(), and cxx::H_list< T, POLICY >::insert_after().

Here is the caller graph for this function:

15.28.2.6 pop_front()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
T* cxx::H_list< T, POLICY >::pop_front ( )  [inline]
```

Remove and return the head element of the list.

Precondition

The list must not be empty or the behaviour will be undefined.

Definition at line 127 of file hlist.

References cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item > >::front(), and cxx::H_list< T, POLICY >::remove().

Here is the call graph for this function:

15.28.2.7 remove()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static void cxx::H_list< T, POLICY >::remove (
            T * e )  [inline], [static]
```

Remove the given element from its list.

Parameters

e	Element to be removed. Must be in a list.

Definition at line 231 of file hlist.

Referenced by cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::alloc(), cxx::Base_slab< Obj_size, Slab_size, Max_free, All
cxx::H_list< T, POLICY >::pop_front(), and L4::Ipc_svr::Timeout_queue::remove().

Here is the caller graph for this function:

15.28.2.8 replace()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static void cxx::H_list< T, POLICY >::replace (
            T * p,
            T * e )  [inline], [static]
```

Replace an element in a list with a new element.

Parameters

p	Element in list to be replaced.
e	Replacement element, must not yet be in a list.

Precondition

`p` and `e` must not be NULL.

After the operation the p element is no longer in the list and may be reused.

Definition at line 215 of file hlist.

The documentation for this class was generated from the following file:

- l4/cxx/hlist

15.29 cxx::H_list_item_t< ELEM_TYPE > Class Template Reference

Basic element type for a double-linked H_list.

Inheritance diagram for cxx::H_list_item_t< ELEM_TYPE >:

Collaboration diagram for cxx::H_list_item_t< ELEM_TYPE >:

Public Member Functions

- H_list_item_t ()

  Constructor.
- ~H_list_item_t () noexcept

  Destructor.

15.29.1 Detailed Description

template<typename ELEM_TYPE>
class cxx::H_list_item_t< ELEM_TYPE >

Basic element type for a double-linked H_list.

Template Parameters

ELEM_TYPE	Base class of the list element.

Definition at line 33 of file hlist.

15.29.2 Constructor & Destructor Documentation

15.29.2.1 H_list_item_t()

```
template<typename ELEM_TYPE >
cxx::H_list_item_t< ELEM_TYPE >::H_list_item_t ( ) [inline]
```

Constructor.

Creates an element that is not in any list.

Definition at line 41 of file hlist.

15.29.2.2 ~H_list_item_t()

```
template<typename ELEM_TYPE >
cxx::H_list_item_t< ELEM_TYPE >::~H_list_item_t ( ) [inline], [noexcept]
```

Destructor.

Automatically removes the element from any list it still might be enchained in.

Definition at line 48 of file hlist.

The documentation for this class was generated from the following file:

- l4/cxx/hlist

15.30 cxx::H_list_t< T > Struct Template Reference

Double-linked list of typed H_list_item_t elements.

Inheritance diagram for cxx::H_list_t< T >:

Collaboration diagram for cxx::H_list_t< T >:

Additional Inherited Members

15.30.1 Detailed Description

template<typename T>
struct cxx::H_list_t< T >

Double-linked list of typed H_list_item_t elements.

Note

H_lists are not self-cleaning. Elements that are still chained during destruction are not removed and will therefore be in an undefined state after the destruction.

Definition at line 259 of file hlist.

The documentation for this struct was generated from the following file:

- l4/cxx/hlist

15.31 cxx::List< D, Alloc > Class Template Reference

Doubly linked list, with internal allocation.

Collaboration diagram for cxx::List< D, Alloc >:

Data Structures

- class Iter

  Iterator.

Public Member Functions

- void push_back (D const &d) throw ()

  Add element at the end of the list.
- void push_front (D const &d) throw ()

  Add element at the beginning of the list.
- void remove (Iter const &i) throw ()

  Remove element pointed to by the iterator.
- unsigned long size () const throw ()

  Get the length of the list.
- D const & operator[ ] (unsigned long idx) const throw ()

  Random access.
- D & operator[ ] (unsigned long idx) throw ()

  Random access.
- Iter items () throw ()

  Get iterator for the list elements.

15.31.1 Detailed Description

template<typename D, template< typename A > class Alloc = New_allocator>
class cxx::List< D, Alloc >

Doubly linked list, with internal allocation.

Container for items of type D, implemented by a doubly linked list. Alloc defines the allocator policy.

Definition at line 334 of file list.

15.31.2 Member Function Documentation

15.31.2.1 operator[]() [1/2]

```
template<typename D , template< typename A > class Alloc = New_allocator>
D& cxx::List< D, Alloc >::operator[] (
            unsigned long idx ) throw ( )   [inline]
```

Random access.

Complexity is O(n).

Definition at line 408 of file list.

15.31.2.2 operator[]() [2/2]

```
template<typename D , template< typename A > class Alloc = New_allocator>
D const& cxx::List< D, Alloc >::operator[] (
            unsigned long idx ) const throw ( )   [inline]
```

Random access.

Complexity is O(n).

Definition at line 404 of file list.

The documentation for this class was generated from the following file:

- l4/cxx/list

15.32 cxx::List< D, Alloc >::Iter Class Reference

Iterator.

Collaboration diagram for cxx::List< D, Alloc >::Iter:

Public Member Functions

- operator E∗ () const throw ()

  operator for testing validity (syntactically equal to pointers)

15.32.1 Detailed Description

template<typename D, template< typename A > class Alloc = New_allocator>
class cxx::List< D, Alloc >::Iter

Iterator.

Forward and backward iteratable.

Definition at line 354 of file list.

The documentation for this class was generated from the following file:

- l4/cxx/list

15.33 cxx::List_alloc Class Reference

Standard list-based allocator.

Collaboration diagram for cxx::List_alloc:

Public Member Functions

- List_alloc ()

  Initializes an empty list allocator.
- void free (void ∗block, unsigned long size, bool initial_free=false)

  Return a free memory block to the allocator.
- void ∗ alloc (unsigned long size, unsigned long align)

  Allocate a memory block.
- void ∗ alloc_max (unsigned long min, unsigned long ∗max, unsigned long align, unsigned granularity)

  Allocate a memory block of `min` *<= size <=* `max`*.*
- unsigned long avail ()

  Get the amount of available memory.

15.33.1 Detailed Description

Standard list-based allocator.

Definition at line 31 of file list_alloc.

15.33.2 Constructor & Destructor Documentation

15.33.2.1 List_alloc()

```
cxx::List_alloc::List_alloc ( ) [inline]
```

Initializes an empty list allocator.

Note

To initialize the allocator with available memory use the free() function.

Definition at line 56 of file list_alloc.

15.33.3 Member Function Documentation

15.33.3.1 alloc()

```
void * cxx::List_alloc::alloc (
            unsigned long size,
            unsigned long align ) [inline]
```

Allocate a memory block.

Parameters

size	Size of the memory block.
align	Alignment constraint.

Returns

Pointer to memory block

Precondition

0 < `size` <= ~`0UL - 32`.

Definition at line 363 of file list_alloc.

15.33.3.2 alloc_max()

```
void * cxx::List_alloc::alloc_max (
            unsigned long min,
            unsigned long * max,
            unsigned long align,
            unsigned granularity ) [inline]
```

Allocate a memory block of `min` <= size <= `max`.

Parameters

	min	Minimal size to allocate (in bytes).
in,out	*max*	Maximum size to allocate (in bytes). The actual allocated size is returned here.
	align	Alignment constraint.
	granularity	Granularity to use for the allocation (power of 2).

Returns

Pointer to memory block

Precondition

$0 < \texttt{min} <= \sim\texttt{0UL} - 32$.

$0 < \texttt{max}$.

Definition at line 266 of file list_alloc.

References l4_round_size(), l4_trunc_size(), cxx::max(), and cxx::min().

Here is the call graph for this function:

15.33.3.3 avail()

```
unsigned long cxx::List_alloc::avail ( ) [inline]
```

Get the amount of available memory.

Returns

Available memory in bytes

Definition at line 434 of file list_alloc.

15.33.3.4 free()

```
void cxx::List_alloc::free (
            void * block,
            unsigned long size,
            bool initial_free = false )  [inline]
```

Return a free memory block to the allocator.

Parameters

block	Pointer to memory block.
size	Size of memory block.
initial_free	Set to true for putting fresh memory to the allocator. This will enforce alignment on that memory.

Precondition

`block` must not be NULL.

`2 * sizeof(void *) <= size <= ~0UL - 32.`

Definition at line 227 of file list_alloc.

The documentation for this class was generated from the following file:

- l4/cxx/list_alloc

15.34 cxx::List_item Class Reference

Basic list item.

Inherited by cxx::T_list_item< T >.

Collaboration diagram for cxx::List_item:

Data Structures

- class Iter

  Iterator for a list of ListItem-s.
- class T_iter

  Iterator for derived classes from ListItem.

Public Member Functions

- List_item ∗ get_prev_item () const throw ()

  Get previous item.
- List_item ∗ get_next_item () const throw ()

  Get next item.
- void insert_prev_item (List_item ∗p) throw ()

  Insert item p before this item.
- void insert_next_item (List_item ∗p) throw ()

  Insert item p after this item.
- void remove_me () throw ()

  Remove this item from the list.

Static Public Member Functions

- template<typename C , typename N >
  static C ∗ push_back (C ∗head, N ∗p) throw ()

  Append item to a list.
- template<typename C , typename N >
  static C ∗ push_front (C ∗head, N ∗p) throw ()

  Prepend item to a list.
- template<typename C , typename N >
  static C ∗ remove (C ∗head, N ∗p) throw ()

  Remove item from a list.

15.34.1 Detailed Description

Basic list item.

Basic item that can be member of a doubly linked, cyclic list.

Definition at line 37 of file list.

15.34.2 Member Function Documentation

15.34.2.1 push_back()

```
template<typename C , typename N >
C * cxx::List_item::push_back (
            C * head,
            N * p ) throw ( )   [inline], [static]
```

Append item to a list.

Convenience function for empty-head corner case.

Parameters

head	Pointer to the current list head.
p	Pointer to new item.

Returns

the pointer to the new head.

Definition at line 248 of file list.

15.34.2.2 push_front()

```
template<typename C , typename N >
C * cxx::List_item::push_front (
            C * head,
            N * p ) throw ( )   [inline], [static]
```

Prepend item to a list.

Convenience function for empty-head corner case.

Parameters

head	pointer to the current list head.
p	pointer to new item.

Returns

the pointer to the new head.

Definition at line 259 of file list.

15.34.2.3 remove()

```
template<typename C , typename N >
C * cxx::List_item::remove (
            C * head,
            N * p ) throw ( )   [inline], [static]
```

Remove item from a list.

Convenience function for remove-head corner case.

Parameters

head	pointer to the current list head.
p	pointer to the item to remove.

Returns

the pointer to the new head.

Definition at line 269 of file list.

The documentation for this class was generated from the following file:

- l4/cxx/list

15.35 cxx::List_item::Iter Class Reference

Iterator for a list of ListItem-s.

Inheritance diagram for cxx::List_item::Iter:

Collaboration diagram for cxx::List_item::Iter:

Public Member Functions

- List_item * remove_me () throw ()

  Remove item pointed to by iterator, and return pointer to element.

15.35.1 Detailed Description

Iterator for a list of ListItem-s.

The Iterator iterates till it finds the first element again.

Definition at line 45 of file list.

The documentation for this class was generated from the following file:

- l4/cxx/list

15.36 cxx::List_item::T_iter< T, Poly > Class Template Reference

Iterator for derived classes from ListItem.

Inheritance diagram for cxx::List_item::T_iter< T, Poly >:

Collaboration diagram for cxx::List_item::T_iter< T, Poly >:

Additional Inherited Members

15.36.1 Detailed Description

template<typename T, bool Poly = false>
class cxx::List_item::T_iter< T, Poly >

Iterator for derived classes from ListItem.

Allows direct access to derived classes by ∗ operator.

Example: class Foo : public ListItem { public: typedef T_iter<Foo> Iter; ... };

Definition at line 119 of file list.

The documentation for this class was generated from the following file:

- l4/cxx/list

15.37 cxx::Lt_functor< Obj > Struct Template Reference

Generic comparator class that defaults to the less-than operator.

Collaboration diagram for cxx::Lt_functor< Obj >:

15.37.1 Detailed Description

template<typename Obj>
struct cxx::Lt_functor< Obj >

Generic comparator class that defaults to the less-than operator.

Definition at line 29 of file std_ops.

The documentation for this struct was generated from the following file:

- l4/cxx/std_ops

15.38 cxx::New_allocator< _Type > Class Template Reference

Standard allocator based on `operator new ()` .

Inheritance diagram for cxx::New_allocator< _Type >:

Collaboration diagram for cxx::New_allocator< _Type >:

15.38.1 Detailed Description

template<typename _Type>
class cxx::New_allocator< _Type >

Standard allocator based on `operator new ()` .

This allocator is the default allocator used for the *cxx Containers*, such as cxx::Avl_set and cxx::Avl_map, to allocate the internal data structures.

Definition at line 67 of file std_alloc.

The documentation for this class was generated from the following file:

- l4/cxx/std_alloc

15.39 cxx::Nothrow Class Reference

Helper type to distinguish the `oeprator new` version that does not throw exceptions.

Collaboration diagram for cxx::Nothrow:

15.39.1 Detailed Description

Helper type to distinguish the `oeprator new` version that does not throw exceptions.

Definition at line 30 of file std_alloc.

The documentation for this class was generated from the following file:

- l4/cxx/std_alloc

15.40 cxx::Pair< First, Second > Struct Template Reference

Pair of two values.

Collaboration diagram for cxx::Pair< First, Second >:

Public Types

- typedef First First_type

  Type of first value.
- typedef Second Second_type

  Type of second value.

Public Member Functions

- template<typename A1 , typename A2 >
  Pair (A1 &&first, A2 &&second)

  Create a pair from the two values.
- Pair ()=default

  Default construction.

Data Fields

- First first

  First value.
- Second second

  Second value.

15.40.1 Detailed Description

template<typename First, typename Second>
struct cxx::Pair< First, Second >

Pair of two values.

Standard container for a pair of values.

Parameters

First	Type of the first value.
Second	Type of the second value.

Definition at line 36 of file pair.

15.40.2 Constructor & Destructor Documentation

15.40.2.1 Pair()

```
template<typename First , typename Second >
template<typename A1 , typename A2 >
```

```
cxx::Pair< First, Second >::Pair (
            A1 && first,
            A2 && second )  [inline]
```

Create a pair from the two values.

Parameters

first	The first value.
second	The second value.

Definition at line 54 of file pair.

The documentation for this struct was generated from the following file:

- l4/cxx/pair

15.41 cxx::Pair_first_compare< Cmp, Typ > Class Template Reference

Comparison functor for Pair.

Collaboration diagram for cxx::Pair_first_compare< Cmp, Typ >:

Public Member Functions

- Pair_first_compare (Cmp const &cmp=Cmp())

  Construction.
- bool operator() (Typ const &l, Typ const &r) const

  Do the comaprison based on the first value.

15.41.1 Detailed Description

template<typename Cmp, typename Typ>
class cxx::Pair_first_compare< Cmp, Typ >

Comparison functor for Pair.

Parameters

Cmp	Comparison functor for the first value of the pair.
Typ	The pair type.

This functor can be used to compare Pair values with respect to the first value.

Definition at line 75 of file pair.

15.41.2 Constructor & Destructor Documentation

15.41.2.1 Pair_first_compare()

```
template<typename Cmp , typename Typ >
cxx::Pair_first_compare< Cmp, Typ >::Pair_first_compare (
            Cmp const & cmp = Cmp() )  [inline]
```

Construction.

Parameters

cmp	The comparison functor used for the first value.

Definition at line 85 of file pair.

15.41.3 Member Function Documentation

15.41.3.1 operator()()

```
template<typename Cmp , typename Typ >
bool cxx::Pair_first_compare< Cmp, Typ >::operator() (
            Typ const & l,
            Typ const & r ) const  [inline]
```

Do the comaprison based on the first value.

Parameters

l	The lefthand value.
r	The righthand value.

Definition at line 92 of file pair.

The documentation for this class was generated from the following file:

- l4/cxx/pair

15.42 cxx::Ref_obj_list_item< T > Struct Template Reference

Item for list linked via cxx::Ref_ptr with default refence counting.

Inheritance diagram for cxx::Ref_obj_list_item< T >:

Collaboration diagram for cxx::Ref_obj_list_item< T >:

15.42.1 Detailed Description

template<typename T>
struct cxx::Ref_obj_list_item< T >

Item for list linked via cxx::Ref_ptr with default refence counting.

Definition at line 27 of file ref_ptr_list.

The documentation for this struct was generated from the following file:

- l4/cxx/ref_ptr_list

15.43 cxx::Ref_ptr< T, CNT > Class Template Reference

A reference-counting pointer with automatic cleanup.

Inheritance diagram for cxx::Ref_ptr< T, CNT >:

Collaboration diagram for cxx::Ref_ptr< T, CNT >:

Public Member Functions

- Ref_ptr () noexcept

  Default constructor creates a pointer with no managed object.
- Ref_ptr (Wp const &o) noexcept

  Create a shared pointer from a weak pointer.
- Ref_ptr (decltype(nullptr) n) noexcept

  allow creation from `nullptr`
- template<typename X >
  Ref_ptr (X *o) noexcept

  Create a shared pointer from a raw pointer.
- Ref_ptr (T *o, bool d) noexcept

  Create a shared pointer from a raw pointer without creating a new reference.
- T * get () const noexcept

  Return a raw pointer to the object this shared pointer points to.
- T * ptr () const noexcept

  Return a raw pointer to the object this shared pointer points to.
- T * release () noexcept

  Release the shared pointer without removing the reference.

15.43.1 Detailed Description

template<typename T = void, template< typename X > class CNT = Default_ref_counter>
class cxx::Ref_ptr< T, CNT >

A reference-counting pointer with automatic cleanup.

Template Parameters

T	Type of object the pointer points to.
CNT	Type of management class that manages the life time of the object.

This pointer is similar to the standard C++-11 shared_ptr but it does the reference counting directly in the object being pointed to, so that no additional management structures need to be allocated from the heap.

Classes that use this pointer type must implement two functions:

```
int remove_ref()
```

is called when a reference is removed and must return 0 when there are no further references to the object.

```
void add_ref()
```

is called when another ref_ptr to the object is created.

Ref_obj provides a simple implementation of this interface from which classes may inherit.

Examples

tmpfs/lib/src/fs.cc.

Definition at line 81 of file ref_ptr.

15.43.2 Constructor & Destructor Documentation

15.43.2.1 Ref_ptr() [1/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
            Wp const & o )  [inline], [noexcept]
```

Create a shared pointer from a weak pointer.

Increases references.

Definition at line 98 of file ref_ptr.

15.43.2.2 Ref_ptr() [2/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
template<typename X >
cxx::Ref_ptr< T, CNT >::Ref_ptr (
            X * o )  [inline], [explicit], [noexcept]
```

Create a shared pointer from a raw pointer.

In contrast to C++11 shared_ptr it is safe to use this constructor multiple times and have the same reference counter.

Definition at line 111 of file ref_ptr.

15.43.2.3 Ref_ptr() [3/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
            T * o,
            bool d )  [inline], [noexcept]
```

Create a shared pointer from a raw pointer without creating a new reference.

Parameters

o	Pointer to the object.
d	Dummy parameter to select this constructor at compile time. The value may be true or false.

This is the counterpart to release().

Definition at line 124 of file ref_ptr.

15.43.3 Member Function Documentation

15.43.3.1 get()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T* cxx::Ref_ptr< T, CNT >::get ( ) const  [inline], [noexcept]
```

Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line 131 of file ref_ptr.

15.43.3.2 ptr()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T* cxx::Ref_ptr< T, CNT >::ptr ( ) const  [inline], [noexcept]
```

Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line 137 of file ref_ptr.

15.43.3.3 release()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T* cxx::Ref_ptr< T, CNT >::release ( )  [inline], [noexcept]
```

Release the shared pointer without removing the reference.

Returns

A raw pointer to the managed object.

Definition at line 148 of file ref_ptr.

The documentation for this class was generated from the following file:

- l4/cxx/ref_ptr

15.44 cxx::S_list< T, POLICY > Class Template Reference

Simple single-linked list.

Inheritance diagram for cxx::S_list< T, POLICY >:

Collaboration diagram for cxx::S_list< T, POLICY >:

Public Member Functions

- void add (T ∗e)

  Add an element to the front of the list.
- void push_front (T ∗e)

  Add an element to the front of the list.
- T ∗ pop_front ()

  Remove and return the head element of the list.

Additional Inherited Members

15.44.1 Detailed Description

template<typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item >>
class cxx::S_list< T, POLICY >

Simple single-linked list.

Template Parameters

T	Type of elements saved in the list. Must inherit from cxx::S_list_item

Definition at line 50 of file slist.

15.44.2 Member Function Documentation

15.44.2.1 pop_front()

```
template<typename T , typename POLICY = Bits::Basic_list_policy< T, S_list_item >>
T* cxx::S_list< T, POLICY >::pop_front ( )  [inline]
```

Remove and return the head element of the list.

Precondition

The list must not be empty or the behaviour will be undefined.

Definition at line 99 of file slist.

References cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > >::_f, and cxx::Bits::Basic_list< Bits::Basic_list_policy< T

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/slist

15.45 cxx::Slab< Type, Slab_size, Max_free, Alloc > Class Template Reference

Slab allocator for object of type `Type`.

Inheritance diagram for cxx::Slab< Type, Slab_size, Max_free, Alloc >:

Collaboration diagram for cxx::Slab< Type, Slab_size, Max_free, Alloc >:

Public Member Functions

- Type * alloc () throw ()

  Allocate an object of type `Type`.
- void free (Type *o) throw ()

  Free the object addressed by `o`.

Additional Inherited Members

15.45.1 Detailed Description

template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator> class cxx::Slab< Type, Slab_size, Max_free, Alloc >

Slab allocator for object of type `Type`.

Template Parameters

Type	The type of the objects to manage.
Slab_size	Size of a slab.
Max_free	The maximum number of free slabs.
Alloc	The allocator for the slabs.

Definition at line 346 of file slab_alloc.

15.45.2 Member Function Documentation

15.45.2.1 alloc()

```
template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A
> class Alloc = New_allocator>
Type* cxx::Slab< Type, Slab_size, Max_free, Alloc >::alloc ( ) throw ( )   [inline]
```

Allocate an object of type `Type`.

Returns

A pointer to the object just allocated, or 0 on failure.

Note

The user is responsible for initializing the object.

Definition at line 366 of file slab_alloc.

References cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::alloc().

Here is the call graph for this function:

15.45.2.2 free()

```
template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A
> class Alloc = New_allocator>
void cxx::Slab< Type, Slab_size, Max_free, Alloc >::free (
            Type * o ) throw ( )   [inline]
```

Free the object addressed by o.

Parameters

o	The pointer to the object to free.

Precondition

The object must have been allocated with this allocator.

Definition at line 377 of file slab_alloc.

The documentation for this class was generated from the following file:

- l4/cxx/slab_alloc

15.46 cxx::Slab_static< Type, Slab_size, Max_free, Alloc > Class Template Reference

Merged slab allocator (allocators for objects of the same size are merged together).

Inheritance diagram for cxx::Slab_static< Type, Slab_size, Max_free, Alloc >:

Collaboration diagram for cxx::Slab_static< Type, Slab_size, Max_free, Alloc >:

Public Member Functions

- Type * alloc () throw ()

  Allocate an object of type `Type`.

Additional Inherited Members

15.46.1 Detailed Description

template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator>
class cxx::Slab_static< Type, Slab_size, Max_free, Alloc >

Merged slab allocator (allocators for objects of the same size are merged together).

Template Parameters

Type	The type of the objects to manage.
Slab_size	The size of a slab.
Max_free	The maximum number of free slabs.
Alloc	The allocator for the slabs.

This slab allocator class is useful for merging slab allocators with the same parameters (equal `sizeof(Type)`, `Slab_size`, `Max_free`, and `Alloc` parameters) together and share the overhead for the slab caches among all equal-sized objects.

Definition at line 476 of file slab_alloc.

15.46.2 Member Function Documentation

15.46.2.1 alloc()

```
template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A
> class Alloc = New_allocator>
Type* cxx::Slab_static< Type, Slab_size, Max_free, Alloc >::alloc ( ) throw ( )   [inline]
```

Allocate an object of type `Type`.

Returns

A pointer to the just allocated object, or 0 on failure.

Note

The object is not zeroed out by the slab allocator.

Definition at line 489 of file slab_alloc.

The documentation for this class was generated from the following file:

- l4/cxx/slab_alloc

15.47 cxx::static_vector< T, IDX > Class Template Reference

Simple encapsulation for a dynamically allocated array.

Collaboration diagram for cxx::static_vector< T, IDX >:

Public Member Functions

- template<typename X , typename = typename enable_if<is_convertible<X, T>::value>::type>
  static_vector (static_vector< X, IDX > const &o)

  Conversion from compatible arrays.
- index_type index (value_type const *o) const

  Get the index of the given element of the array.

15.47.1 Detailed Description

template<typename T, typename IDX = unsigned>
class cxx::static_vector< T, IDX >

Simple encapsulation for a dynamically allocated array.

The main purpose of this class is to support C++11 range for for simple dynamically allocated array with static size.

Definition at line 16 of file static_vector.

The documentation for this class was generated from the following file:

- l4/cxx/static_vector

15.48 cxx::String Class Reference

Allocation free string class with explicit length field.

Inheritance diagram for cxx::String:

Collaboration diagram for cxx::String:

Public Types

- typedef char const ∗ Index

  Character index type.

Public Member Functions

- String (char const ∗s) throw ()

  Initialize from a zero-terminated string.
- String (char const ∗s, unsigned long len) throw ()

  Initialize from a pointer to first character and a length.
- String (char const ∗s, char const ∗e) throw ()

  Initialize with start and end pointer.
- String ()

  Zero-initialize. Create an invalid string.
- Index start () const

  Pointer to first character.
- Index end () const

  Pointer to first byte behind the string.
- int len () const

  Length.
- void start (char const ∗s)

  Set start.
- void len (unsigned long len)

  Set length.
- bool empty () const

  Check if the string has length zero.
- String head (Index end) const

Return prefix up to index.

- String head (unsigned long end) const

  Prefix of length `end`.
- String substr (unsigned long idx, unsigned long len=~0UL) const

  Substring of length `len` starting at `idx`.
- String substr (char const *start, unsigned long len=0) const

  Substring of length `len` starting at `start`.
- template<typename F >
  char const * find_match (F &&match) const

  Find matching character. `match` should be a function such as `isspace`.
- char const * find (char const *c) const

  Find character. Return end() if not found.
- char const * find (int c) const

  Find character. Return end() if not found.
- char const * rfind (char const *c) const

  Find right-most character. Return end() if not found.
- Index starts_with (cxx::String const &c) const

  Check if `c` is a prefix of string.
- char const * find (int c, char const *s) const

  Find character `c` starting at position `s`. Return end() if not found.
- char const * find (char const *c, char const *s) const

  Find character set at position.
- char const & operator[] (unsigned long idx) const

  Get character at `idx`.
- char const & operator[] (int idx) const

  Get character at `idx`.
- char const & operator[] (Index idx) const

  Get character at `idx`.
- bool eof (char const *s) const

  Check if pointer `s` points behind string.
- template<typename INT >
  int from_dec (INT *v) const

  Convert decimal string to integer.
- template<typename INT >
  int from_hex (INT *v) const

  Convert hex string to integer.
- bool operator== (String const &o) const

  Equality.
- bool operator!= (String const &o) const

  Inequality.

15.48.1 Detailed Description

Allocation free string class with explicit length field.

This class is used to group characters of a string which belong to one syntactical token types number, identifier, string, whitespace or another single character.

Stings in this class can contain null bytes and may denote parts of other strings.

Examples

tmpfs/lib/src/fs.cc.

Definition at line 41 of file string.

15.48.2 Constructor & Destructor Documentation

15.48.2.1 String()

```
cxx::String::String (
            char const * s,
            char const * e ) throw ( )   [inline]
```

Initialize with start and end pointer.

Parameters

s	first character of the string
e	pointer to first byte behind the string

Definition at line 59 of file string.

15.48.3 Member Function Documentation

15.48.3.1 find()

```
char const* cxx::String::find (
            char const * c,
            char const * s ) const  [inline]
```

Find character set at position.

Parameters

c	zero-terminated string of characters to search for
s	start position of search in string

Return values

end()	if no char in `c` is contained in string at or behind `s`.
position	in string of some character in `c`.

Definition at line 202 of file string.

References end(), and eof().

Here is the call graph for this function:

15.48.3.2 from_dec()

```
template<typename INT >
int cxx::String::from_dec (
            INT * v ) const  [inline]
```

Convert decimal string to integer.

Template Parameters

INT	result integer type

Parameters

out	*v*	conversion result

Returns

position of first character not converted.

Definition at line 239 of file string.

References eof(), and start().

Here is the call graph for this function:

15.48.3.3 from_hex()

```
template<typename INT >
int cxx::String::from_hex (
            INT * v ) const  [inline]
```

Convert hex string to integer.

Template Parameters

INT	result integer type

Parameters

out	*v*	conversion result

Return values

-1	if the maximal amount of digits fitting into `INT` have been read,
position	of first character not converted otherwise.

Definition at line 268 of file string.

References eof(), and start().

Here is the call graph for this function:

15.48.3.4 starts_with()

```
Index cxx::String::starts_with (
            cxx::String const & c ) const  [inline]
```

Check if c is a prefix of string.

Returns

0 if c is not a prefix, if it is a prefix, return first position not in c (which might be end()).

Definition at line 166 of file string.

References start().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/string

15.49 cxx::Weak_ref< T > Class Template Reference

Typed weak reference to an object of type `T`.

Inheritance diagram for cxx::Weak_ref< T >:

Collaboration diagram for cxx::Weak_ref< T >:

Additional Inherited Members

15.49.1 Detailed Description

template<typename T>
class cxx::Weak_ref< T >

Typed weak reference to an object of type `T`.

Template Parameters

T	The type of the referenced object.

A weak reference is a reference that is invalidated when the referenced object is about to be deleted. All weak references to an object are kept in a linked list and all the weak references are iterated and reset by the Weak_↩ ref_base::List destructor or Weak_ref_base::reset().

The type `T` must provide two methods that handle the housekeeping of weak references: remove_weak_ref(Weak↩ _ref_base ∗) and add_weak_ref(Weak_ref_base ∗). These functions must handle the insertion and removal of the weak reference into the respective Weak_ref_base::List object. For convenience one can use the cxx::Weak_ref_obj as a base class that handles weak references for you.

Definition at line 67 of file weak_ref.

The documentation for this class was generated from the following file:

- l4/cxx/weak_ref

15.50 cxx::Weak_ref_base Class Reference

Generic (base) weak reference to some object.

Inheritance diagram for cxx::Weak_ref_base:

Collaboration diagram for cxx::Weak_ref_base:

Additional Inherited Members

15.50.1 Detailed Description

Generic (base) weak reference to some object.

A weak reference is a reference that gets reset to NULL when the object shall be deleted. All weak references to the same object are kept in a linked list of weak references.

For typed weak references see `cxx::Weak_ref`.

Definition at line 25 of file weak_ref.

The documentation for this class was generated from the following file:

- l4/cxx/weak_ref

15.51 Elf32_Auxv Struct Reference

Auxiliary vector (32-bit).

```
#include <elf.h>
```

Collaboration diagram for Elf32_Auxv:

Data Fields

- Elf32_Word atype

15.51.1 Detailed Description

Auxiliary vector (32-bit).

Definition at line 958 of file elf.h.

15.51.2 Field Documentation

15.51.2.1 atype

```
Elf32_Word Elf32_Auxv::atype
```

See also

Elf_ATs

Definition at line 960 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.52 Elf32_Dyn Struct Reference

ELF32 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Dyn:

Data Fields

- Elf32_Sword d_tag
- Elf32_Word d_val

  integer values with various interpret.
- Elf32_Addr d_ptr

  program virtual addresses

15.52.1 Detailed Description

ELF32 dynamic entry.

Definition at line 509 of file elf.h.

15.52.2 Field Documentation

15.52.2.1 d_tag

```
Elf32_Sword Elf32_Dyn::d_tag
```

See also

Elf_DTs

Definition at line 511 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.53 Elf32_Ehdr Struct Reference

ELF32 header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Ehdr:

Data Fields

- unsigned char e_ident [EI_NIDENT]

  see Elf_EIs
- Elf32_Half e_type

  type of ELF file
- Elf32_Half e_machine

  required architecture
- Elf32_Word e_version

  file version
- Elf32_Addr e_entry

  initial program counter
- Elf32_Off e_phoff

  offset of program header table
- Elf32_Off e_shoff

  offset of file header table
- Elf32_Word e_flags

  processor-specific flags
- Elf32_Half e_ehsize

  size of ELF header

- Elf32_Half e_phentsize

  size of program header entry
- Elf32_Half e_phnum

  number of entries in program header table
- Elf32_Half e_shentsize

  size of section header entry
- Elf32_Half e_shnum

  number of entries in section header table
- Elf32_Half e_shstrndx

  section header table index of strtab

15.53.1 Detailed Description

ELF32 header.

Definition at line 121 of file elf.h.

15.53.2 Field Documentation

15.53.2.1 e_flags

`Elf32_Word Elf32_Ehdr::e_flags`

processor-specific flags

See also

Elf_EF_ARM_s

Definition at line 130 of file elf.h.

15.53.2.2 e_machine

`Elf32_Half Elf32_Ehdr::e_machine`

required architecture

See also

Elf_EMs

Definition at line 125 of file elf.h.

15.53.2.3 e_type

```
Elf32_Half Elf32_Ehdr::e_type
```

type of ELF file

See also

Elf_ETs

Definition at line 124 of file elf.h.

15.53.2.4 e_version

```
Elf32_Word Elf32_Ehdr::e_version
```

file version

See also

Elf_EVs

Definition at line 126 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.54 Elf32_Phdr Struct Reference

ELF32 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Phdr:

Data Fields

- Elf32_Word p_type
  type of program section
- Elf32_Off p_offset
  file offset of program section
- Elf32_Addr p_vaddr
  memory address of prog section
- Elf32_Addr p_paddr
  physical address (ignored)
- Elf32_Word p_filesz
  file size of program section
- Elf32_Word p_memsz
  memory size of program section
- Elf32_Word p_flags
  flags
- Elf32_Word p_align
  alignment of section

15.54.1 Detailed Description

ELF32 program header.

Definition at line 420 of file elf.h.

15.54.2 Field Documentation

15.54.2.1 p_flags

```
Elf32_Word Elf32_Phdr::p_flags
```

flags

See also

Elf_PFs

Definition at line 428 of file elf.h.

15.54.2.2 p_type

```
Elf32_Word Elf32_Phdr::p_type
```

type of program section

See also

Elf_PTs

Definition at line 422 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.55 Elf32_Rel Struct Reference

ELF32 relocation entry w/o addend.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Rel:

15.55.1 Detailed Description

ELF32 relocation entry w/o addend.

Definition at line 627 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.56 Elf32_Rela Struct Reference

ELF32 relocation entry w/ addend.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Rela:

15.56.1 Detailed Description

ELF32 relocation entry w/ addend.

Definition at line 634 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.57 Elf32_Shdr Struct Reference

ELF32 section header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Shdr:

Data Fields

- Elf32_Word sh_name

  name of sect (idx into strtab)
- Elf32_Word sh_type

  section's type
- Elf32_Word sh_flags

  section's flags
- Elf32_Addr sh_addr

  memory address of section
- Elf32_Off sh_offset

  file offset of section
- Elf32_Word sh_size

  file size of section
- Elf32_Word sh_link

  idx to associated header section
- Elf32_Word sh_info

  extra info of header section
- Elf32_Word sh_addralign

  address alignment constraints
- Elf32_Word sh_entsize

  size of entry if sect is table

15.57.1 Detailed Description

ELF32 section header.

Definition at line 342 of file elf.h.

15.57.2 Field Documentation

15.57.2.1 sh_flags

`Elf32_Word Elf32_Shdr::sh_flags`

section's flags

See also

Elf_SHFs

Definition at line 346 of file elf.h.

15.57.2.2 sh_type

`Elf32_Word Elf32_Shdr::sh_type`

section's type

See also

Elf_SHTs

Definition at line 345 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.58 Elf32_Sym Struct Reference

ELF32 symbol table entry.

`#include <elf.h>`

Collaboration diagram for Elf32_Sym:

Data Fields

- Elf32_Word st_name

  name of symbol (idx symstrtab)
- Elf32_Addr st_value

  value of associated symbol
- Elf32_Word st_size

  size of associated symbol
- unsigned char st_info

  type and binding info
- unsigned char st_other

  undefined
- Elf32_Half st_shndx

  associated section header

15.58.1 Detailed Description

ELF32 symbol table entry.

Definition at line 867 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.59 Elf64_Auxv Struct Reference

Auxiliary vector (64-bit).

```
#include <elf.h>
```

Collaboration diagram for Elf64_Auxv:

Data Fields

- Elf64_Word atype

15.59.1 Detailed Description

Auxiliary vector (64-bit).

Definition at line 965 of file elf.h.

15.59.2 Field Documentation

15.59.2.1 atype

```
Elf64_Word Elf64_Auxv::atype
```

See also

Elf_ATs

Definition at line 967 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.60 Elf64_Dyn Struct Reference

ELF64 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Dyn:

Data Fields

- Elf64_Sxword d_tag
- Elf64_Xword d_val

  integer values with various interpret.
- Elf64_Addr d_ptr

  program virtual addresses

15.60.1 Detailed Description

ELF64 dynamic entry.

Definition at line 520 of file elf.h.

15.60.2 Field Documentation

15.60.2.1 d_tag

```
Elf64_Sxword Elf64_Dyn::d_tag
```

See also

Elf_DTs

Definition at line 522 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.61 Elf64_Ehdr Struct Reference

ELF64 header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Ehdr:

Data Fields

- unsigned char e_ident [EI_NIDENT]

  see Elf_EIs
- Elf64_Half e_type

  type of ELF file
- Elf64_Half e_machine

  required architecture
- Elf64_Word e_version

  file version
- Elf64_Addr e_entry

  initial program counter
- Elf64_Off e_phoff

  offset of program header table
- Elf64_Off e_shoff

  offset of file header table
- Elf64_Word e_flags

  processor-specific flags
- Elf64_Half e_ehsize

  size of ELF header
- Elf64_Half e_phentsize

  size of program header entry
- Elf64_Half e_phnum

  number of entries in program header table
- Elf64_Half e_shentsize

  size of section header entry

- Elf64_Half e_shnum

  number of entries in section header table
- Elf64_Half e_shstrndx

  section header table index of strtab

15.61.1 Detailed Description

ELF64 header.

Definition at line 142 of file elf.h.

15.61.2 Field Documentation

15.61.2.1 e_flags

`Elf64_Word Elf64_Ehdr::e_flags`

processor-specific flags

See also

Elf_EF_ARM_s

Definition at line 151 of file elf.h.

15.61.2.2 e_machine

`Elf64_Half Elf64_Ehdr::e_machine`

required architecture

See also

Elf_EMs

Definition at line 146 of file elf.h.

15.61.2.3 e_type

```
Elf64_Half Elf64_Ehdr::e_type
```

type of ELF file

See also

Elf_ETs

Definition at line 145 of file elf.h.

15.61.2.4 e_version

```
Elf64_Word Elf64_Ehdr::e_version
```

file version

See also

Elf_EVs

Definition at line 147 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.62 Elf64_Phdr Struct Reference

ELF64 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Phdr:

Data Fields

- Elf64_Word p_type
  type of program section
- Elf64_Word p_flags
  flags
- Elf64_Off p_offset
  file offset of program section
- Elf64_Addr p_vaddr
  memory address of prog section
- Elf64_Addr p_paddr
  physical address (ignored)
- Elf64_Xword p_filesz
  file size of program section
- Elf64_Xword p_memsz
  memory size of program section
- Elf64_Xword p_align
  alignment of section

15.62.1 Detailed Description

ELF64 program header.

Definition at line 433 of file elf.h.

15.62.2 Field Documentation

15.62.2.1 p_flags

```
Elf64_Word Elf64_Phdr::p_flags
```

flags

See also

Elf_PFs

Definition at line 436 of file elf.h.

15.62.2.2 p_type

```
Elf64_Word Elf64_Phdr::p_type
```

type of program section

See also

Elf_PTs

Definition at line 435 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.63 Elf64_Rel Struct Reference

ELF64 relocation entry w/o addend.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Rel:

15.63.1 Detailed Description

ELF64 relocation entry w/o addend.

Definition at line 642 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.64 Elf64_Rela Struct Reference

ELF64 relocation entry w/ addend.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Rela:

15.64.1 Detailed Description

ELF64 relocation entry w/ addend.

Definition at line 649 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.65 Elf64_Shdr Struct Reference

ELF64 section header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Shdr:

Data Fields

- Elf64_Word sh_name

  name of sect (idx into strtab)
- Elf64_Word sh_type

  section's type
- Elf64_Xword sh_flags

  section's flags
- Elf64_Addr sh_addr

  memory address of section
- Elf64_Off sh_offset

  file offset of section
- Elf64_Xword sh_size

  file size of section
- Elf64_Word sh_link

  idx to associated header section
- Elf64_Word sh_info

  extra info of header section
- Elf64_Xword sh_addralign

  address alignment constraints
- Elf64_Xword sh_entsize

  size of entry if sect is table

15.65.1 Detailed Description

ELF64 section header.

Definition at line 357 of file elf.h.

15.65.2 Field Documentation

15.65.2.1 sh_flags

`Elf64_Xword Elf64_Shdr::sh_flags`

section's flags

See also

Elf_SHFs

Definition at line 361 of file elf.h.

15.65.2.2 sh_type

`Elf64_Word Elf64_Shdr::sh_type`

section's type

See also

Elf_SHTs

Definition at line 360 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.66 Elf64_Sym Struct Reference

ELF64 symbol table entry.

`#include <elf.h>`

Collaboration diagram for Elf64_Sym:

Data Fields

- Elf64_Word st_name
  name of symbol (idx symstrtab)
- unsigned char st_info
  type and binding info
- unsigned char st_other
  undefined
- Elf64_Half st_shndx
  associated section header
- Elf64_Addr st_value
  value of associated symbol
- Elf64_Xword st_size
  size of associated symbol

15.66.1 Detailed Description

ELF64 symbol table entry.

Definition at line 878 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

15.67 gfxbitmap_offset Struct Reference

offsets in pmap[] and bmap[]

```
#include <bitmap.h>
```

Collaboration diagram for gfxbitmap_offset:

Data Fields

- l4_uint32_t preskip_x

  skip pixels at beginning of line
- l4_uint32_t preskip_y

  skip lines
- l4_uint32_t endskip_x

  skip pixels at end of line

15.67.1 Detailed Description

offsets in pmap[] and bmap[]

Definition at line 67 of file bitmap.h.

The documentation for this struct was generated from the following file:

- l4/libgfxbitmap/bitmap.h

15.68 L4::Alloc_list Class Reference

A simple list-based allocator.

```
#include <alloc.h>
```

Collaboration diagram for L4::Alloc_list:

15.68.1 Detailed Description

A simple list-based allocator.

Definition at line 31 of file alloc.h.

The documentation for this class was generated from the following file:

- l4/cxx/alloc.h

15.69 L4::Arm_smccc Class Reference

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

Inherits L4::Kobject_0t< Derived, PROTO, S_DEMAND >.

Collaboration diagram for L4::Arm_smccc:

Public Member Functions

- l4_msgtag_t call (l4_umword_t func, l4_umword_t in0, l4_umword_t in1, l4_umword_t in2, l4_umword_t in3, l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0, l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3, l4_umword_t client_id)

  ARM SMC/HVC function call.

15.69.1 Detailed Description

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

See l4_arm_smccc_call() for the corresponding C interface.

Definition at line 24 of file arm_smccc.

15.69.2 Member Function Documentation

15.69.2.1 call()

```
l4_msgtag_t L4::Arm_smccc::call (
            l4_umword_t func,
            l4_umword_t in0,
            l4_umword_t in1,
            l4_umword_t in2,
            l4_umword_t in3,
            l4_umword_t in4,
            l4_umword_t in5,
            l4_umword_t * out0,
            l4_umword_t * out1,
            l4_umword_t * out2,
            l4_umword_t * out3,
            l4_umword_t client_id )
```

ARM SMC/HVC function call.

The input parameters consist of a function identifier, 6 arguments and a client id. Results are returned in 4 output parameters.

Parameters

	func	Function identifier. • Bit 31 has to be set: This marks the call as *Fast Call*. *Yielding Calls* (bit 31 unset) are rejected by the kernel. • Bit 30 defines the calling convention: • Bit 30 == 1: 64-bit calling convention. • Bit 30 == 0: 32-bit calling convention. • Bits 24..29 determine the service call ID. Only service IDs >= 0x30000000 (*Trusted Application Calls* and *Trusted OS Calls*) are allowed.
in	*in0*	First input parameter.
in	*in1*	Second input parameter.
in	*in2*	Third input parameter.
in	*in3*	Fourth input parameter.
in	*in4*	Fifth input parameter.
in	*in5*	Sixth input parameter.
out	*out0*	First output parameter.
out	*out1*	Second output parameter.
out	*out2*	Third output parameter.
out	*out3*	Fourth output parameter.
in	*client↩ _id*	Client ID. According to the specification, this value might be ignored by certain functions.

Return values

-L4_ENOSYS	Either bit 31 of the function call not set or service ID < 0x30000000.
-L4_EINVAL	Invalid number of parameters.
<0	Other L4 error.
0	Success.

The documentation for this class was generated from the following file:

- l4/sys/arm_smccc

15.70 L4::Base_exception Class Reference

Base class for all exceptions, thrown by the L4Re framework.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Base_exception:

Collaboration diagram for L4::Base_exception:

Public Member Functions

- virtual char const * str () const =0 throw ()

  Return a human readable string for the exception.
- virtual ~Base_exception () throw ()

  Destruction.

Protected Member Functions

- Base_exception () throw ()

  Create a base exception.

15.70.1 Detailed Description

Base class for all exceptions, thrown by the L4Re framework.

This is the abstract base of all exceptions thrown within the L4Re framework. It is basically also a good idea to use it as base of all user defined exceptions.

Definition at line 116 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

15.71 L4::Basic_registry Class Reference

This registry returns the corresponding server object based on the label of an Ipc_gate.

Inheritance diagram for L4::Basic_registry:

Collaboration diagram for L4::Basic_registry:

Static Public Member Functions

- static Value * find (l4_umword_t label)

Get the server object for an lpc_gate label.

- static l4_msgtag_t dispatch (l4_msgtag_t tag, l4_umword_t label, l4_utcb_t *utcb)

  The dispatch function called by the server loop.

15.71.1 Detailed Description

This registry returns the corresponding server object based on the label of an lpc_gate.

Definition at line 536 of file ipc_epiface.

15.71.2 Member Function Documentation

15.71.2.1 dispatch()

```
static l4_msgtag_t L4::Basic_registry::dispatch (
            l4_msgtag_t tag,
            l4_umword_t label,
            l4_utcb_t * utcb )  [inline], [static]
```

The dispatch function called by the server loop.

This function forwards the message to the server object identified by the given *label*.

Parameters

tag	The message tag used for the invocation.
label	The label used to find the object including the rights bits of the invoked capability.
utcb	The UTCB used for the invocation.

Returns

The return code from the object's dispatch function or -L4_ENOENT if the object does not exist.

Definition at line 561 of file ipc_epiface.

15.71.2.2 find()

```
static Value* L4::Basic_registry::find (
            l4_umword_t label )  [inline], [static]
```

Get the server object for an lpc_gate label.

Parameters

label	The label usually stored in an Ipc_gate.

Returns

A pointer to the Epiface identified by the given label.

Definition at line 545 of file ipc_epiface.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_epiface

15.72 L4::Bounds_error Class Reference

Access out of bounds.

Inheritance diagram for L4::Bounds_error:

Collaboration diagram for L4::Bounds_error:

Additional Inherited Members

15.72.1 Detailed Description

Access out of bounds.

Definition at line 289 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

15.73 L4::Cap< T > Class Template Reference

C++ interface for capabilities.

```
#include <capability.h>
```

Inheritance diagram for L4::Cap< T >:

Collaboration diagram for L4::Cap< T >:

Public Member Functions

- template<typename O >
  Cap (Cap< O > const &o) noexcept

  Create a copy from o, supporting implicit type casting.
- Cap (Cap_type cap) noexcept

  Constructor to create an invalid capability selector.
- Cap (l4_default_caps_t cap) noexcept

  Initialize capability with one of the default capability selectors.
- Cap (l4_cap_idx_t idx=L4_INVALID_CAP) noexcept

  Initialize capability, defaults to the invalid capability selector.
- Cap (No_init_type) noexcept

Create an uninitialized cap selector.

- Cap move (Cap const &src) const

  Move a capability to this cap slot.

- Cap copy (Cap const &src) const

  Copy a capability to this cap slot.

- T * operator-> () const noexcept

  Member access of a `T`.

Friends

- class **L4::Kobject**

Additional Inherited Members

15.73.1 Detailed Description

template<typename T>
class L4::Cap< T >

C++ interface for capabilities.

Template Parameters

T	Type of the object the capability points to.

The C++ version of a capability is comparable to a pointer, in fact it is a kind of smart pointer for our kernel objects and the objects derived from the kernel objects (L4::Kobject).

Add

```
#include <l4/sys/capability>
```

to your code to use the capability interface.

Examples

examples/clntsrv/client.cc, examples/libs/l4re/c++/shared_ds/ds_clnt.cc, and examples/libs/l4re/streammap/client.cc.

Definition at line 219 of file capability.h.

15.73.2 Constructor & Destructor Documentation

15.73.2.1 Cap() [1/4]

```
template<typename T >
template<typename O >
L4::Cap< T >::Cap (
            Cap< O > const & o )  [inline], [noexcept]
```

Create a copy from `o`, supporting implicit type casting.

Parameters

o	The source selector that shall be copied (and casted).

Definition at line 245 of file capability.h.

15.73.2.2 Cap() [2/4]

```
template<typename T >
L4::Cap< T >::Cap (
            Cap_type cap )  [inline], [noexcept]
```

Constructor to create an invalid capability selector.

Parameters

cap	Capability selector.

Definition at line 252 of file capability.h.

15.73.2.3 Cap() [3/4]

```
template<typename T >
L4::Cap< T >::Cap (
            l4_default_caps_t cap )  [inline], [noexcept]
```

Initialize capability with one of the default capability selectors.

Parameters

cap	Capability selector.

Definition at line 258 of file capability.h.

15.73.2.4 Cap() [4/4]

```
template<typename T >
L4::Cap< T >::Cap (
            l4_cap_idx_t idx = L4_INVALID_CAP )  [inline], [explicit], [noexcept]
```

Initialize capability, defaults to the invalid capability selector.

Parameters

idx	Capability selector.

Definition at line 264 of file capability.h.

15.73.3 Member Function Documentation

15.73.3.1 copy()

```
template<typename T >
Cap L4::Cap< T >::copy (
            Cap< T > const & src ) const  [inline]
```

Copy a capability to this cap slot.

Parameters

src	the source capability slot.

Definition at line 287 of file capability.h.

References L4::Cap_base::copy().

Here is the call graph for this function:

15.73.3.2 move()

```
template<typename T >
Cap L4::Cap< T >::move (
            Cap< T > const & src ) const  [inline]
```

Move a capability to this cap slot.

Parameters

src	the source capability slot.

After this operation the source slot is no longer valid.

Definition at line 277 of file capability.h.

References L4::Cap_base::move().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/cxx/capability.h

15.74 L4::Cap_base Class Reference

Base class for all kinds of capabilities.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Cap_base:

Collaboration diagram for L4::Cap_base:

Public Types

- enum No_init_type { No_init }

  Special value for uninitialized capability objects.
- enum Cap_type { Invalid = L4_INVALID_CAP }

  Invalid capability type.

Public Member Functions

- l4_cap_idx_t cap () const noexcept

  Return capability selector.
- bool is_valid () const noexcept

  Test whether the capability is a valid capability index (i.e., not L4_INVALID_CAP).
- l4_fpage_t fpage (unsigned rights=L4_CAP_FPAGE_RWS) const noexcept

  Return flex-page for the capability.
- l4_umword_t snd_base (unsigned grant=L4_MAP_ITEM_MAP, l4_cap_idx_t base=L4_INVALID_CAP) const noexcept

  Return send base.
- bool operator== (Cap_base const &o) const noexcept

  Test if two capabilities are equal.
- bool operator!= (Cap_base const &o) const noexcept

  Test if two capabilities are not equal.
- l4_msgtag_t validate (l4_utcb_t *u=l4_utcb()) const noexcept

  Check whether a capability is present (refers to an object).

- l4_msgtag_t validate (Cap< Task > task, l4_utcb_t *u=l4_utcb()) const noexcept

  Check whether a capability is present (refers to an object).
- void invalidate () noexcept

  Set this capability to invalid (L4_INVALID_CAP).

Protected Member Functions

- Cap_base (l4_cap_idx_t c) noexcept

  Generate a capability from its C representation.
- Cap_base (Cap_type cap) noexcept

  Constructor to create an invalid capability.
- Cap_base (l4_default_caps_t cap) noexcept

  Initialize capability with one of the default capabilities.
- Cap_base () noexcept

  Create an uninitialized instance.
- void move (Cap_base const &src) const

  Replace this capability with the contents of `src`.
- void copy (Cap_base const &src) const

  Copy a capability.

Protected Attributes

- l4_cap_idx_t _c

  The C representation of a capability selector.

15.74.1 Detailed Description

Base class for all kinds of capabilities.

Attention

This class is not for direct use, use L4::Cap instead.

This class contains all the things that are independent of the type of the object referred by the capability.

See also

L4::Cap for typed capabilities.

Definition at line 25 of file capability.h.

15.74.2 Member Enumeration Documentation

15.74.2.1 Cap_type

`enum L4::Cap_base::Cap_type`

Invalid capability type.

Enumerator

Invalid	Invalid capability selector.

Definition at line 43 of file capability.h.

15.74.2.2 No_init_type

```
enum L4::Cap_base::No_init_type
```

Special value for uninitialized capability objects.

Enumerator

No_init	Special value for constructing uninitialized Cap objects.

Definition at line 32 of file capability.h.

15.74.3 Constructor & Destructor Documentation

15.74.3.1 Cap_base() [1/2]

```
L4::Cap_base::Cap_base (
            l4_cap_idx_t c )  [inline], [explicit], [protected], [noexcept]
```

Generate a capability from its C representation.

Parameters

c	The C capability

Definition at line 145 of file capability.h.

15.74.3.2 Cap_base() [2/2]

```
L4::Cap_base::Cap_base (
            l4_default_caps_t cap )  [inline], [explicit], [protected], [noexcept]
```

Initialize capability with one of the default capabilities.

Parameters

cap	Capability.

Definition at line 156 of file capability.h.

15.74.4 Member Function Documentation

15.74.4.1 cap()

```
l4_cap_idx_t L4::Cap_base::cap ( ) const  [inline], [noexcept]
```

Return capability selector.

Returns

Capability selector.

Examples

examples/libs/l4re/streammap/client.cc.

Definition at line 52 of file capability.h.

Referenced by L4Re::Util::Br_manager::alloc_buffer_demand(), L4vbus::Vbus::assign_dma_domain(), L4::Icu::bind(), L4::Irq_mux::chain(), L4Re::Smart_count_cap< Unmap_flags >::copy(), L4Re::Util::Smart_count_cap< Unmap_flags >::copy(), L4Re::Util::Smart_cap_auto< Unmap_flags >::free(), L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process(), L4Re::Util::Br_manager::realloc_rcv_cap(), L4::Icu::unbind(), and L4vbus::Icu::vicu().

Here is the caller graph for this function:

15.74.4.2 copy()

```
void L4::Cap_base::copy (
            Cap_base const & src ) const  [inline], [protected]
```

Copy a capability.

Parameters

src	the source capability.

After this operation this capability refers to the same object as `src`.

Definition at line 188 of file capability.h.

References fpage(), is_valid(), L4_BASE_TASK_CAP, L4_CAP_FPAGE_RWSD, L4_FPAGE_C_OBJ_RIGHTS, and l4_task_map().

Referenced by L4::Cap< T >::copy().

Here is the call graph for this function:

Here is the caller graph for this function:

15.74.4.3 fpage()

```
l4_fpage_t L4::Cap_base::fpage (
            unsigned rights = L4_CAP_FPAGE_RWS ) const  [inline], [noexcept]
```

Return flex-page for the capability.

Parameters

rights	Rights, defaults to 'rws'

Returns

flex-page

Definition at line 72 of file capability.h.

References l4_obj_fpage().

Referenced by copy(), and move().

Here is the call graph for this function:

Here is the caller graph for this function:

15.74.4.4 is_valid()

```
bool L4::Cap_base::is_valid ( ) const  [inline], [noexcept]
```

Test whether the capability is a valid capability index (i.e., not L4_INVALID_CAP).

Returns

True if capability is not invalid, false if invalid

Examples

examples/clntsrv/client.cc, examples/libs/l4re/c++/mem_alloc/ma+rm.cc, examples/libs/l4re/c++/shared_ds/ds_clnt.cc, and examples/libs/l4re/streammap/client.cc.

Definition at line 60 of file capability.h.

Referenced by copy(), L4Re::Smart_count_cap< Unmap_flags >::copy(), L4Re::Util::Smart_cap_auto< Unmap_flags >::free(), L4Re::Util::Smart_cap_auto< Unmap_flags >::invalidate(), move(), and L4Re::Video::View::valid().

Here is the caller graph for this function:

15.74.4.5 move()

```
void L4::Cap_base::move (
            Cap_base const & src ) const  [inline], [protected]
```

Replace this capability with the contents of `src`.

Parameters

src	the source capability.

After the operation this capability refers to the object formerly referred to by the source capability `src`, and the source capability no longer refers to an object.

Definition at line 172 of file capability.h.

References fpage(), is_valid(), L4_BASE_TASK_CAP, L4_CAP_FPAGE_RWSD, L4_FPAGE_C_OBJ_RIGHTS, L4_MAP_ITEM_GRANT, and l4_task_map().

Referenced by L4::Cap< T >::move().

Here is the call graph for this function:

Here is the caller graph for this function:

15.74.4.6 snd_base()

```
l4_umword_t L4::Cap_base::snd_base (
            unsigned grant = L4_MAP_ITEM_MAP,
            l4_cap_idx_t base = L4_INVALID_CAP ) const  [inline], [noexcept]
```

Return send base.

Parameters

grant	Indicates if object shall be granted. Allowed values: L4_MAP_ITEM_MAP, L4_MAP_ITEM_GRANT.
base	Base capability (first in a bundle of aligned capabilities)

Returns

Map object.

Definition at line 84 of file capability.h.

References L4_INVALID_CAP, and l4_map_obj_control().

Here is the call graph for this function:

15.74.4.7 validate() [1/2]

```
l4_msgtag_t L4::Cap_base::validate (
            Cap< Task > task,
            l4_utcb_t * u = l4_utcb() ) const  [inline], [noexcept]
```

Check whether a capability is present (refers to an object).

Parameters

task	Task to check the capability in.
u	UTCB to be used for this operation, usually the UTCB of the calling thread.

Return values

tag.label()	> 0 Capability is present (refers to an object).
tag.label()	== 0 No capability present (void object or invalid capability slot).

A capability is considered present when it refers to an existing kernel object.

Definition at line 83 of file capability.

15.74.4.8 validate() [2/2]

```
l4_msgtag_t L4::Cap_base::validate (
            l4_utcb_t * u = l4_utcb() ) const  [inline], [noexcept]
```

Check whether a capability is present (refers to an object).

Parameters

u	UTCB to be used for this operation, usually the UTCB of the calling thread.

Return values

tag.label()	> 0 Capability is present (refers to an object).
tag.label()	== 0 No capability present (void object or invalid capability slot).

A capability is considered present when it refers to an existing kernel object.

Definition at line 90 of file capability.

The documentation for this class was generated from the following files:

- l4/sys/cxx/capability.h
- l4/sys/capability

15.75 L4::Com_error Class Reference

Error conditions during IPC.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Com_error:

Collaboration diagram for L4::Com_error:

Public Member Functions

- Com_error (long err) throw ()

  Create a Com_error for the given L4 IPC error code.

Additional Inherited Members

15.75.1 Detailed Description

Error conditions during IPC.

This exception encapsulates all IPC error conditions of L4 IPC.

Definition at line 274 of file exceptions.

15.75.2 Constructor & Destructor Documentation

15.75.2.1 Com_error()

```
L4::Com_error::Com_error (
            long err ) throw ( )   [inline], [explicit]
```

Create a Com_error for the given L4 IPC error code.

Parameters

err	The L4 IPC error code (l4_ipc... return value).

Definition at line 281 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

15.76 L4::Debugger Class Reference

C++ kernel debugger API.

Inheritance diagram for L4::Debugger:

Collaboration diagram for L4::Debugger:

Public Member Functions

- l4_msgtag_t set_object_name (const char ∗name, l4_utcb_t ∗utcb=l4_utcb()) noexcept

  Set the name of a kernel object.
- unsigned long global_id (l4_utcb_t ∗utcb=l4_utcb()) noexcept

  Get the globally unique ID of the object behind a capability.
- unsigned long kobj_to_id (l4_addr_t kobjp, l4_utcb_t ∗utcb=l4_utcb()) noexcept

  Get the globally unique ID of the object behind the kobject pointer.
- long query_log_typeid (const char ∗name, unsigned idx, l4_utcb_t ∗utcb=l4_utcb()) noexcept

  Query the log-id for a log type.
- long query_log_name (unsigned idx, char ∗name, unsigned namelen, char ∗shortname, unsigned shortnamelen, l4_utcb_t ∗utcb=l4_utcb()) noexcept

  Query the name of a log type given the ID.
- l4_msgtag_t switch_log (const char ∗name, unsigned on_off, l4_utcb_t ∗utcb=l4_utcb()) noexcept

Set or unset log.

- l4_msgtag_t get_object_name (unsigned id, char *name, unsigned size, l4_utcb_t *utcb=l4_utcb()) noexcept

  Get name of object with Id `id`.

Additional Inherited Members

15.76.1 Detailed Description

C++ kernel debugger API.

Attention

This API is subject to change! Do not rely on it in production code.

This API is to be used for debugging exclusively.

This is the API for accessing kernel-debugger functionality from user-level programs. Specifically, it provides functionality to enrich the kernel debugger with insights into the program. The purpose is to facilitate debugging with the kernel debugger. For instance, a developer might choose to name the threads of her program so that she can find them in the kernel debugger thread list.

This API interacts with a kernel object that interfaces with the kernel debugger, the jdb-kernel object. The jdb-kernel object is fix and only available when the kernel debugger is built into the microkernel. The developer needs to pass the capability through to her program.

Include File

```
#include <l4/sys/debugger>
```

Definition at line 53 of file debugger.

15.76.2 Member Function Documentation

15.76.2.1 get_object_name()

```
l4_msgtag_t L4::Debugger::get_object_name (
            unsigned id,
            char * name,
            unsigned size,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Get name of object with Id `id`.

Parameters

	id	Id of the object whose name is asked.
out	*name*	Buffer to copy the name into. The buffer must be allocated by the caller.
	size	Length of the `name` buffer.
	utcb	The UTCB to use for the operation.

Returns

Syscall return tag

Definition at line 159 of file debugger.

15.76.2.2 global_id()

```
unsigned long L4::Debugger::global_id (
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Get the globally unique ID of the object behind a capability.

Parameters

utcb	The UTCB to use for the operation.

Return values

~0UL	The capability is invalid.
>=0	The global debugger id.

Definition at line 82 of file debugger.

15.76.2.3 kobj_to_id()

```
unsigned long L4::Debugger::kobj_to_id (
            l4_addr_t kobjp,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Get the globally unique ID of the object behind the kobject pointer.

Parameters

kobjp	Kobject pointer
utcb	The UTCB to use for the operation.

Return values

~0UL	The capability or the Kobject pointer are invalid.
>=0	The globally unique id.

Definition at line 94 of file debugger.

15.76.2.4 query_log_name()

```
long L4::Debugger::query_log_name (
            unsigned idx,
            char * name,
            unsigned namelen,
            char * shortname,
            unsigned shortnamelen,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Query the name of a log type given the ID.

Parameters

	idx	ID to query.
out	*name*	Buffer to copy name to. The buffer must be allocated by the caller.
	namelen	Buffer length of name.
out	*shortname*	Buffer to copy `shortname` to. The buffer must be allocated by the caller.
	shortnamelen	Buffer length of `shortname`.
	utcb	The UTCB to use for the operation.

Return values

0	Success
<0	Error

Definition at line 127 of file debugger.

15.76.2.5 query_log_typeid()

```
long L4::Debugger::query_log_typeid (
            const char * name,
            unsigned idx,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Query the log-id for a log type.

Parameters

name	Name to query for.
idx	Idx to start searching, start with 0
utcb	The UTCB to use for the operation.

Return values

>=0	Id
<0	Error

Definition at line 108 of file debugger.

15.76.2.6 set_object_name()

```
l4_msgtag_t L4::Debugger::set_object_name (
            const char * name,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Set the name of a kernel object.

Parameters

name	Name
utcb	The UTCB to use for the operation.

Returns

System call return tag.

Definition at line 70 of file debugger.

15.76.2.7 switch_log()

```
l4_msgtag_t L4::Debugger::switch_log (
            const char * name,
            unsigned on_off,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Set or unset log.

Parameters

name	Name of the log type.
on_off	1: turn log on, 0: turn log off
utcb	The UTCB to use for the operation.

Returns

Syscall return tag

Definition at line 144 of file debugger.

The documentation for this class was generated from the following file:

- l4/sys/debugger

15.77 L4::Element_already_exists Class Reference

Exception for duplicate element insertions.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element_already_exists:

Collaboration diagram for L4::Element_already_exists:

Additional Inherited Members

15.77.1 Detailed Description

Exception for duplicate element insertions.

Definition at line 203 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

15.78 L4::Element_not_found Class Reference

Exception for a failed lookup (element not found).

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element_not_found:

Collaboration diagram for L4::Element_not_found:

Additional Inherited Members

15.78.1 Detailed Description

Exception for a failed lookup (element not found).

Definition at line 231 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

15.79 L4::Epiface Struct Reference

Base class for interface implementations.

Inheritance diagram for L4::Epiface:

Collaboration diagram for L4::Epiface:

Public Types

- typedef lpc_svr::Server_iface Server_iface

  Type for abstract server interface.
- typedef lpc_svr::Server_iface::Demand Demand

  Type for server-side receive buffer demand.

Public Member Functions

- Epiface ()

  Make a server object.
- virtual l4_msgtag_t dispatch (l4_msgtag_t tag, unsigned rights, l4_utcb_t ∗utcb)=0

  The abstract handler for client requests to the object.
- virtual Demand get_buffer_demand () const =0

  Get the server-side receive buffer demand for this object.
- virtual ∼Epiface ()=0

  Destroy the object.
- Stored_cap obj_cap () const

  Get the capability to the kernel object belonging to this object.
- Server_iface ∗ server_iface () const

  Get pointer to server interface at which the object is currently registered.
- int set_server (Server_iface ∗srv, Cap< void > cap, bool managed=false)

  Set server registration info for the object.
- void set_obj_cap (Cap< void > const &cap)

  Deprecated server registration function.

15.79.1 Detailed Description

Base class for interface implementations.

An Epiface is the base interface of objects registered in the server loop. Incomming IPC gets dispatched to the appropriate Epiface object where the call is then handled appropriately.

Note

Server loops are allowed to internally keep raw pointers to Epiface objects for dispatching calls. Instances must therefore never be copied or moved.

Definition at line 153 of file ipc_epiface.

15.79.2 Member Function Documentation

15.79.2.1 dispatch()

```
virtual l4_msgtag_t L4::Epiface::dispatch (
            l4_msgtag_t tag,
            unsigned rights,
            l4_utcb_t * utcb )  [pure virtual]
```

The abstract handler for client requests to the object.

Parameters

tag	The message tag for this invocation.
rights	The rights bits in the invoked capability.
utcb	The UTCB used for the invocation.

Return values

-L4_ENOREPLY	No reply message is send.
<0	Error, reply with error code.
>=0	Success, reply with return value.

This function must be implemented by application specific server objects.

Implemented in L4::Irqep_t< Derived, BASE, bool >, L4::Irqep_t< Irq_object >, L4::Epiface_t< Derived, IFACE, BASE, bool >, L4::Epiface_t< Block_dev< Ds_data >, L4virtio::Device >, L4::Epiface_t< Null_handler, L4::Kobject >, and L4::Server_object.

15.79.2.2 get_buffer_demand()

```
virtual Demand L4::Epiface::get_buffer_demand ( ) const  [pure virtual]
```

Get the server-side receive buffer demand for this object.

Note

This function is usually not implemented directly, but by using Server_object_t template with an IPC interface definition.

Returns

The needed server-side receive buffers for this object

Implemented in L4::Epiface_t0< RPC_IFACE, BASE >, L4::Epiface_t0< void, Epiface >, L4::Epiface_t0< IFACE, L4::Epiface >, L4::Server_object_t< IFACE, BASE >, L4::Server_object_t< Kobject >, and L4::Server_object_t< IFACE, L4::Server_object >.

15.79.2.3 obj_cap()

```
Stored_cap L4::Epiface::obj_cap ( ) const  [inline]
```

Get the capability to the kernel object belonging to this object.

Returns

Capability for the kernel object behind the server.

This is usually either an Ipc_gate or an Irq.

Definition at line 214 of file ipc_epiface.

Referenced by L4Re::Util::Object_registry::unregister_obj().

Here is the caller graph for this function:

15.79.2.4 server_iface()

```
Server_iface* L4::Epiface::server_iface ( ) const  [inline]
```

Get pointer to server interface at which the object is currently registered.

Returns

Pointer to the server at which the object is currently registered, NULL if the object is not registered at any server.

Definition at line 221 of file ipc_epiface.

15.79.2.5 set_server()

```
int L4::Epiface::set_server (
            Server_iface * srv,
            Cap< void > cap,
            bool managed = false )  [inline]
```

Set server registration info for the object.

Parameters

srv	The server to register at
cap	The capability that connects the object.
managed	Mark the capability as managed or unmanaged. Typical server implementations use this flag to remember whether the capability was internally allocated or not.

Returns

0 on success, -L4_EINVAL if the srv and cap are not consistent.

Definition at line 232 of file ipc_epiface.

References L4_EINVAL.

Referenced by L4Re::Util::Object_registry::unregister_obj().

Here is the caller graph for this function:

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_epiface

15.80 L4::Epiface_t< Derived, IFACE, BASE, bool > Struct Template Reference

Epiface implementation for Kobject-based interface implementations.

Inheritance diagram for L4::Epiface_t< Derived, IFACE, BASE, bool >:

Collaboration diagram for L4::Epiface_t< Derived, IFACE, BASE, bool >:

Public Member Functions

- l4_msgtag_t dispatch (l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb) final

  The abstract handler for client requests to the object.

Additional Inherited Members

15.80.1 Detailed Description

template<typename Derived, typename IFACE, typename BASE = L4::Epiface, bool = cxx::is_polymorphic<BASE>::value>
struct L4::Epiface_t< Derived, IFACE, BASE, bool >

Epiface implementation for Kobject-based interface implementations.

Template Parameters

Derived	Class providing the interface implementations.
BASE	Epiface base class.

Examples

examples/clntsrv/server.cc.

Definition at line 510 of file ipc_epiface.

15.80.2 Member Function Documentation

15.80.2.1 dispatch()

```
template<typename Derived , typename IFACE , typename BASE = L4::Epiface, bool = cxx::is_↩
polymorphic<BASE>::value>
l4_msgtag_t L4::Epiface_t< Derived, IFACE, BASE, bool >::dispatch (
            l4_msgtag_t tag,
            unsigned rights,
            l4_utcb_t * utcb )  [inline], [final], [virtual]
```

The abstract handler for client requests to the object.

Parameters

tag	The message tag for this invocation.
rights	The rights bits in the invoked capability.
utcb	The UTCB used for the invocation.

Return values

-L4_ENOREPLY	No reply message is send.
<0	Error, reply with error code.
>=0	Success, reply with return value.

This function must be implemented by application specific server objects.

Implements L4::Epiface.

Definition at line 513 of file ipc_epiface.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_epiface

15.81 L4::Epiface_t0< RPC_IFACE, BASE > Struct Template Reference

Epiface mixin for generic Kobject-based interfaces.

Inheritance diagram for L4::Epiface_t0< RPC_IFACE, BASE >:

Collaboration diagram for L4::Epiface_t0< RPC_IFACE, BASE >:

Public Types

- typedef RPC_IFACE Interface

  Data type of the IPC interface definition.

Public Member Functions

- Type_info::Demand get_buffer_demand () const

  Get the server-side buffer demand based in IFACE.
- Cap< RPC_IFACE > obj_cap () const

  Get the (typed) capability to this object.

15.81.1 Detailed Description

template<typename RPC_IFACE, typename BASE = Epiface>
struct L4::Epiface_t0< RPC_IFACE, BASE >

Epiface mixin for generic Kobject-based interfaces.

Template Parameters

RPC_IFACE	Data type of the IPC interface definition.
BASE	Base Epiface class.

Definition at line 264 of file ipc_epiface.

15.81.2 Member Function Documentation

15.81.2.1 obj_cap()

```
template<typename RPC_IFACE , typename BASE = Epiface>
Cap<RPC_IFACE> L4::Epiface_t0< RPC_IFACE, BASE >::obj_cap ( ) const  [inline]
```

Get the (typed) capability to this object.

Returns

Capability for the kernel object behind the server.

Definition at line 277 of file ipc_epiface.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_epiface

15.82 L4::Exception Class Reference

Exception interface.

Inherits L4::Kobject_0t< Derived, PROTO, S_DEMAND >.

Collaboration diagram for L4::Exception:

Public Types

- typedef L4::Typeid::Rpc_nocode< exception_t > Rpcs

  Exception call.

15.82.1 Detailed Description

Exception interface.

This class defines the interface for handling exception IPC. When an exception occurs during program execution, for example due to a division by zero, the kernel will synthesise an exception IPC and send it to the thread's exception handler, who can then handle it.

The exception handler is set with the L4::Thread::control interface.

Definition at line 42 of file exception.

15.82.2 Member Typedef Documentation

15.82.2.1 Rpcs

```
typedef L4::Typeid::Rpc_nocode<exception_t> L4::Exception::Rpcs
```

Exception call.

Parameters

	regs	Register state of the faulting thread.
	rwin	Receive window in the address space.
out	*fp*	Optional flex-page to resolve the exception.

Returns

Message tag containing error code.

Definition at line 62 of file exception.

The documentation for this class was generated from the following file:

- l4/sys/exception

15.83 L4::Exception_tracer Class Reference

Back-trace support for exceptions.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Exception_tracer:

Collaboration diagram for L4::Exception_tracer:

Public Member Functions

- void const ∗const ∗ pc_array () const throw ()

  Get the array containing the call trace.
- int frame_count () const throw ()

  Get the number of entries that are valid in the call trace.

Protected Member Functions

- Exception_tracer () throw ()

  Create a back trace.

15.83.1 Detailed Description

Back-trace support for exceptions.

This class holds an array of at most L4_CXX_EXCEPTION_BACKTRACE instruction pointers containing the call trace at the instant when an exception was thrown.

Definition at line 62 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

15.84 L4::Factory Class Reference

C++ Factory interface, see Factory for the C interface.

Inheritance diagram for L4::Factory:

Collaboration diagram for L4::Factory:

Data Structures

- struct Lstr

  Special type to add a pascal string into the factory create stream.
- struct Nil

  Special type to add a void argument into the factory create stream.
- class S

  Stream class for the create() argument stream.

Public Member Functions

- S create (Cap< void > target, long obj, l4_utcb_t *utcb=l4_utcb()) noexcept

  Generic create call to the factory.

- template<typename OBJ >
  S create (Cap< OBJ > target, l4_utcb_t *utcb=l4_utcb()) noexcept

  Create call for typed capabilities.
- l4_msgtag_t create_task (Cap< Task > const &target_cap, l4_fpage_t const &utcb_area, l4_utcb_t *utcb=l4_utcb()) noexcept

  Create a new task.
- l4_msgtag_t create_factory (Cap< Factory > const &target_cap, unsigned long limit, l4_utcb_t *utcb=l4_utcb()) noexcept

  Create a new factory.
- l4_msgtag_t create_gate (Cap< void > const &target_cap, Cap< Thread > const &thread_cap, l4_umword_t label, l4_utcb_t *utcb=l4_utcb()) noexcept

  Create a new IPC gate.

Additional Inherited Members

15.84.1 Detailed Description

C++ Factory interface, see Factory for the C interface.

Factories provide an interface to create objects which are accessed via capabilities.

For additional information about which objects can be created via this interface, see server-specific information in Kernel Factory and L4Re Servers.

Include File

```
#include <l4/sys/factory>
```

For the C interface refer to Factory.

Definition at line 48 of file factory.

15.84.2 Member Function Documentation

15.84.2.1 create() [1/2]

```
template<typename OBJ >
S L4::Factory::create (
            Cap< OBJ > target,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Create call for typed capabilities.

Template Parameters

OBJ	Capability type of the object to be created.

Parameters

out	*target*	Capability of type OBJ.
	utcb	UTCB to use.

Returns

A create stream that allows additional arguments to be passed to the create() call via the left-shift (<<) operator.

This method does not directly invoke the factory. The factory is invoked when the create stream returned by this method is converted to an `l4_msgtag_t`, or otherwise when the stream goes out of scope.

Note

The create stream uses the UTCB to store parameters for the service call. During the lifetime of a create stream or, until it is converted to a `l4_msgtag_t`, other UTCB-using operations must not be used.

Usage:

```
L4::Cap<L4Re::Dataspace> ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
factory->create(ds) « l4_mword_t(size_in_bytes);
```

Definition at line 308 of file factory.

References L4::Kobject::cap().

Here is the call graph for this function:

15.84.2.2 create() [2/2]

```
S L4::Factory::create (
            Cap< void > target,
            long obj,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Generic create call to the factory.

Parameters

out	*target*	Capability selector for the new object. The caller must allocate the capability slot. The kernel stores the new objects's capability into this slot.
	obj	The protocol ID that specifies which kind of object shall be created.
	utcb	The UTCB to use for the operation.

Returns

A create stream that allows additional arguments to be passed to the create() call via the left-shift (<<) operator.

This method does not directly invoke the factory. The factory is invoked when the create stream returned by this method is converted to an `l4_msgtag_t`, or otherwise when the stream goes out of scope.

Note

The create stream uses the UTCB to store parameters for the service call. During the lifetime of a create stream or, until it is converted to a `l4_msgtag_t`, other UTCB-using operations must not be used.

See also

create(Cap<OBJ>, l4_utcb_t *)

Definition at line 277 of file factory.

References L4::Kobject::cap().

Referenced by L4Re::Mem_alloc::alloc().

Here is the call graph for this function:

Here is the caller graph for this function:

15.84.2.3 create_factory()

```
l4_msgtag_t L4::Factory::create_factory (
            Cap< Factory > const & target_cap,
            unsigned long limit,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Create a new factory.

Parameters

out	*target_cap*	The kernel stores the new factory's capability into this slot.
	limit	Limit for the new factory in bytes.
	utcb	The UTCB to use for the operation.

Returns

Syscall return tag

Note

In addition to memory needed for internal data structures, the `limit` (quota) of the new factory is counted towards the quota of the creating factory. The `limit` must be within `1 ≤ limit ≤ 2^(8 * sizeof(l4_umword_t) - 1) - 2` otherwise the behavior is undefined.

This method is only guaranteed to work with the Kernel Factory. For other services, use the generic create() method and consult the service documentation for information on the arguments that need to be passed to the create stream.

Definition at line 368 of file factory.

15.84.2.4 create_gate()

```
l4_msgtag_t L4::Factory::create_gate (
            Cap< void > const & target_cap,
            Cap< Thread > const & thread_cap,
            l4_umword_t label,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Create a new IPC gate.

Parameters

out	*target_cap*	The kernel stores the new IPC gate's capability into this slot.
	thread_cap	Optional capability selector of the thread to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate.
	label	Optional label of the gate (precisely used if `thread_cap` is valid). If `thread_cap` is valid, `label` must be present.
	utcb	The UTCB to use for the operation.

Returns

Syscall return tag containing one of the following return codes.

Return values

L4_EOK	No error occurred.
-L4_ENOMEM	Out-of-memory during allocation of the lpc_gate object.

Return values

-L4_ENOENT	`thread_cap` is void or points to something that is not a thread.
-L4_EPERM	No L4_CAP_FPAGE_S rights on `thread_cap`.

An unbound IPC gate can be bound to a thread using L4::Ipc_gate::bind_thread().

Note

This method is only guaranteed to work with the Kernel Factory.

See also

L4::Ipc_gate

Definition at line 402 of file factory.

15.84.2.5 create_task()

```
l4_msgtag_t L4::Factory::create_task (
            Cap< Task > const & target_cap,
            l4_fpage_t const & utcb_area,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Create a new task.

Parameters

`out`	*target_cap*	The kernel stores the new task's capability into this slot.
	utcb_area	Flexpage that describes an area in the address space of the new task, where the kernel should map the kernel-allocated kernel-user memory to. The kernel uses the kernel-user memory to store UTCBs and vCPU state-save-areas of the new task.
	utcb	The UTCB to use for the operation.

Returns

Syscall return tag

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to L4::Task::add_ku_mem / l4_task_add_ku_mem() for adding more of this type of memory.

This method is only guaranteed to work with the Kernel Factory.

See also

L4::Task

Definition at line 341 of file factory.

The documentation for this class was generated from the following file:

- l4/sys/factory

15.85 L4::Factory::Lstr Struct Reference

Special type to add a pascal string into the factory create stream.

Collaboration diagram for L4::Factory::Lstr:

Public Member Functions

- Lstr (char const ∗s, unsigned len) noexcept

Data Fields

- char const ∗ s

  The character buffer.
- unsigned len

  The number of characters in the buffer.

15.85.1 Detailed Description

Special type to add a pascal string into the factory create stream.

This encapsulates a string that has an explicit length.

Definition at line 64 of file factory.

15.85.2 Constructor & Destructor Documentation

15.85.2.1 Lstr()

```
L4::Factory::Lstr::Lstr (
            char const * s,
            unsigned len )  [inline], [noexcept]
```

Parameters

s	Pointer to the c-style string.
len	Length in number of characters of the string s.

Definition at line 80 of file factory.

The documentation for this struct was generated from the following file:

- l4/sys/factory

15.86 L4::Factory::Nil Struct Reference

Special type to add a void argument into the factory create stream.

Collaboration diagram for L4::Factory::Nil:

15.86.1 Detailed Description

Special type to add a void argument into the factory create stream.

Definition at line 57 of file factory.

The documentation for this struct was generated from the following file:

- l4/sys/factory

15.87 L4::Factory::S Class Reference

Stream class for the create() argument stream.

Collaboration diagram for L4::Factory::S:

Public Member Functions

- S (S &&o) noexcept

  Move ...
- S (l4_cap_idx_t f, long obj, L4::Cap< void > target, l4_utcb_t *utcb) noexcept

  Create a stream for a specific create() call.
- ~S () noexcept

  Commit the operation in the destructor to have a cool syntax for create().
- operator l4_msgtag_t () noexcept

  Explicitly commits the operation and returns the result.
- void put (l4_mword_t i) noexcept

  Put a single l4_mword_t as next argument.
- void put (l4_umword_t i) noexcept

  Put a single l4_umword_t as next argument.
- void put (char const *s) &noexcept

  Add a zero-terminated string as next argument.
- void put (Lstr const &s) &noexcept

  Add a pascal string as next argument.
- void put (Nil) &noexcept

  Add an empty argument.
- void put (l4_fpage_t d) &noexcept

  Add a flex page as next argument.

15.87.1 Detailed Description

Stream class for the create() argument stream.

This stream allows a variable number of arguments to be added to a create() call.

Definition at line 89 of file factory.

15.87.2 Constructor & Destructor Documentation

15.87.2.1 S() [1/2]

```
L4::Factory::S::S (
            S && o )  [inline], [noexcept]
```

Move ...

Parameters

o	Instance of S to move.

Definition at line 108 of file factory.

References l4_msgtag_t::raw.

15.87.2.2 S() [2/2]

```
L4::Factory::S::S (
            l4_cap_idx_t f,
            long obj,
            L4::Cap< void > target,
            l4_utcb_t * utcb )  [inline], [noexcept]
```

Create a stream for a specific create() call.

Parameters

	f	The capability for the factory object (L4::Factory).
	obj	The protocol ID to describe the type of the object that shall be created.
out	*target*	The capability selector for the new object. The caller must allocate the capability slot. The kernel stores the new object's capability into this slot.
	utcb	The UTCB to use for the operation.

Definition at line 132 of file factory.

15.87.3 Member Function Documentation

15.87.3.1 operator l4_msgtag_t()

```
L4::Factory::S::operator l4_msgtag_t ( ) [inline], [noexcept]
```

Explicitly commits the operation and returns the result.

Returns

The result of the create() operation.

Definition at line 152 of file factory.

15.87.3.2 put() [1/6]

```
void L4::Factory::S::put (
            char const * s ) & [inline], [noexcept]
```

Add a zero-terminated string as next argument.

Parameters

s	The string to add as next argument.

Returns

Reference to this stream.

The string will be added with the zero-terminator.

Definition at line 192 of file factory.

15.87.3.3 put() [2/6]

```
void L4::Factory::S::put (
            l4_fpage_t d ) & [inline], [noexcept]
```

Add a flex page as next argument.

Parameters

d	The flex page to add (there will be no map operation).

Returns

Reference to this stream.

Definition at line 230 of file factory.

15.87.3.4 put() [3/6]

```
void L4::Factory::S::put (
            l4_mword_t i )  [inline], [noexcept]
```

Put a single l4_mword_t as next argument.

Parameters

i	The value to add as next argument.

Returns

Reference to this stream.

Definition at line 166 of file factory.

15.87.3.5 put() [4/6]

```
void L4::Factory::S::put (
            l4_umword_t i )  [inline], [noexcept]
```

Put a single l4_umword_t as next argument.

Parameters

i	The value to add as next argument.

Returns

Reference to this stream.

Definition at line 178 of file factory.

15.87.3.6 put() [5/6]

```
void L4::Factory::S::put (
            Lstr const & s ) &  [inline], [noexcept]
```

Add a pascal string as next argument.

Parameters

s	The string to add as next argument.

Returns

Reference to this stream.

The string will be added with the exact length given. It is the responsibility of the caller to make sure that the string is zero- terminated when that is required by the server.

Definition at line 208 of file factory.

15.87.3.7 put() [6/6]

```
void L4::Factory::S::put (
            Nil  ) &  [inline], [noexcept]
```

Add an empty argument.

Returns

Reference to this stream.

Definition at line 218 of file factory.

The documentation for this class was generated from the following file:

- l4/sys/factory

15.88 L4::Icu Class Reference

C++ Icu interface, see Interrupt controller for the C interface.

Inheritance diagram for L4::Icu:

Collaboration diagram for L4::Icu:

Data Structures

- class Info

  This class encapsulates information about an ICU.

Public Member Functions

- l4_msgtag_t bind (unsigned irqnum, L4::Cap< Triggerable > irq, l4_utcb_t *utcb=l4_utcb()) noexcept

  Bind an interrupt line of an interrupt controller to an interrupt object.
- l4_msgtag_t unbind (unsigned irqnum, L4::Cap< Triggerable > irq, l4_utcb_t *utcb=l4_utcb()) noexcept

  Remove binding of an interrupt line from the interrupt controller object.

- l4_msgtag_t info (l4_icu_info_t *info, l4_utcb_t *utcb=l4_utcb()) noexcept

  Get information about the ICU features.
- l4_msgtag_t msi_info (l4_umword_t irqnum, l4_uint64_t source, l4_icu_msi_info_t *msi_info)

  Get MSI info about IRQ.
- l4_msgtag_t mask (unsigned irqnum, l4_umword_t *label=0, l4_timeout_t to=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept

  Mask an IRQ line.
- l4_msgtag_t set_mode (unsigned irqnum, l4_umword_t mode, l4_utcb_t *utcb=l4_utcb()) noexcept

  Set interrupt mode.

Additional Inherited Members

15.88.1 Detailed Description

C++ Icu interface, see Interrupt controller for the C interface.

Note

"ICU" is short for "interrupt control unit".

This class defines the interface for interrupt controllers. It defines functions for binding L4::Irq objects to interrupt lines and other interrupt sources, as well as functions for masking and unmasking of interrupts.

To setup an interrupt line the following steps are required:

1. set_mode() (optional if interrupt has a default mode)
2. L4::Rcv_endpoint::bind_thread() to attach the L4::Irq object to a thread
3. bind()
4. unmask() to receive the first interrupt

For certain interrupt sources only some of these steps are necessary and supported, see L4::Scheduler and L4::Vcon.

At most one L4::Irq object can be bound to a certain interrupt source and a certain L4::Irq object can be bound to at most one interrupt source.

Include File

```
#include <l4/sys/icu>
```

Definition at line 262 of file irq.

15.88.2 Member Function Documentation

15.88.2.1 bind()

```
l4_msgtag_t L4::Icu::bind (
            unsigned irqnum,
            L4::Cap< Triggerable > irq,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Bind an interrupt line of an interrupt controller to an interrupt object.

Parameters

irqnum	IRQ line at the ICU.
irq	IRQ object for the given IRQ line to bind to this ICU.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag. The caller should check the return value using l4_error() to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of `0` means a direct unmask via the IRQ object using L4::Irq::unmask. A return value of `1` means that the interrupt has to be unmasked via the ICU using L4::Icu::unmask.

Return values

L4_EINVAL	`irq` is bound to an interrupt source.
L4_EPERM	The ICU instance requires L4_CAP_FPAGE_W on `irq` and L4_CAP_FPAGE_W was not present.

In case the `irq` is already bound to an interrupt source, it is unbound first. In case the `irq` is bound and the interrupt source is bound to a different L4::Irq object, only the unbinding happens. An L4::Irq object that is bound to an interrupt source will get unbound if the L4::Irq object is deleted.

Definition at line 321 of file irq.

References L4::Cap_base::cap(), L4::Kobject::cap(), and l4_icu_bind_u().

Referenced by L4virtio::Driver::Device::bind_notification_irq(), and L4virtio::Driver::Device::driver_connect().

Here is the call graph for this function:

Here is the caller graph for this function:

15.88.2.2 info()

```
l4_msgtag_t L4::Icu::info (
            l4_icu_info_t * info,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Get information about the ICU features.

Parameters

out	*info*	Info structure to be filled with information.
	utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 356 of file irq.

References L4::Kobject::cap(), and l4_icu_info_u().

Here is the call graph for this function:

15.88.2.3 mask()

```
l4_msgtag_t L4::Icu::mask (
            unsigned irqnum,
            l4_umword_t * label = 0,
            l4_timeout_t to = L4_IPC_NEVER,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Mask an IRQ line.

Parameters

irqnum	IRQ line at the ICU.
label	If NULL this function is a send-only message to the ICU. If not NULL this function will enter an open wait after sending the mask message.
to	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL `label` only.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 399 of file irq.

References L4::Kobject::cap(), and l4_icu_mask_u().

Here is the call graph for this function:

15.88.2.4 msi_info()

```
l4_msgtag_t L4::Icu::msi_info (
            l4_umword_t irqnum,
            l4_uint64_t source,
            l4_icu_msi_info_t * msi_info )
```

Get MSI info about IRQ.

Parameters

	irqnum	IRQ line at the ICU.
	source	Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification.
out	*msi_info*	A l4_icu_msi_info_t structure receiving the address and the data value to trigger this MSI.

Returns

Syscall return tag

15.88.2.5 set_mode()

```
l4_msgtag_t L4::Icu::set_mode (
            unsigned irqnum,
            l4_umword_t mode,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Set interrupt mode.

Parameters

irqnum	IRQ line at the ICU.
mode	Mode, see L4_irq_mode.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 427 of file irq.

References L4::Kobject::cap(), and l4_icu_set_mode_u().

Here is the call graph for this function:

15.88.2.6 unbind()

```
l4_msgtag_t L4::Icu::unbind (
            unsigned irqnum,
            L4::Cap< Triggerable > irq,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Remove binding of an interrupt line from the interrupt controller object.

Parameters

irqnum	IRQ line at the ICU.
irq	IRQ object to remove from the ICU.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 339 of file irq.

References L4::Cap_base::cap(), L4::Kobject::cap(), and l4_icu_unbind_u().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/irq

15.89 L4::Icu::Info Class Reference

This class encapsulates information about an ICU.

Inheritance diagram for L4::Icu::Info:

Collaboration diagram for L4::Icu::Info:

Public Member Functions

- bool supports_msi () const noexcept

  True, if the ICU has support for MSIs.

Additional Inherited Members

15.89.1 Detailed Description

This class encapsulates information about an ICU.

Definition at line 289 of file irq.

The documentation for this class was generated from the following file:

- l4/sys/irq

15.90 L4::Invalid_capability Class Reference

Indicates that an invalid object was invoked.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Invalid_capability:

Collaboration diagram for L4::Invalid_capability:

Public Member Functions

- Invalid_capability (Cap< void > const &o) throw ()

  Create an Invalid_object exception for the Object o.
- char const * str () const throw ()

  Return a human readable string for the exception.
- Cap< void > const & cap () const throw ()

  Get the object that caused the error.

Additional Inherited Members

15.90.1 Detailed Description

Indicates that an invalid object was invoked.

An Object is invalid if it has L4_INVALID_ID as server L4 UID, or if the server does not know the object ID.

Definition at line 245 of file exceptions.

15.90.2 Constructor & Destructor Documentation

15.90.2.1 Invalid_capability()

```
L4::Invalid_capability::Invalid_capability (
            Cap< void > const & o ) throw ( )   [inline], [explicit]
```

Create an Invalid_object exception for the Object o.

Parameters

o	The object that caused the server side error.

Definition at line 255 of file exceptions.

15.90.3 Member Function Documentation

15.90.3.1 cap()

```
Cap<void> const& L4::Invalid_capability::cap ( ) const throw ( )   [inline]
```

Get the object that caused the error.

Returns

The object that caused the error on invocation.

Definition at line 264 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

15.91 L4::Io_pager Class Reference

Io_pager interface.

Inheritance diagram for L4::Io_pager:

Collaboration diagram for L4::Io_pager:

Public Member Functions

- l4_msgtag_t io_page_fault (l4_fpage_t io_pfa, l4_umword_t pc, L4::Ipc::Rcv_fpage rwin, L4::Ipc::Opt< L4::Ipc::Snd_fpage & > fp)

  IO page fault protocol message.

15.91.1 Detailed Description

Io_pager interface.

Note

This interface is IA32 specific.

This class defines the interface for handling IO page faults. IO page faults happen when a thread tries to access an IO port that it does not currently have access to.

Depending on the microkernel's implementation, IO page faults can be handled in two ways.

If the microkernel does not support IO page faults, this IO pagefault interface is not used. Instead, the microkernel sends an exception IPC to the thread's exception handler (L4::Exception), indicating a `GP` (exception number 13). The exception handler must consult the faulting instruction to determine the cause of the exception. This is the default in Fiasco.OC.

In contrast, if the microkernel supports IO page faults, the microkernel will generate an IO page fault message and send it to the thread's page fault handler (pager). The page fault handler can implement this interface to handle the IO page faults.

Note

A program may use this mechanism to implement a lazy IO port access scheme.

The page fault and exception handlers are set with the L4::Thread::control interface.

Definition at line 61 of file pager.

15.91.2 Member Function Documentation

15.91.2.1 io_page_fault()

```
l4_msgtag_t L4::Io_pager::io_page_fault (
            l4_fpage_t io_pfa,
            l4_umword_t pc,
            L4::Ipc::Rcv_fpage rwin,
            L4::Ipc::Opt< L4::Ipc::Snd_fpage & > fp )
```

IO page fault protocol message.

Parameters

	io_pfa	Flex-page describing the faulting IO-port.
	pc	Faulting program counter.
	rwin	The receive window for a flex-page mapping.
out	*fp*	Optional: flex-page descriptor to send to the task raising the page fault.

Returns

System call message tag; use l4_error() to check for errors.

IO-port fault messages are usually generated by the kernel and an IO-page-fault handler needs to be in place to handle such faults and generate a reply, potentially filling in `fp`.

The documentation for this class was generated from the following file:

- l4/sys/pager

15.92 L4::Iommu Class Reference

Interface for IO-MMUs used for DMA remapping.

Inheritance diagram for L4::Iommu:

Collaboration diagram for L4::Iommu:

Public Member Functions

- l4_msgtag_t bind (l4_uint64_t src_id, Ipc::Cap< Task > dma_space)

  Associate `dma_space` *with the set of device(s) specified by* `src_id`*.*
- l4_msgtag_t unbind (l4_uint64_t src_id, Ipc::Cap< Task > dma_space)

  Remove the association of the given DMA address space from the device(s) specified by `src_id`*.*

15.92.1 Detailed Description

Interface for IO-MMUs used for DMA remapping.

This interface allows to associate a DMA address space with a platform dependent set of devices.

Definition at line 16 of file iommu.

15.92.2 Member Function Documentation

15.92.2.1 bind()

```
l4_msgtag_t L4::Iommu::bind (
            l4_uint64_t src_id,
            Ipc::Cap< Task > dma_space )
```

Associate `dma_space` with the set of device(s) specified by `src_id`.

Parameters

src_id	Platform dependent source ID specifying the set of devices that shall use `dma_space` for DMA remapping.
dma_space	The DMA space (L4::Task created with L4_PROTO_DMA_SPACE) providing the mappings that shall be used for the device(s).

15.92.2.2 unbind()

```
l4_msgtag_t L4::Iommu::unbind (
            l4_uint64_t src_id,
            Ipc::Cap< Task > dma_space )
```

Remove the association of the given DMA address space from the device(s) specified by `src_id`.

Parameters

src_id	Platform dependent source ID specifying the set of devices that shall no longer use `dma_space` for DMA remapping.
dma_space	The DMA space formerly associated with bind().

The documentation for this class was generated from the following file:

- l4/sys/iommu

15.93 L4::IOModifier Class Reference

Modifier class for the IO stream.

Collaboration diagram for L4::IOModifier:

15.93.1 Detailed Description

Modifier class for the IO stream.

An IO Modifier can be used to change properties of an IO stream for example the number format.

Definition at line 33 of file basic_ostream.

The documentation for this class was generated from the following file:

- l4/cxx/basic_ostream

15.94 L4::Ipc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference

Array data type for dynamically sized arrays in RPCs.

Inheritance diagram for L4::Ipc::Array< ELEM_TYPE, LEN_TYPE >:

Collaboration diagram for L4::Ipc::Array< ELEM_TYPE, LEN_TYPE >:

Public Member Functions

- Array ()

  Make array.
- Array (LEN_TYPE length, ELEM_TYPE *data)

  Make array from length and data pointer.
- Array (typename Non_const< ELEM_TYPE >::type const &other)

  Make a const array from a non-const array.

15.94.1 Detailed Description

template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
struct L4::Ipc::Array< ELEM_TYPE, LEN_TYPE >

Array data type for dynamically sized arrays in RPCs.

Template Parameters

ELEM_TYPE	The data type of an array element, should be 'const' when used as input.
LEN_TYPE	Data type used to store the number of elements in the array.

An Array generally encapsulates a data pointer and a length (number of elements). Array does *not* provide any storage for the data itself. The storage is either provided by a client-side caller or in the case of Array_ref is the message itself.

Arrays can be used as input or as output arguments, when used as input ELEM_TYPE should be qualified *const*, when used as output a reference to an array must be used and the ELEM_TYPE must *not* be qualified *const*. It is the caller's responsibility to provide an array buffer of sufficient length. If a message from the server is too large it will be silently truncated.

If backward compatibility with Ipc::Stream is required, then LEN_TYPE must be `unsigned long`.

Definition at line 85 of file ipc_array.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

15.95 L4::Ipc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference

Server-side copy in buffer for Array.

Collaboration diagram for L4::Ipc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >:

Public Member Functions

- void copy_in (const_array a)

  copy in data from a source array
- Array_in_buf (const_array a)

  Make Array_in_buf from a const array.
- Array_in_buf (array a)

  Make Array_in_buf from a non-const array.

Data Fields

- ELEM_TYPE data [MAX]

  The data elements.
- LEN_TYPE length

  The length of the array.

15.95.1 Detailed Description

template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default, LEN_TYPE MAX = (L4_UTCB_GENERIC_DATA_SIZE * sizeof(l4_umword_t)) / sizeof(ELEM_TYPE)>
struct L4::Ipc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >

Server-side copy in buffer for Array.

Template Parameters

ELEM_TYPE	Data type of an array element.
LEN_TYPE	Data type for the number of elements in the array.
MAX	The maximum number of elements in the buffer. If the actual message is longer than the buffer, it will be silently truncated.

This type is assignment compatible to Array_ref<ELEM_TYPE, LEN_TYPE> and provides a transparent server-side copy-in mechanism for array parameters. The Array_in_buf provides the storage for the array data and receives a copy of the data passed to the server-function.

Definition at line 123 of file ipc_array.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

15.96 L4::Ipc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference

Array reference data type for arrays located in the message.

Inheritance diagram for L4::Ipc::Array_ref< ELEM_TYPE, LEN_TYPE >:

Collaboration diagram for L4::Ipc::Array_ref< ELEM_TYPE, LEN_TYPE >:

15.96.1 Detailed Description

template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
struct L4::Ipc::Array_ref< ELEM_TYPE, LEN_TYPE >

Array reference data type for arrays located in the message.

Note

Use Array for normal RPC interfaces, Array_ref is usually used as server-side argument, see Array.

Template Parameters

ELEM_TYPE	The data type of an array element, should be 'const' when used as input.
LEN_TYPE	Data type used to store the number of elements in the array.

Definition at line 39 of file ipc_array.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

15.97 L4::Ipc::As_value< T > Struct Template Reference

Pass the argument as plain data value.

Collaboration diagram for L4::Ipc::As_value< T >:

15.97.1 Detailed Description

template<typename T>
struct L4::Ipc::As_value< T >

Pass the argument as plain data value.

Template Parameters

T	The type of the original argument.

As_value<T> is used when *T* would be otherwise interpreted specially, for example as flex page. When using As_value<> then the argument is transmitted as plain data element.

Definition at line 127 of file ipc_types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_types

15.98 L4::Ipc::Buf_item Class Reference

RPC warpper for a receive item.

Collaboration diagram for L4::Ipc::Buf_item:

15.98.1 Detailed Description

RPC warpper for a receive item.

Definition at line 305 of file ipc_types.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_types

15.99 L4::Ipc::Call Struct Reference

RPC attribute for a standard RPC call.

Inheritance diagram for L4::Ipc::Call:

Collaboration diagram for L4::Ipc::Call:

15.99.1 Detailed Description

RPC attribute for a standard RPC call.

This is the default for the *FLAGS* parameter for L4::Ipc::Msg::Rpc_call L4::Ipc::Msg::Rpc_inline_call templates and declares the RPC to have default call semantics and timeouts.

Examples:
```
L4_RPC(long, send, (unsigned value), L4::Ipc::Call);
```

which is equivalent to:
```
L4_RPC(long, send, (unsigned value));
```

Definition at line 226 of file ipc_iface.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_iface

15.100 L4::Ipc::Call_t< RIGHTS > Struct Template Reference

RPC attribute for an RPC call with required rights.

Inheritance diagram for L4::Ipc::Call_t< RIGHTS >:

Collaboration diagram for L4::Ipc::Call_t< RIGHTS >:

15.100.1 Detailed Description

template<unsigned RIGHTS>
struct L4::Ipc::Call_t< RIGHTS >

RPC attribute for an RPC call with required rights.

Template Parameters

RIGHTS	The capability rights required for this call. L4_CAP_FPAGE_W and L4_CAP_FPAGE_S are checked within the server (and -L4_EPERM shall be returned if the caller has insufficient rights). L4_CAP_FPAGE_R is always on but might be specified for documentation purposes. Other rights cannot be used in this context, because they cannot be checked at the server side.

Examples:
```
L4_RPC(long, func, (unsigned value), L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
```

Definition at line 257 of file ipc_iface.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_iface

15.101 L4::Ipc::Call_zero_send_timeout Struct Reference

RPC attribute for an RPC call, with zero send timeout.

Inheritance diagram for L4::Ipc::Call_zero_send_timeout:

Collaboration diagram for L4::Ipc::Call_zero_send_timeout:

15.101.1 Detailed Description

RPC attribute for an RPC call, with zero send timeout.

Definition at line 236 of file ipc_iface.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_iface

15.102 L4::Ipc::Cap< T > Class Template Reference

Capability type for RPC interfaces (see `L4::Cap<T>`).

Collaboration diagram for L4::Ipc::Cap< T >:

Public Types

- enum { Rights_mask = 0xff , Cap_mask = L4_CAP_MASK }

Public Member Functions

- template<typename O >
  Cap (Cap< O > const &o) noexcept

  Make copy with conversion.
- Cap (L4::Cap< T > cap) noexcept

  Make a Cap from L4::Cap<T>, with minimal rights.
- template<typename O >
  Cap (L4::Cap< O > cap) noexcept

  Make IPC Cap from L4::Cap with conversion (and minimal rights).
- Cap () noexcept

  Make an invalid cap.
- Cap (L4::Cap< T > cap, unsigned char rights) noexcept

  Make a Cap from L4::Cap<T> with the given rights.
- L4::Cap< T > cap () const noexcept

  Return the L4::Cap<T> of this Cap.
- unsigned rights () const noexcept

  Return the rights bits stored in this IPC cap.
- L4::Ipc::Snd_fpage fpage () const noexcept

  Return the send flexpage for this Cap (see l4_fpage_t)
- bool is_valid () const noexcept

  Return true if this Cap is valid.

Static Public Member Functions

- static Cap from_ci (l4_cap_idx_t c) noexcept

  Create an IPC capability from a C capability index plus rights.

15.102.1 Detailed Description

template<typename T>
class L4::Ipc::Cap< T >

Capability type for RPC interfaces (see `L4::Cap<T>`).

Template Parameters

T	type of the interface referenced by the capability.

In contrast to `L4::Cap<T>` this type additionally stores a rights mask that shall be used when the capability is transferred to the receiver. This allows to apply restrictions to the transferred capability in the form of a subset of the rights possessed by the sender.

See also

L4::Ipc::make_cap()

Definition at line 541 of file ipc_types.

15.102.2 Member Enumeration Documentation

15.102.2.1 anonymous enum

```
template<typename T >
anonymous enum
```

Enumerator

Rights_mask	Mask for rights bits stored internally. L4_FPAGE_RIGHTS_MASK | L4_FPAGE_C_NO_REF_CNT | L4_FPAGE_C_OBJ_RIGHTS).
Cap_mask	Mask for significant capability bits. (incl. the invalid bit to support invalid caps)

Definition at line 547 of file ipc_types.

15.102.3 Constructor & Destructor Documentation

15.102.3.1 Cap()

```
template<typename T >
L4::Ipc::Cap< T >::Cap (
            L4::Cap< T > cap,
            unsigned char rights )  [inline], [noexcept]
```

Make a Cap from L4::Cap<T> with the given rights.

Parameters

cap	Capability to be sent.
rights	Rights to be sent. Consists of L4_fpage_rights and L4_obj_fpage_ctl.

Definition at line 589 of file ipc_types.

15.102.4 Member Function Documentation

15.102.4.1 from_ci()

```
template<typename T >
static Cap L4::Ipc::Cap< T >::from_ci (
            l4_cap_idx_t c )  [inline], [static], [noexcept]
```

Create an IPC capability from a C capability index plus rights.

Parameters

c	C capability index with the lowest 8 bits used as rights for the map operation (see L4_fpage_rights).

Definition at line 597 of file ipc_types.

References L4::Ipc::Cap< T >::Cap_mask, and L4::Ipc::Cap< T >::Rights_mask.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_types

15.103 L4::Ipc::Gen_fpage< T > Class Template Reference

Generic RPC wrapper for L4 flex-pages.

Inherits T.

Collaboration diagram for L4::Ipc::Gen_fpage< T >:

Public Types

- enum Type

  Type of mapping object, see L4_fpage_type.
- enum Map_type

  Kind of mapping.
- enum Cacheopt

  Caching options, see l4_fpage_cacheability_opt_t.

Public Member Functions

- bool is_valid () const noexcept

  Check if the capability is valid.
- bool cap_received () const noexcept

  Check if the capability has been mapped.
- bool id_received () const noexcept

  Check if a label was received instead of a mapping.
- bool local_id_received () const noexcept

  Check if a local capability id has been received.
- bool is_compound () const noexcept

  Check if the received item has the compound bit set.
- l4_umword_t data () const noexcept

  Return the raw flex page descriptor.
- l4_umword_t base_x () const noexcept

  Return the raw base descriptor.

15.103.1 Detailed Description

template<typename T>
class L4::Ipc::Gen_fpage< T >

Generic RPC wrapper for L4 flex-pages.

Template Parameters

T	Underlying specific flexpage type.

Definition at line 321 of file ipc_types.

15.103.2 Member Function Documentation

15.103.2.1 cap_received()

```
template<typename T >
bool L4::Ipc::Gen_fpage< T >::cap_received ( ) const  [inline], [noexcept]
```

Check if the capability has been mapped.

The capability itself can then be retrieved from the cap slot that has been provided in the receive operation.

Definition at line 438 of file ipc_types.

15.103.2.2 id_received()

```
template<typename T >
bool L4::Ipc::Gen_fpage< T >::id_received ( ) const  [inline], [noexcept]
```

Check if a label was received instead of a mapping.

For IPC gates, if the L4_RCV_ITEM_LOCAL_ID has been set, then only the label of the IPC gate will be provided if the gate is local to the receiver, i.e. the target thread of the IPC gate is in the same task as the receiving thread.

The label can be retrieved with Gen_fpage::data().

Definition at line 450 of file ipc_types.

15.103.2.3 is_compound()

```
template<typename T >
bool L4::Ipc::Gen_fpage< T >::is_compound ( ) const  [inline], [noexcept]
```

Check if the received item has the compound bit set.

A set compound bit means the next message item of the same type will be mapped to the same receive buffer as this message item.

Definition at line 468 of file ipc_types.

15.103.2.4 local_id_received()

```
template<typename T >
bool L4::Ipc::Gen_fpage< T >::local_id_received ( ) const  [inline], [noexcept]
```

Check if a local capability id has been received.

If the L4_RCV_ITEM_LOCAL_ID flag has been set by the receiver, and sender and receiver are in the same task, then only the capability index is transferred.

The capability can be retrieved with Gen_fpage::data().

Definition at line 460 of file ipc_types.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_types

15.104 L4::Ipc::In_out< T > Struct Template Reference

Mark an argument as in-out argument.

Collaboration diagram for L4::Ipc::In_out< T >:

15.104.1 Detailed Description

template<typename T>
struct L4::Ipc::In_out< T >

Mark an argument as in-out argument.

Template Parameters

T	The original argument type, usually a pointer or a reference.

In_out<> is used when an otherwise output-only value shall also be used as input value.

Definition at line 52 of file ipc_types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_types

15.105 L4::Ipc::Iostream Class Reference

Input/Output stream for IPC [un]marshalling.

Inheritance diagram for L4::Ipc::Iostream:

Collaboration diagram for L4::Ipc::Iostream:

Public Member Functions

- lostream (l4_utcb_t *utcb)

  Create an IPC IO stream with a single message buffer.
- void reset ()

  Reset the stream to its initial state.

IPC operations.

- l4_msgtag_t call (l4_cap_idx_t dst, l4_timeout_t timeout, long proto=0)
  Do an IPC call using the message in the output stream and receive the reply in the input stream.
- l4_msgtag_t **call** (l4_cap_idx_t dst, long proto=0)
- l4_msgtag_t reply_and_wait (l4_umword_t *src_dst, long proto=0)
  Do an IPC reply and wait.
- l4_msgtag_t **send_and_wait** (l4_cap_idx_t dest, l4_umword_t *src, long proto=0)
- l4_msgtag_t reply_and_wait (l4_umword_t *src_dst, l4_timeout_t timeout, long proto=0)
  Do an IPC reply and wait.
- l4_msgtag_t **send_and_wait** (l4_cap_idx_t dest, l4_umword_t *src, l4_timeout_t timeout, long proto=0)
- l4_msgtag_t **reply** (l4_timeout_t timeout, long proto=0)
- l4_msgtag_t **reply** (long proto=0)

Get/Put functions.

These functions are basically used to implement the insertion operators (<<) and should not be called directly.

- template<typename T >
  unsigned long get (T *buf, unsigned long elems)
  Copy out an array of type `T` with `size` elements.
- template<typename T >
  unsigned long get (Msg_ptr< T > const &buf, unsigned long elems=1)
  Read one size elements of type T from the stream and return a pointer.
- template<typename T >
  bool get (T &v)
  Extract a single element of type T from the stream.
- bool **get** (Ipc::Varg *va)
- bool **put** (Buf_item const &)
- bool **put** (Small_buf const &)
- template<typename T >
  bool put (T *buf, unsigned long size)
  Put an array with `size` elements of type `T` into the stream.
- template<typename T >
  bool put (T const &v)
  Insert an element of type `T` into the stream.
- int **put** (Varg const &va)
- template<typename T >
  int **put** (Varg_t< T > const &va)

15.105.1 Detailed Description

Input/Output stream for IPC [un]marshalling.

The Ipc::Iostream is part of the AW Env IPC framework as well as Ipc::Istream and Ipc::Ostream. In particular an Ipc::Iostream is a combination of an Ipc::Istream and an Ipc::Ostream. It can use either a single message buffer for receiving and sending messages or a pair of a receive and a send buffer. The stream also supports combined IPC operations such as call() and reply_and_wait(), which can be used to implement RPC functionality.

Examples

examples/libs/l4re/c++/shared_ds/ds_srv.cc, examples/libs/l4re/streammap/client.cc, and examples/libs/l4re/streammap/server.c

Definition at line 800 of file ipc_stream.

15.105.2 Constructor & Destructor Documentation

15.105.2.1 Iostream()

```
L4::Ipc::Iostream::Iostream (
            l4_utcb_t * utcb )  [inline], [explicit]
```

Create an IPC IO stream with a single message buffer.

Parameters

utcb	The message buffer used as backing store.

The created IO stream uses the same message buffer for sending and receiving IPC messages.

Definition at line 812 of file ipc_stream.

15.105.3 Member Function Documentation

15.105.3.1 call()

```
l4_msgtag_t L4::Ipc::Iostream::call (
            l4_cap_idx_t dst,
            l4_timeout_t timeout,
            long proto = 0 )  [inline]
```

Do an IPC call using the message in the output stream and receive the reply in the input stream.

Parameters

dst	The destination to call.
timeout	The IPC timeout for the call.
proto	The protocol value to use in the message tag.

Returns

The result tag of the IPC operation.

This is a combined IPC operation consisting of a send and a receive to/from the given destination `dst`.

A call is usually used by clients for RPCs to a server.

Examples

examples/libs/l4re/streammap/client.cc.

Definition at line 971 of file ipc_stream.

References l4_ipc_call(), and L4::Ipc::Istream::tag().

Here is the call graph for this function:

15.105.3.2 get() [1/3]

```
template<typename T >
unsigned long L4::Ipc::Istream::get (
            typename T  )  [inline]
```

Read one size elements of type T from the stream and return a pointer.

Parameters

buf	A Msg_ptr that is actually set to point to the element in the stream.
elems	Number of elements to extract (default is 1).

Returns

The number of elements extracted.

In contrast to a normal get, this version does actually not copy the data but returns a pointer to the data.

See Istream::operator>>()

Definition at line 450 of file ipc_stream.

15.105.3.3 get() [2/3]

```
template<typename T >
bool L4::Ipc::Istream::get (
            typename T  )  [inline]
```

Extract a single element of type T from the stream.

Parameters

out	*v*	The element.

Return values

true	An element was successfully extracted.
false	An element could not be extracted.

See Istream::operator>>()

Definition at line 475 of file ipc_stream.

15.105.3.4 get() [3/3]

```
template<typename T >
unsigned long L4::Ipc::Istream::get (
            typename T  )  [inline]
```

Copy out an array of type `T` with `size` elements.

Parameters

buf	Pointer to a buffer for size elements of type T.
elems	Number of elements of type T to copy out.

Returns

The number of elements copied out.

See Istream::operator>>()

Definition at line 405 of file ipc_stream.

15.105.3.5 put() [1/2]

```
template<typename T >
bool L4::Ipc::Ostream::put (
            typename T  )  [inline]
```

Put an array with `size` elements of type `T` into the stream.

Parameters

buf	A pointer to the array to insert into the buffer.
size	The number of elements in the array.

Definition at line 669 of file ipc_stream.

15.105.3.6 put() [2/2]

```
template<typename T >
bool L4::Ipc::Ostream::put (
            typename T  )  [inline]
```

Insert an element of type T into the stream.

Parameters

v	The element to insert.

Definition at line 687 of file ipc_stream.

15.105.3.7 reply_and_wait() [1/2]

```
l4_msgtag_t L4::Ipc::Iostream::reply_and_wait (
            l4_umword_t * src_dst,
            l4_timeout_t timeout,
            long proto = 0 )  [inline]
```

Do an IPC reply and wait.

Parameters

in,out	*src_dst*	Input: the destination for the send operation. Output: the source of the received message.
	timeout	Timeout used for IPC.
	proto	Protocol to use.

Returns

The result tag of the IPC operation.

This is a combined IPC operation consisting of a send operation and an open wait for any message.

A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 986 of file ipc_stream.

References l4_ipc_reply_and_wait(), and L4::Ipc::Istream::tag().

Here is the call graph for this function:

15.105.3.8 reply_and_wait() [2/2]

```
l4_msgtag_t L4::Ipc::Iostream::reply_and_wait (
            l4_umword_t * src_dst,
            long proto = 0 )  [inline]
```

Do an IPC reply and wait.

Parameters

in,out	*src_dst*	Input: the destination for the send operation. Output: the source of the received message.
	proto	Protocol to use.

Returns

The result tag of the IPC operation.

This is a combined IPC operation consisting of a send operation and an open wait for any message.

A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 885 of file ipc_stream.

References L4_IPC_SEND_TIMEOUT_0.

15.105.3.9 reset()

```
void L4::Ipc::Iostream::reset ( )  [inline]
```

Reset the stream to its initial state.

Input as well as the output stream are reset.

Definition at line 826 of file ipc_stream.

References L4::Ipc::Istream::reset(), and L4::Ipc::Ostream::reset().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/ipc_stream

15.106 L4::Ipc::Istream Class Reference

Input stream for IPC unmarshalling.

Inheritance diagram for L4::Ipc::Istream:

Collaboration diagram for L4::Ipc::Istream:

Public Member Functions

- Istream (l4_utcb_t *utcb)

  Create an input stream for the given message buffer.
- void reset ()

  Reset the stream to empty, and ready for receive()/wait().

- template<typename T >
  bool has_more (unsigned long count=1)
  Check whether a value of type T can be obtained from the stream.
- l4_utcb_t * utcb () const
  Return utcb pointer.

Get/Put Functions.

- template<typename T >
  unsigned long get (T *buf, unsigned long elems)
  Copy out an array of type `T` with `size` elements.
- template<typename T >
  void skip (unsigned long elems)
  Skip size elements of type T in the stream.
- template<typename T >
  unsigned long get (Msg_ptr< T > const &buf, unsigned long elems=1)
  Read one size elements of type T from the stream and return a pointer.
- template<typename T >
  bool get (T &v)
  Extract a single element of type T from the stream.
- bool **get** (Ipc::Varg *va)
- l4_msgtag_t tag () const
  Get the message tag of a received IPC.
- l4_msgtag_t & tag ()
  Get the message tag of a received IPC.

IPC operations.

- l4_msgtag_t wait (l4_umword_t *src)
  Wait for an incoming message from any sender.
- l4_msgtag_t wait (l4_umword_t *src, l4_timeout_t timeout)
  Wait for an incoming message from any sender.
- l4_msgtag_t receive (l4_cap_idx_t src)
  Wait for a message from the specified sender.
- l4_msgtag_t **receive** (l4_cap_idx_t src, l4_timeout_t timeout)

15.106.1 Detailed Description

Input stream for IPC unmarshalling.

Ipc::Istream is part of the dynamic IPC marshalling infrastructure, as well as Ipc::Ostream and Ipc::Iostream.

Ipc::Istream is an input stream supporting extraction of values from an IPC message buffer. A received IPC message can be unmarshalled using the usual extraction operator (>>).

There exist some special wrapper classes to extract arrays (see Ipc_buf_cp_in and Ipc_buf_in) and indirect strings (see Msg_in_buffer and Msg_io_buffer).

Definition at line 345 of file ipc_stream.

15.106.2 Constructor & Destructor Documentation

15.106.2.1 Istream()

```
L4::Ipc::Istream::Istream (
            l4_utcb_t * utcb )  [inline]
```

Create an input stream for the given message buffer.

The given message buffer is used for IPC operations wait()/receive() and received data can be extracted using the >> operator afterwards. In the case of indirect message parts a buffer of type Msg_in_buffer must be inserted into the stream before the IPC operation and contains received data afterwards.

Parameters

utcb	The message buffer to receive IPC messages.

Definition at line 359 of file ipc_stream.

15.106.3 Member Function Documentation

15.106.3.1 get() [1/3]

```
template<typename T >
unsigned long L4::Ipc::Istream::get (
            Msg_ptr< T > const & buf,
            unsigned long elems = 1 )  [inline]
```

Read one size elements of type T from the stream and return a pointer.

Parameters

buf	A Msg_ptr that is actually set to point to the element in the stream.
elems	Number of elements to extract (default is 1).

Returns

The number of elements extracted.

In contrast to a normal get, this version does actually not copy the data but returns a pointer to the data.

See Istream::operator>>()

Definition at line 450 of file ipc_stream.

References L4_UNLIKELY.

15.106.3.2 get() [2/3]

```
template<typename T >
bool L4::Ipc::Istream::get (
            T & v )  [inline]
```

Extract a single element of type T from the stream.

Parameters

out	*v*	The element.

Return values

true	An element was successfully extracted.
false	An element could not be extracted.

See Istream::operator>>()

Definition at line 475 of file ipc_stream.

References L4_UNLIKELY.

15.106.3.3 get() [3/3]

```
template<typename T >
unsigned long L4::Ipc::Istream::get (
            T * buf,
            unsigned long elems )  [inline]
```

Copy out an array of type `T` with `size` elements.

Parameters

buf	Pointer to a buffer for size elements of type T.
elems	Number of elements of type T to copy out.

Returns

The number of elements copied out.

See Istream::operator>>()

Definition at line 405 of file ipc_stream.

References L4_UNLIKELY.

Referenced by operator>>().

Here is the caller graph for this function:

15.106.3.4 receive()

```
l4_msgtag_t L4::Ipc::Istream::receive (
            l4_cap_idx_t src ) [inline]
```

Wait for a message from the specified sender.

Parameters

src	The sender id to receive from.

Returns

The IPC result tag (l4_msgtag_t).

This is commonly known as 'closed wait'.

Definition at line 583 of file ipc_stream.

References L4_IPC_NEVER.

15.106.3.5 reset()

```
void L4::Ipc::Istream::reset ( ) [inline]
```

Reset the stream to empty, and ready for receive()/wait().

The stream is reset to the same state as on its creation.

Definition at line 369 of file ipc_stream.

Referenced by L4::Ipc::Iostream::reset().

Here is the caller graph for this function:

15.106.3.6 skip()

```
template<typename T >
void L4::Ipc::Istream::skip (
            unsigned long elems )  [inline]
```

Skip size elements of type T in the stream.

Parameters

elems	Number of elements to skip.

Definition at line 425 of file ipc_stream.

References L4_UNLIKELY.

15.106.3.7 tag() [1/2]

```
l4_msgtag_t& L4::Ipc::Istream::tag ( )  [inline]
```

Get the message tag of a received IPC.

Returns

A reference to the L4 message tag for the received IPC.

This is in particular useful for handling page faults or exceptions.

See Istream::operator>>()

Definition at line 528 of file ipc_stream.

15.106.3.8 tag() [2/2]

```
l4_msgtag_t L4::Ipc::Istream::tag ( ) const  [inline]
```

Get the message tag of a received IPC.

Returns

The L4 message tag for the received IPC.

This is in particular useful for handling page faults or exceptions.

See Istream::operator>>()

Definition at line 516 of file ipc_stream.

Referenced by L4::Ipc::Iostream::call(), operator>>(), L4::Ipc::Iostream::reply_and_wait(), and wait().

Here is the caller graph for this function:

15.106.3.9 wait() [1/2]

```
l4_msgtag_t L4::Ipc::Istream::wait (
            l4_umword_t * src )  [inline]
```

Wait for an incoming message from any sender.

Parameters

out	*src*	Contains the sender after a successful IPC operation.

Returns

Syscall return tag.

This wait is actually known as 'open wait'.

Definition at line 559 of file ipc_stream.

References L4_IPC_NEVER.

15.106.3.10 wait() [2/2]

```
l4_msgtag_t L4::Ipc::Istream::wait (
            l4_umword_t * src,
            l4_timeout_t timeout )  [inline]
```

Wait for an incoming message from any sender.

Parameters

out	*src*	Contains the sender after a successful IPC operation.
	timeout	Timeout used for IPC.

Returns

The IPC result tag (l4_msgtag_t).

This wait is actually known as 'open wait'.

Definition at line 1018 of file ipc_stream.

References l4_ipc_wait(), and tag().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/ipc_stream

15.107 L4::Ipc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference

Defines client-side handling of 'MTYPE' as RPC argument.

Inheritance diagram for L4::Ipc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >:

Collaboration diagram for L4::Ipc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >:

15.107.1 Detailed Description

template<typename MTYPE, typename DIR, typename CLASS>
struct L4::Ipc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >

Defines client-side handling of 'MTYPE' as RPC argument.

Template Parameters

MTYPE	Elem<T>::arg_type (where T is the type used in the RPC definition)
DIR	Dir_in (client -> server), or Dir_out (server -> client)
CLASS	Cls_data, Cls_item, or Cls_buffer

Definition at line 221 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.108 L4::Ipc::Msg::Cls_buffer Struct Reference

Marker type for receive buffer values.

Inheritance diagram for L4::Ipc::Msg::Cls_buffer:

Collaboration diagram for L4::Ipc::Msg::Cls_buffer:

15.108.1 Detailed Description

Marker type for receive buffer values.

Definition at line 165 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.109 L4::Ipc::Msg::Cls_data Struct Reference

Marker type for data values.

Inheritance diagram for L4::Ipc::Msg::Cls_data:

Collaboration diagram for L4::Ipc::Msg::Cls_data:

15.109.1 Detailed Description

Marker type for data values.

Definition at line 161 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.110 L4::Ipc::Msg::Cls_item Struct Reference

Marker type for item values.

Inheritance diagram for L4::Ipc::Msg::Cls_item:

Collaboration diagram for L4::Ipc::Msg::Cls_item:

15.110.1 Detailed Description

Marker type for item values.

Definition at line 163 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.111 L4::Ipc::Msg::Dir_in Struct Reference

Marker type for input values.

Inheritance diagram for L4::Ipc::Msg::Dir_in:

Collaboration diagram for L4::Ipc::Msg::Dir_in:

15.111.1 Detailed Description

Marker type for input values.

Definition at line 156 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.112 L4::Ipc::Msg::Dir_out Struct Reference

Marker type for output values.

Inheritance diagram for L4::Ipc::Msg::Dir_out:

Collaboration diagram for L4::Ipc::Msg::Dir_out:

15.112.1 Detailed Description

Marker type for output values.

Definition at line 158 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.113 L4::Ipc::Msg::Do_in_data Struct Reference

Marker for Input data.

Inheritance diagram for L4::Ipc::Msg::Do_in_data:

Collaboration diagram for L4::Ipc::Msg::Do_in_data:

15.113.1 Detailed Description

Marker for Input data.

Definition at line 169 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.114 L4::Ipc::Msg::Do_in_items Struct Reference

Marker for Input items.

Inheritance diagram for L4::Ipc::Msg::Do_in_items:

Collaboration diagram for L4::Ipc::Msg::Do_in_items:

15.114.1 Detailed Description

Marker for Input items.

Definition at line 173 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.115 L4::Ipc::Msg::Do_out_data Struct Reference

Marker for Output data.

Inheritance diagram for L4::Ipc::Msg::Do_out_data:

Collaboration diagram for L4::Ipc::Msg::Do_out_data:

15.115.1 Detailed Description

Marker for Output data.

Definition at line 171 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.116 L4::Ipc::Msg::Do_out_items Struct Reference

Marker for Output items.

Inheritance diagram for L4::Ipc::Msg::Do_out_items:

Collaboration diagram for L4::Ipc::Msg::Do_out_items:

15.116.1 Detailed Description

Marker for Output items.

Definition at line 175 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.117 L4::Ipc::Msg::Do_rcv_buffers Struct Reference

Marker for receive buffers.

Inheritance diagram for L4::Ipc::Msg::Do_rcv_buffers:

Collaboration diagram for L4::Ipc::Msg::Do_rcv_buffers:

15.117.1 Detailed Description

Marker for receive buffers.

Definition at line 177 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.118 L4::Ipc::Msg::Elem< Array< A, LEN > & > Struct Template Reference

Array as output argument.

Collaboration diagram for L4::Ipc::Msg::Elem< Array< A, LEN > & >:

Public Types

- typedef Array< A, LEN > & arg_type

  Array<> & at the interface.
- typedef Array_ref< A, LEN > svr_type

  Array_ref<> as server storage type.
- typedef svr_type & svr_arg_type

  Array_ref<> & at the server side.

15.118.1 Detailed Description

template<typename A, typename LEN>
struct L4::Ipc::Msg::Elem< Array< A, LEN > & >

Array as output argument.

Definition at line 167 of file ipc_array.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

15.119 L4::Ipc::Msg::Elem< Array< A, LEN > > Struct Template Reference

Array as input arguments.

Collaboration diagram for L4::Ipc::Msg::Elem< Array< A, LEN > >:

Public Types

- typedef Array< A, LEN > arg_type

  Array<> as argument at the interface.
- typedef Array_ref< A, LEN > svr_type

  Array_ref<> at the server side.

15.119.1 Detailed Description

template<typename A, typename LEN>
struct L4::Ipc::Msg::Elem< Array< A, LEN > >

Array as input arguments.

Definition at line 155 of file ipc_array.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

15.120 L4::Ipc::Msg::Elem< Array_ref< A, LEN > & > Struct Template Reference

Array_ref as output argument.

Collaboration diagram for L4::Ipc::Msg::Elem< Array_ref< A, LEN > & >:

Public Types

- typedef Array_ref< A, LEN > & arg_type

  Array_ref<> at the interface.
- typedef Array_ref< typename L4::Types::Remove_const< A >::type, LEN > svr_type

  Array_ref<> as server storage.
- typedef svr_type & svr_arg_type

  Array_ref<> & as server argument.

15.120.1 Detailed Description

template<typename A, typename LEN>
struct L4::Ipc::Msg::Elem< Array_ref< A, LEN > & >

Array_ref as output argument.

Definition at line 180 of file ipc_array.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

15.121 L4::Ipc::Msg::Is_valid_rpc_type< T > Struct Template Reference

Type trait defining a valid RPC parameter type.

Inheritance diagram for L4::Ipc::Msg::Is_valid_rpc_type< T >:

Collaboration diagram for L4::Ipc::Msg::Is_valid_rpc_type< T >:

Additional Inherited Members

15.121.1 Detailed Description

template<typename T>
struct L4::Ipc::Msg::Is_valid_rpc_type< T >

Type trait defining a valid RPC parameter type.

Definition at line 350 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.122 L4::Ipc::Msg::Svr_arg_pack< IPC_TYPE > Struct Template Reference

Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function.

Collaboration diagram for L4::Ipc::Msg::Svr_arg_pack< IPC_TYPE >:

15.122.1 Detailed Description

template<typename IPC_TYPE>
struct L4::Ipc::Msg::Svr_arg_pack< IPC_TYPE >

Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function.

Definition at line 155 of file ipc_server.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server

15.123 L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference

Defines server-side handling for `MTYPE` server arguments.

Inheritance diagram for L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >:

Collaboration diagram for L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >:

15.123.1 Detailed Description

template<typename MTYPE, typename DIR, typename CLASS>
struct L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >

Defines server-side handling for `MTYPE` server arguments.

Template Parameters

MTYPE	Elem<T>::svr_type (where T is the type used in the RPC definition)
DIR	Dir_in (client -> server), or Dir_out (server -> client)
CLASS	Cls_data, Cls_item, or Cls_buffer

Definition at line 275 of file ipc_basics.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

15.124 L4::Ipc::Msg_ptr< T > Class Template Reference

Pointer to an element of type T in an Ipc::Istream.

Collaboration diagram for L4::Ipc::Msg_ptr< T >:

Public Member Functions

- Msg_ptr (T *&p)

  Create a Msg_ptr object that set pointer p to point into the message buffer.

15.124.1 Detailed Description

template<typename T>
class L4::Ipc::Msg_ptr< T >

Pointer to an element of type T in an Ipc::Istream.

This wrapper can be used to extract an element of type T from an Ipc::Istream, whereas the data is not copied out, but a pointer into the message buffer itself is returned. With is mechanism it is possible to avoid an extra copy of large data structures from a received IPC message, instead the returned pointer gives direct access to the data in the message.

See msg_ptr().

Definition at line 240 of file ipc_stream.

15.124.2 Constructor & Destructor Documentation

15.124.2.1 Msg_ptr()

```
template<typename T >
L4::Ipc::Msg_ptr< T >::Msg_ptr (
            T *& p )  [inline], [explicit]
```

Create a Msg_ptr object that set pointer p to point into the message buffer.

Parameters

p	The pointer that is adjusted to point into the message buffer.

Definition at line 251 of file ipc_stream.

The documentation for this class was generated from the following file:

- l4/cxx/ipc_stream

15.125 L4::Ipc::Opt< T > Struct Template Reference

Attribute for defining an optional RPC argument.

Collaboration diagram for L4::Ipc::Opt< T >:

Public Member Functions

- Opt () noexcept

  Make an absent optional argument.
- Opt (T value) noexcept

  Make a present optional argument with the given value.
- Opt & operator= (T value) noexcept

  Assign a value to the optional argument (makes the argument present)
- void set_valid (bool valid=true) noexcept

  Set the argument to present or absent.
- T ∗ operator-> () noexcept

  Get the pointer to the value.
- T const ∗ operator-> () const noexcept

Get the const pointer to the value.

- T value () const noexcept

  Get the value.

- T & value () noexcept

  Get the value.

- bool is_valid () const noexcept

  Get true if present, false if not.

Data Fields

- T _value

  The value.

- bool _valid

  True if the optional argument is present, false else.

15.125.1 Detailed Description

template<typename T>
struct L4::Ipc::Opt< T >

Attribute for defining an optional RPC argument.

Definition at line 147 of file ipc_types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_types

15.126 L4::Ipc::Ostream Class Reference

Output stream for IPC marshalling.

Inheritance diagram for L4::Ipc::Ostream:

Collaboration diagram for L4::Ipc::Ostream:

Public Member Functions

- Ostream (l4_utcb_t *utcb)

  Create an IPC output stream using the given message buffer `utcb`.
- void reset ()

  Reset the stream to empty, same state as a newly created stream.
- l4_utcb_t * utcb () const

Return utcb pointer.

Get/Put functions.

These functions are basically used to implement the insertion operators (<<) and should not be called directly.

- template<typename T >
  bool put (T *buf, unsigned long size)
  Put an array with `size` elements of type `T` into the stream.
- template<typename T >
  bool put (T const &v)
  Insert an element of type `T` into the stream.
- int **put** (Varg const &va)
- template<typename T >
  int **put** (Varg_t< T > const &va)
- l4_msgtag_t tag () const
  Extract the L4 message tag from the stream.
- l4_msgtag_t & tag ()
  Extract a reference to the L4 message tag from the stream.

IPC operations.

- l4_msgtag_t send (l4_cap_idx_t dst, long proto=0, unsigned flags=0)
  Send the message via IPC to the given receiver.

15.126.1 Detailed Description

Output stream for IPC marshalling.

Ipc::Ostream is part of the dynamic IPC marshalling infrastructure, as well as Ipc::Istream and Ipc::Iostream.

Ipc::Ostream is an output stream supporting insertion of values into an IPC message buffer. A IPC message can be marshalled using the usual insertion operator <<, see IPC stream operators .

There exist some special wrapper classes to insert arrays (see Ipc::Buf_cp_out) and indirect strings (see Msg_↩out_buffer and Msg_io_buffer).

Definition at line 632 of file ipc_stream.

15.126.2 Member Function Documentation

15.126.2.1 put() [1/2]

```
template<typename T >
bool L4::Ipc::Ostream::put (
            T * buf,
            unsigned long size )  [inline]
```

Put an array with `size` elements of type `T` into the stream.

Parameters

buf	A pointer to the array to insert into the buffer.
size	The number of elements in the array.

Definition at line 669 of file ipc_stream.

Referenced by operator<<().

Here is the caller graph for this function:

15.126.2.2 put() [2/2]

```
template<typename T >
bool L4::Ipc::Ostream::put (
            T const & v )  [inline]
```

Insert an element of type `T` into the stream.

Parameters

v	The element to insert.

Definition at line 687 of file ipc_stream.

15.126.2.3 send()

```
l4_msgtag_t L4::Ipc::Ostream::send (
            l4_cap_idx_t dst,
            long proto = 0,
            unsigned flags = 0 )  [inline]
```

Send the message via IPC to the given receiver.

Parameters

dst	The destination for the message.
proto	Protocol to use.
flags	Flags to use.

Returns

The syscall return tag.

Definition at line 964 of file ipc_stream.

References L4_IPC_NEVER, l4_ipc_send(), L4_MSGTAG_FLAGS, and tag().

Here is the call graph for this function:

15.126.2.4 tag() [1/2]

```
l4_msgtag_t& L4::Ipc::Ostream::tag ( ) [inline]
```

Extract a reference to the L4 message tag from the stream.

Returns

A reference to the L4 message tag.

Definition at line 722 of file ipc_stream.

15.126.2.5 tag() [2/2]

```
l4_msgtag_t L4::Ipc::Ostream::tag ( ) const [inline]
```

Extract the L4 message tag from the stream.

Returns

The extracted L4 message tag.

Definition at line 715 of file ipc_stream.

Referenced by operator<<(), and send().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/ipc_stream

15.127 L4::Ipc::Out< T > Struct Template Reference

Mark an argument as a output value in an RPC signature.

Collaboration diagram for L4::Ipc::Out< T >:

15.127.1 Detailed Description

template<typename T>
struct L4::Ipc::Out< T >

Mark an argument as a output value in an RPC signature.

Template Parameters

T	The original type of the argument.

Note

The use of Out<> is usually not needed, because typical out-put data types in C++ (pointers to non-const objects or non-const references are interpreted as output values anyway. However, there are some data types, such as returned capabilities that can be marked as such by using Out<>.

Definition at line 42 of file ipc_types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_types

15.128 L4::Ipc::Ret_array< T > Struct Template Reference

Dynamically sized output array of type T.

Collaboration diagram for L4::Ipc::Ret_array< T >:

15.128.1 Detailed Description

template<typename T>
struct L4::Ipc::Ret_array< T >

Dynamically sized output array of type T.

Template Parameters

T	The data-type of each array element.

Ret_array<> is a special dynamically sized output array where the number of transmitted elements is passed in the return value of the call (if positive)

Definition at line 34 of file ipc_ret_array.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_ret_array

15.129 L4::Ipc::Send_only Struct Reference

RPC attribute for a send-only RPC.

Collaboration diagram for L4::Ipc::Send_only:

15.129.1 Detailed Description

RPC attribute for a send-only RPC.

This class can be used as FLAGS parameter to L4::Ipc::Msg::Rpc_call and L4::Ipc::Msg::Rpc_inline_call templates and declares the RPC to use send-only semantics and timeouts.

Examples:

```
L4_RPC(long, send, (unsigned value), L4::Ipc::Send_only);
```

Definition at line 274 of file ipc_iface.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_iface

15.130 L4::Ipc::Small_buf Class Reference

A receive item for receiving a single capability.

Collaboration diagram for L4::Ipc::Small_buf:

Public Member Functions

- Small_buf (L4::Cap< void > cap, unsigned long flags=0) noexcept

  Create a receive item from a C++ cap.
- Small_buf (l4_cap_idx_t cap, unsigned long flags=0) noexcept

  Create a receive item from a C cap.

15.130.1 Detailed Description

A receive item for receiving a single capability.

This class is the main abstraction for receiving capabilities via Ipc::Istream. To receive a capability an instance of Small_buf that refers to an empty capability slot must be inserted into the Ipc::Istream before the receive operation.

Definition at line 268 of file ipc_types.

15.130.2 Constructor & Destructor Documentation

15.130.2.1 Small_buf() [1/2]

```
L4::Ipc::Small_buf::Small_buf (
            L4::Cap< void > cap,
            unsigned long flags = 0 )  [inline], [explicit], [noexcept]
```

Create a receive item from a C++ cap.

Parameters

cap	Capability slot where to save the capability.
flags	Receive buffer flags, see l4_msg_item_consts_t. L4_RCV_ITEM_SINGLE_CAP will always be set.

Definition at line 278 of file ipc_types.

15.130.2.2 Small_buf() [2/2]

```
L4::Ipc::Small_buf::Small_buf (
            l4_cap_idx_t cap,
            unsigned long flags = 0 )  [inline], [explicit], [noexcept]
```

Create a receive item from a C cap.

Parameters

cap	Capability slot where to save the capability.
flags	Receive buffer flags, see l4_msg_item_consts_t. L4_RCV_ITEM_SINGLE_CAP will always be set.

Definition at line 285 of file ipc_types.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_types

15.131 L4::Ipc::Snd_item Class Reference

RPC wrapper for a send item.

Collaboration diagram for L4::Ipc::Snd_item:

15.131.1 Detailed Description

RPC wrapper for a send item.

Definition at line 294 of file ipc_types.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_types

15.132 L4::Ipc::Str_cp_in< T > Class Template Reference

Abstraction for extracting a zero-terminated string from an Ipc::Istream.

Collaboration diagram for L4::Ipc::Str_cp_in< T >:

Public Member Functions

- Str_cp_in (T *v, unsigned long &size)

  Create a buffer for extracting an array from an Ipc::Istream.

15.132.1 Detailed Description

template<typename T>
class L4::Ipc::Str_cp_in< T >

Abstraction for extracting a zero-terminated string from an Ipc::Istream.

An instance of Str_cp_in can be used to extract a zero-terminated string an Ipc::Istream. The data from the received message is thereby copied to the given buffer and size is set to the number of characters found in the stream. The string is zero terminated in any circumstances. When the given buffer is smaller than the received string the last byte in the buffer will be the zero terminator. In the case the received string is shorter than the given buffer the zero termination will be placed behind the received data. This provides a zero-terminated result even in cases where the sender did not provide proper termination or in cases of too small receiver buffers.

See also

str_cp_in().

Definition at line 189 of file ipc_stream.

15.132.2 Constructor & Destructor Documentation

15.132.2.1 Str_cp_in()

```
template<typename T >
L4::Ipc::Str_cp_in< T >::Str_cp_in (
            T * v,
            unsigned long & size )  [inline]
```

Create a buffer for extracting an array from an Ipc::Istream.

Parameters

	v	The buffer for string.
in,out	*size*	Input: The number of bytes available in v Output: The number of bytes received (including the terminator).

Definition at line 200 of file ipc_stream.

The documentation for this class was generated from the following file:

- l4/cxx/ipc_stream

15.133 L4::Ipc::Varg Class Reference

Variably sized RPC argument.

Inherited by L4::Ipc::Varg_t< T >.

Collaboration diagram for L4::Ipc::Varg:

Public Types

- typedef l4_umword_t Tag

  The data type for the tag.

Public Member Functions

- L4_varg_type type () const
- unsigned length () const

  Get the size of the RPC argument.
- Tag tag () const
- void tag (Tag tag)

  Set Varg tag (usually from message)
- void data (char const *d)

  Set Varg to indirect data value (usually in UTCB)
- char const * data () const
- Varg ()=default

  Make uninitialized Varg.
- Varg (L4_varg_type t, void const *v, int len)

  Make an indirect varg.
- template<typename V >
  Va_type< V >::Ret_value value () const
- template<typename T >
  bool is_of () const
- bool is_nil () const
- bool is_of_int () const
- template<typename T >
  bool get_value (typename Va_type< T >::Value *v) const

  Get the value of the Varg as type T.
- template<typename T >
  void set_value (void const *d)

  Set to indirect value of type T.
- template<typename T >
  void set_direct_value (T val, typename L4::Types::Enable_if< sizeof(T)<=sizeof(char const *), bool >↩
  ::type=true)

  Set to directly stored value of type T.
- template<typename T >
  Varg (T const *data)

  Make Varg from indirect value (pointer)
- Varg (char const *data)

  Make Varg from null-terminated string.
- template<typename T >
  Varg (T data, typename L4::Types::Enable_if< sizeof(T)<=sizeof(char const *), bool >::type=true)

  Make Varg from direct value.

15.133.1 Detailed Description

Variably sized RPC argument.

Definition at line 96 of file ipc_varg.

15.133.2 Member Function Documentation

15.133.2.1 data()

```
char const* L4::Ipc::Varg::data ( ) const  [inline]
```

Returns

pointer to the data, also safe for direct data

Definition at line 123 of file ipc_varg.

Referenced by Varg().

Here is the caller graph for this function:

15.133.2.2 get_value()

```
template<typename T >
bool L4::Ipc::Varg::get_value (
            typename Va_type< T >::Value * v ) const  [inline]
```

Get the value of the Varg as type T.

Template Parameters

T	The expected type of the Varg.

Parameters

v	Pointer to store the value

Returns

true when the Varg is of type T, false if not

Definition at line 184 of file ipc_varg.

15.133.2.3 is_nil()

```
bool L4::Ipc::Varg::is_nil ( ) const  [inline]
```

Returns

true if the Varg is of nil type.

Definition at line 171 of file ipc_varg.

Referenced by L4::Ipc::Varg_list_ref::Iterator::equals(), L4::Ipc::Varg_list_ref::Iterator::operator bool(), and L4::Ipc::Varg_list_ref::Iterator::operator++().

Here is the caller graph for this function:

15.133.2.4 is_of()

```
template<typename T >
bool L4::Ipc::Varg::is_of ( ) const  [inline]
```

Returns

true if the Varg is of type T

Definition at line 168 of file ipc_varg.

References type().

Here is the call graph for this function:

15.133.2.5 is_of_int()

```
bool L4::Ipc::Varg::is_of_int ( ) const [inline]
```

Returns

true if the Varg is an integer type (signed or unsigned).

Definition at line 174 of file ipc_varg.

References type().

Here is the call graph for this function:

15.133.2.6 length()

```
unsigned L4::Ipc::Varg::length ( ) const  [inline]
```

Get the size of the RPC argument.

Returns

The size of the RPC argument

Definition at line 114 of file ipc_varg.

15.133.2.7 tag()

```
Tag L4::Ipc::Varg::tag ( ) const  [inline]
```

Returns

the tag value (the Direct_data bit masked)

Definition at line 116 of file ipc_varg.

15.133.2.8 type()

```
L4_varg_type L4::Ipc::Varg::type ( ) const  [inline]
```

Returns

the type field of the tag

Definition at line 109 of file ipc_varg.

Referenced by is_of(), and is_of_int().

Here is the caller graph for this function:

15.133.2.9 value()

```
template<typename V >
Va_type<V>::Ret_value L4::Ipc::Varg::value ( ) const  [inline]
```

Template Parameters

V	The data type of the value to retrieve.

Precondition

The Varg must be of type *V* (otherwise the result is unpredictable).

Returns

The value of the Varg as type V.

Definition at line 154 of file ipc_varg.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_varg

15.134 L4::Ipc::Varg_list< MAX > Class Template Reference

Self-contained list of variable-sized RPC parameters.

Inheritance diagram for L4::Ipc::Varg_list< MAX >:

Collaboration diagram for L4::Ipc::Varg_list< MAX >:

Public Member Functions

- Varg_list (Varg_list_ref const &r)

  Create a parameter list as a copy from a referencing list.

15.134.1 Detailed Description

template<unsigned MAX>
class L4::Ipc::Varg_list< MAX >

Self-contained list of variable-sized RPC parameters.

Works like Varg_list_ref but contains a full copy of the data. Use this as a parameter in server functions, if the handler function needs to use the UTCB (e.g. while sending further IPC).

Definition at line 410 of file ipc_varg.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_varg

15.135 L4::Ipc::Varg_list_ref Class Reference

List of variable-sized RPC parameters as received by the server.

Inheritance diagram for L4::Ipc::Varg_list_ref:

Collaboration diagram for L4::Ipc::Varg_list_ref:

Data Structures

- class Iterator

  Iterator for Valists.

Public Member Functions

- Varg_list_ref ()=default

  Create an empty parameter list.
- Varg_list_ref (void const *start, void const *end)

  Create a parameter list over a given memory region.
- Varg pop_front ()

  Get the next parameter in the list.
- Varg next ()

  Get the next parameter in the list.
- Iterator begin () const

  Returns an interator to the first Varg.
- Iterator end () const

  Returns the end of the list.

15.135.1 Detailed Description

List of variable-sized RPC parameters as received by the server.

The list can be traversed exactly once using *next()*.

This is a reference list, where the returned Varg point to data in the underlying storage, conventionally the UTCB. This type should only be used in server functions when the implementation can ensure that all content is read before the UTCB is reused (e.g. for IPC), otherwise use Varg_list.

Definition at line 252 of file ipc_varg.

15.135.2 Constructor & Destructor Documentation

15.135.2.1 Varg_list_ref()

```
L4::Ipc::Varg_list_ref::Varg_list_ref (
            void const * start,
            void const * end )  [inline]
```

Create a parameter list over a given memory region.

Parameters

start	Pointer to start of the parameter list.
end	Pointer to end of the list (inclusive).

Definition at line 331 of file ipc_varg.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_varg

15.136 L4::Ipc::Varg_list_ref::Iterator Class Reference

Iterator for Valists.

Collaboration diagram for L4::Ipc::Varg_list_ref::Iterator:

Public Member Functions

- Iterator (Iter_state const &s)

  Create a new iterator.
- operator bool () const

  validity check for the iterator
- Iterator & operator++ ()

  increment iterator to the next arg
- Varg operator∗ () const

  dereference the iterator, get Varg
- bool equals (Iterator const &o) const

  check for equality

15.136.1 Detailed Description

Iterator for Valists.

Definition at line 337 of file ipc_varg.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_varg

15.137 L4::Ipc_gate Class Reference

The C++ IPC gate interface, see IPC-Gate API for the C interface.

Inheritance diagram for L4::Ipc_gate:

Collaboration diagram for L4::Ipc_gate:

Public Member Functions

- l4_msgtag_t get_infos (l4_umword_t *label)

  Get information about the IPC-gate.

Additional Inherited Members

15.137.1 Detailed Description

The C++ IPC gate interface, see IPC-Gate API for the C interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the L4::Factory interface.

Depending on the permissions of the capability used, an IPC gate forwards IPC to the L4::Thread that is *bound* to the IPC gate (cf. bind_thread()). If the capability has the L4_FPAGE_C_IPCGATE_SVR permission, only IPC using a protocol different from the L4_PROTO_KOBJECT protocol is forwarded. Without the L4_FPAGE_C_IPCGATE_SVR permission, all IPC is forwarded. The latter is the usual case for a client in a client/server scenario. When no thread is bound yet, the forwarded IPC blocks until a thread is bound or the IPC times out.

Forwarded IPC is always forwarded to the userland of the bound thread. That means, the L4::Thread interface of the bound thread is not accessible via an IPC gate. The L4::Ipc_gate interface of an IPC gate is only accessible if the capability used has the L4_FPAGE_C_IPCGATE_SVR permission (cf. previous paragraph). Conversely that means, if the capability used lacks the L4_FPAGE_C_IPCGATE_SVR permission, L4::Ipc_gate interface calls are forwarded to the bound thread instead of being processed by the IPC gate itself. In a client/server scenario, a client should only get IPC gate capabilities without L4_FPAGE_C_IPCGATE_SVR permission so the client cannot tamper with the IPC gate.

When binding a thread to an IPC gate, a user-defined, kernel protected, machine-word sized payload called the IPC gate's *label* is assigned to the IPC gate (cf. bind_thread()). When a send-only IPC or call IPC is forwarded via an IPC gate, the label provided by the sender is ignored and replaced by the IPC gate's label where the two least significant bits are the result of bitwise disjunction of the corresponding label bits with the L4_CAP_FPAGE_S and L4_CAP_FPAGE_W permissions of the capability used. Hence, the label provided via bind_thread() should usually have its two least significant bits set to zero. The replaced label is only visible to the bound thread upon receive. However, the configured label of an IPC gate can also be queried via get_infos() if the capability used has the L4_FPAGE_C_IPCGATE_SVR permission.

When deleting an IPC gate or when unbinding it from a thread, the label of IPC already in flight won't be changed. To ensure that no IPC from this IPC gate is received by a thread with an unexpected label, L4::Thread::modify_senders() shall be used to change the labels of every pending IPC to that gate. This is also required if the label of an already bound IPC gate is changed. It is not necessary after binding the IPC gate to a thread for the first time.

Include File

```
#include <l4/sys/ipc_gate>
```

For the C interface refer to the C IPC-Gate API.

See also

Object Invocation

Definition at line 89 of file ipc_gate.

15.137.2 Member Function Documentation

15.137.2.1 get_infos()

```
l4_msgtag_t L4::Ipc_gate::get_infos (
            l4_umword_t * label )
```

Get information about the IPC-gate.

Parameters

out	*label*	The label of the IPC gate is returned here.

Returns

System call return tag.

The documentation for this class was generated from the following file:

- l4/sys/ipc_gate

15.138 L4::Ipc_svr::Br_manager_no_buffers Class Reference

Empty implementation of Server_iface.

Inheritance diagram for L4::Ipc_svr::Br_manager_no_buffers:

Collaboration diagram for L4::Ipc_svr::Br_manager_no_buffers:

Public Member Functions

- int alloc_buffer_demand (Demand const &demand)

  Tells the server to allocate buffers for the given demand.
- L4::Cap< void > get_rcv_cap (int) const

  Returns L4::Cap< void >::Invalid, we have no buffer management.
- int realloc_rcv_cap (int)

  Returns -L4_ENOMEM, we have no buffer management.
- int add_timeout (Timeout *, l4_kernel_clock_t)

  Returns -L4_ENOSYS, we have no timeout queue.
- int remove_timeout (Timeout *)

  Returns -L4_ENOSYS, we have no timeout queue.

Protected Member Functions

- unsigned first_free_br () const

  Returns 1 as first free buffer.
- void setup_wait (l4_utcb_t *utcb, L4::Ipc_svr::Reply_mode)

  Setup wait function for the server loop (Server<>).

Additional Inherited Members

15.138.1 Detailed Description

Empty implementation of Server_iface.

This implementation of Server_iface provides no buffer or timeout management at all it just returns errors for all calls that express other than empty demands. However, this may be useful for very simple servers that serve simple server objects only.

Definition at line 184 of file ipc_server_loop.

15.138.2 Member Function Documentation

15.138.2.1 alloc_buffer_demand()

```
int L4::Ipc_svr::Br_manager_no_buffers::alloc_buffer_demand (
            Demand const & demand )  [inline], [virtual]
```

Tells the server to allocate buffers for the given demand.

Parameters

demand	The total server-side demand of receive buffers needed for a given interface, see Demand.

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Returns

success (0) if demand is empty, -L4_ENOMEM else.

Implements L4::Ipc_svr::Server_iface.

Definition at line 191 of file ipc_server_loop.

References L4_ENOMEM, L4_EOK, and L4::Type_info::Demand::no_demand().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_server_loop

15.139 L4::Ipc_svr::Compound_reply Struct Reference

Mix in for LOOP_HOOKS to always use compound reply and wait.

Inheritance diagram for L4::Ipc_svr::Compound_reply:

Collaboration diagram for L4::Ipc_svr::Compound_reply:

15.139.1 Detailed Description

Mix in for LOOP_HOOKS to always use compound reply and wait.

Definition at line 77 of file ipc_server_loop.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

15.140 L4::Ipc_svr::Default_loop_hooks Struct Reference

Default LOOP_HOOKS.

Inheritance diagram for L4::Ipc_svr::Default_loop_hooks:

Collaboration diagram for L4::Ipc_svr::Default_loop_hooks:

Additional Inherited Members

15.140.1 Detailed Description

Default LOOP_HOOKS.

Combination of Ignore_errors, Default_timeout, Compound_reply, and Br_manager_no_buffers.

Definition at line 236 of file ipc_server_loop.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

15.141 L4::Ipc_svr::Default_setup_wait Struct Reference

Mix in for LOOP_HOOKS for setup_wait no op.

Collaboration diagram for L4::Ipc_svr::Default_setup_wait:

15.141.1 Detailed Description

Mix in for LOOP_HOOKS for setup_wait no op.

Definition at line 88 of file ipc_server_loop.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

15.142 L4::Ipc_svr::Default_timeout Struct Reference

Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.

Inheritance diagram for L4::Ipc_svr::Default_timeout:

Collaboration diagram for L4::Ipc_svr::Default_timeout:

15.142.1 Detailed Description

Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.

Definition at line 69 of file ipc_server_loop.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

15.143 L4::Ipc_svr::Direct_dispatch< R > Struct Template Reference

Direct dispatch helper, for forwarding dispatch calls to a registry *R*.

Inheritance diagram for L4::Ipc_svr::Direct_dispatch< R >:

Collaboration diagram for L4::Ipc_svr::Direct_dispatch< R >:

Public Member Functions

- Direct_dispatch (R &r)

  Make a direct dispatcher.
- l4_msgtag_t operator() (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)

  call operator forwarding to r.dispatch()

Data Fields

- R & r

  stores a reference to the registry object

15.143.1 Detailed Description

template<typename R>
struct L4::Ipc_svr::Direct_dispatch< R >

Direct dispatch helper, for forwarding dispatch calls to a registry *R*.

Template Parameters

R	Data type of the registry that is used for dispatching to different server objects, usually based on the protected IPC label.

Definition at line 99 of file ipc_server_loop.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

15.144 L4::Ipc_svr::Direct_dispatch< R * > Struct Template Reference

Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry *R*.

Collaboration diagram for L4::Ipc_svr::Direct_dispatch< R * >:

Public Member Functions

- Direct_dispatch (R *r)

  Make a direct dispatcher.
- l4_msgtag_t operator() (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)

  call operator forwarding to r->dispatch()

Data Fields

- R * r

  stores a pointer to the registry object

15.144.1 Detailed Description

template<typename R>
struct L4::Ipc_svr::Direct_dispatch< R * >

Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry *R*.

Template Parameters

R	Data type of the registry that is used for dispatching to different server objects, usually based on the protected IPC label.

Definition at line 120 of file ipc_server_loop.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

15.145 L4::Ipc_svr::Exc_dispatch< R, Exc > Struct Template Reference

Dispatch helper wrapping try {} catch {} around the dispatch call.

Inheritance diagram for L4::Ipc_svr::Exc_dispatch< R, Exc >:

Collaboration diagram for L4::Ipc_svr::Exc_dispatch< R, Exc >:

Public Member Functions

- Exc_dispatch (R r)

  Make an exception handling dispatcher.
- l4_msgtag_t operator() (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)

  Dispatch the call via Direct_dispatch<R>() and handle exceptions.

15.145.1 Detailed Description

template<typename R, typename Exc>
struct L4::Ipc_svr::Exc_dispatch< R, Exc >

Dispatch helper wrapping try {} catch {} around the dispatch call.

Template Parameters

R	Data type of the registry used for dispatching to objects.
Exc	Data type of the exceptions that shall be catched. This data type must provide a member err_no() that returns the negative integer (int) error code for the exception.

This dispatcher wraps Direct_dispatch<R> with a try-catch (Exc).

Definition at line 144 of file ipc_server_loop.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

15.146 L4::Ipc_svr::Ignore_errors Struct Reference

Mix in for LOOP_HOOKS to ignore IPC errors.

Inheritance diagram for L4::Ipc_svr::Ignore_errors:

Collaboration diagram for L4::Ipc_svr::Ignore_errors:

15.146.1 Detailed Description

Mix in for LOOP_HOOKS to ignore IPC errors.

Definition at line 61 of file ipc_server_loop.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

15.147 L4::Ipc_svr::Server_iface Class Reference

Interface for server-loop related functions.

Inheritance diagram for L4::Ipc_svr::Server_iface:

Collaboration diagram for L4::lpc_svr::Server_iface:

Public Types

- typedef L4::Type_info::Demand Demand

  Data type expressing server-side demand for receive buffers.

Public Member Functions

- Server_iface ()

  Make a server interface.
- virtual int alloc_buffer_demand (Demand const &demand)=0

  Tells the server to allocate buffers for the given demand.
- virtual L4::Cap< void > get_rcv_cap (int index) const =0

  Get capability slot allocated to the given receive buffer.
- virtual int realloc_rcv_cap (int index)=0

  Allocate a new capability for the given receive buffer.
- virtual int add_timeout (Timeout *timeout, l4_kernel_clock_t time)=0

  Add a timeout to the server internal timeout queue.
- virtual int remove_timeout (Timeout *timeout)=0

  Remove the given timeout from the timer queue.
- template<typename T >
  L4::Cap< T > rcv_cap (int index) const

  Get given receive buffer as typed capability.
- L4::Cap< void > rcv_cap (int index) const

  Get receive cap with the given index as generic (void) type.

15.147.1 Detailed Description

Interface for server-loop related functions.

This interface provides access to high-level server-loop related functions, such as management of receive buffers and timeouts.

Definition at line 47 of file ipc_epiface.

15.147.2 Member Function Documentation

15.147.2.1 add_timeout()

```
virtual int L4::Ipc_svr::Server_iface::add_timeout (
            Timeout * timeout,
            l4_kernel_clock_t time )  [pure virtual]
```

Add a timeout to the server internal timeout queue.

Parameters

timeout	The timeout object to register.
time	The time (absolute) at which the timeout shall expire.

Precondition

timeout must not be in any queue.

Returns

0 on success, 1 if timeout is already expired, $<$ 0 on error.

Implemented in L4::Ipc_svr::Timeout_queue_hooks$<$ HOOKS, BR_MAN $>$, L4::Ipc_svr::Timeout_queue_hooks$<$ Br_manager_time
L4::Ipc_svr::Br_manager_no_buffers, and L4Re::Util::Br_manager.

15.147.2.2 alloc_buffer_demand()

```
virtual int L4::Ipc_svr::Server_iface::alloc_buffer_demand (
            Demand const & demand )  [pure virtual]
```

Tells the server to allocate buffers for the given demand.

Parameters

demand	The total server-side demand of receive buffers needed for a given interface, see Demand.

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Implemented in L4::Ipc_svr::Br_manager_no_buffers, and L4Re::Util::Br_manager.

15.147.2.3 get_rcv_cap()

```
virtual L4::Cap<void> L4::Ipc_svr::Server_iface::get_rcv_cap (
            int index ) const  [pure virtual]
```

Get capability slot allocated to the given receive buffer.

Parameters

index	The receive buffer index of the expected capability argument (0 <= `index` < `caps` registered with alloc_buffer_demand()).

Precondition

0 <= `index` < `caps` registered with alloc_buffer_demand()

Returns

Capability slot currently allocated to the given receive buffer.

Implemented in L4::Ipc_svr::Br_manager_no_buffers, and L4Re::Util::Br_manager.

Referenced by rcv_cap().

Here is the caller graph for this function:

15.147.2.4 rcv_cap() [1/2]

```
template<typename T >
L4::Cap<T> L4::Ipc_svr::Server_iface::rcv_cap (
            int index ) const [inline]
```

Get given receive buffer as typed capability.

See also

get_rcv_cap()

Parameters

index	The receive buffer index of the expected capability argument. (0 <= `index` < `caps` registered with alloc_buffer_demand().)

Precondition

0 <= `index` < `caps` registered with alloc_buffer_demand()

Returns

Capability slot currently allocated to the given receive buffer.

Note

This is a convenience wrapper for get_rcv_cap() to avoid L4::cap_cast<>().

Definition at line 122 of file ipc_epiface.

References get_rcv_cap().

Here is the call graph for this function:

15.147.2.5 rcv_cap() [2/2]

```
L4::Cap<void> L4::Ipc_svr::Server_iface::rcv_cap (
            int index ) const [inline]
```

Get receive cap with the given index as generic (void) type.

Parameters

index	The index of the cap receive buffer of the expected capability. (0 <= `index` < `caps` registered with alloc_buffer_demand().)

Returns

Capability slot currently allocated to the given capability buffer.

Note

This is a convenience wrapper for get_rcv_cap().

Definition at line 134 of file ipc_epiface.

References get_rcv_cap().

Here is the call graph for this function:

15.147.2.6 realloc_rcv_cap()

```
virtual int L4::Ipc_svr::Server_iface::realloc_rcv_cap (
            int index )  [pure virtual]
```

Allocate a new capability for the given receive buffer.

Parameters

index	The receive buffer index of the expected capability argument (0 <= `index` < `caps` registered with alloc_buffer_demand()).

Precondition

0 <= `index` < `caps` registered with alloc_buffer_demand()

Returns

0 on success, < 0 on error.

Implemented in L4::Ipc_svr::Br_manager_no_buffers, and L4Re::Util::Br_manager.

15.147.2.7 remove_timeout()

```
virtual int L4::Ipc_svr::Server_iface::remove_timeout (
            Timeout * timeout )  [pure virtual]
```

Remove the given timeout from the timer queue.

Parameters

timeout	The timout object to remove.

Returns

0 on success, < 0 on error.

Implemented in L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >, L4::Ipc_svr::Timeout_queue_hooks< Br_manager_time
L4::Ipc_svr::Br_manager_no_buffers, and L4Re::Util::Br_manager.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_epiface

15.148 L4::Ipc_svr::Timeout Class Reference

Callback interface for Timeout_queue.

Inheritance diagram for L4::Ipc_svr::Timeout:

Collaboration diagram for L4::Ipc_svr::Timeout:

Public Member Functions

- Timeout ()

Make a timeout.

- virtual ~Timeout ()=0

  Destroy a timeout.

- virtual void expired ()=0

  callback function to be called when timeout happened

- l4_kernel_clock_t timeout () const

  return absolute timeout of this callback.

15.148.1 Detailed Description

Callback interface for Timeout_queue.

Definition at line 28 of file ipc_timeout_queue.

15.148.2 Member Function Documentation

15.148.2.1 expired()

```
virtual void L4::Ipc_svr::Timeout::expired ( )  [pure virtual]
```

callback function to be called when timeout happened

Note

The timeout object is already dequeued when this function is called, this means the timeout may be safely queued again within the expired() function.

Referenced by L4::Ipc_svr::Timeout_queue::handle_expired_timeouts().

Here is the caller graph for this function:

15.148.2.2 timeout()

```
l4_kernel_clock_t L4::Ipc_svr::Timeout::timeout ( ) const [inline]
```

return absolute timeout of this callback.

Returns

absolute timeout for this instance of the timeout.

Precondition

The timeout object must have been in a queue before, otherwise the timeout is not set.

Definition at line 52 of file ipc_timeout_queue.

The documentation for this class was generated from the following file:

- l4/cxx/ipc_timeout_queue

15.149 L4::Ipc_svr::Timeout_queue Class Reference

Timeout queue to be used in l4re server loop.

Collaboration diagram for L4::Ipc_svr::Timeout_queue:

Public Types

- typedef L4::Ipc_svr::Timeout Timeout

  Provide a local definition of Timeout for backward compat.

Public Member Functions

- l4_kernel_clock_t next_timeout () const

  Get the time for the next timeout.
- bool timeout_expired (l4_kernel_clock_t now) const

  Determine if a timeout has happened.
- void handle_expired_timeouts (l4_kernel_clock_t now)

  run the callbacks of expired timeouts
- void add (Timeout *timeout, l4_kernel_clock_t time)

  Add a timeout to the queue.
- void remove (Timeout *timeout)

  Remove timeout from the queue.

15.149.1 Detailed Description

Timeout queue to be used in l4re server loop.

Definition at line 65 of file ipc_timeout_queue.

15.149.2 Member Function Documentation

15.149.2.1 add()

```
void L4::Ipc_svr::Timeout_queue::add (
            Timeout * timeout,
            l4_kernel_clock_t time )  [inline]
```

Add a timeout to the queue.

Parameters

timeout	timeout object to add
time	the time when the timeout expires

Precondition

timeout must not be in any queue already

Definition at line 121 of file ipc_timeout_queue.

References cxx::Bits::Basic_list< POLICY >::begin(), cxx::Bits::Basic_list< POLICY >::end(), and cxx::H_list< T, POLICY >::insert_

Referenced by L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::add_timeout().

Here is the call graph for this function:

Here is the caller graph for this function:

15.149.2.2 handle_expired_timeouts()

```
void L4::Ipc_svr::Timeout_queue::handle_expired_timeouts (
            l4_kernel_clock_t now )  [inline]
```

run the callbacks of expired timeouts

Parameters

now	the current time.

Definition at line 101 of file ipc_timeout_queue.

References cxx::Bits::Basic_list< POLICY >::begin(), cxx::Bits::Basic_list< POLICY >::empty(), cxx::H_list< T, POLICY >::erase(), and L4::Ipc_svr::Timeout::expired().

Referenced by L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::setup_wait().

Here is the call graph for this function:

Here is the caller graph for this function:

15.149.2.3 next_timeout()

```
l4_kernel_clock_t L4::Ipc_svr::Timeout_queue::next_timeout ( ) const [inline]
```

Get the time for the next timeout.

Returns

the time for the next timeout or 0 if there is none

Definition at line 75 of file ipc_timeout_queue.

References cxx::Bits::Basic_list< POLICY >::front().

Referenced by L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::timeout(), and timeout_expired().

Here is the call graph for this function:

Here is the caller graph for this function:

15.149.2.4 remove()

```
void L4::Ipc_svr::Timeout_queue::remove (
            Timeout * timeout )  [inline]
```

Remove *timeout* from the queue.

Parameters

timeout	timeout to remove from timeout queue

Precondition

timeout must be in this queue

Definition at line 136 of file ipc_timeout_queue.

References cxx::H_list< T, POLICY >::remove().

Referenced by L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::remove_timeout().

Here is the call graph for this function:

Here is the caller graph for this function:

15.149.2.5 timeout_expired()

```
bool L4::Ipc_svr::Timeout_queue::timeout_expired (
            l4_kernel_clock_t now ) const  [inline]
```

Determine if a timeout has happened.

Parameters

now	The current time.

Return values

true	There is at least one expired timeout in the queue. false No expired timeout in the queue.

Definition at line 91 of file ipc_timeout_queue.

References next_timeout().

Referenced by L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::before_reply(), and L4::Ipc_svr::Timeout_queue_hooks<

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/ipc_timeout_queue

15.150 L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference

Loop hooks mixin for integrating a timeout queue into the server loop.

Inheritance diagram for L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >:

Collaboration diagram for L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >:

Public Member Functions

- l4_timeout_t timeout ()

  get the time for the next timeout
- void setup_wait (l4_utcb_t *utcb, L4::lpc_svr::Reply_mode mode)

  setup_wait() for the server loop
- L4::lpc_svr::Reply_mode before_reply (l4_msgtag_t, l4_utcb_t *)

  server loop hook
- int add_timeout (Timeout *timeout, l4_kernel_clock_t time)

  Add a timout to the queue for time time.
- int remove_timeout (Timeout *timeout)

  Remove timeout from the queue.

Data Fields

- Timeout_queue queue

  Use this timeout queue.

Additional Inherited Members

15.150.1 Detailed Description

template<typename HOOKS, typename BR_MAN = Br_manager_no_buffers>
class L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >

Loop hooks mixin for integrating a timeout queue into the server loop.

Template Parameters

HOOKS	has to inherit from Timeout_queue_hooks<> and provide the functions now() that has to return the current time.
BR_MAN	This used as a base class for and provides the API for selecting the buffer register (BR) that is used to store the timeout value. This is usually L4Re::Util::Br_manager or L4::Ipc_svr::Br_manager_no_buffers.

Definition at line 160 of file ipc_timeout_queue.

15.150.2 Member Function Documentation

15.150.2.1 add_timeout()

```
template<typename HOOKS , typename BR_MAN = Br_manager_no_buffers>
int L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::add_timeout (
            Timeout * timeout,
            l4_kernel_clock_t time )  [inline], [virtual]
```

Add a timout to the queue for time *time*.

Parameters

timeout	The timeout object to add into the queue (must not be in any queue currently).
time	The time when the timeout shall expire.

Precondition

timeout must not be in any queue.

Note

The timeout is automatically dequeued before the Timeout::expired() function is called

Implements L4::Ipc_svr::Server_iface.

Definition at line 212 of file ipc_timeout_queue.

References L4::Ipc_svr::Timeout_queue::add(), L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::queue, and L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::timeout().

Here is the call graph for this function:

15.150.2.2 remove_timeout()

```
template<typename HOOKS , typename BR_MAN = Br_manager_no_buffers>
int L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::remove_timeout (
            Timeout * timeout )  [inline], [virtual]
```

Remove timeout from the queue.

Parameters

timeout	The timeout object to be removed from the queue.

Note

This function may be safely called even if the timeout is not currently enqueued.

in Timeout::expired() the timeout is already dequeued!

Implements L4::Ipc_svr::Server_iface.

Definition at line 225 of file ipc_timeout_queue.

References L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::queue, L4::Ipc_svr::Timeout_queue::remove(), and L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::timeout().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/ipc_timeout_queue

15.151 L4::Irq Class Reference

C++ Irq interface, see IRQs for the C interface.

Inheritance diagram for L4::Irq:

Collaboration diagram for L4::Irq:

Public Member Functions

- l4_msgtag_t detach (l4_utcb_t *utcb=l4_utcb()) noexcept

  Detach from this interrupt.
- l4_msgtag_t receive (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept

  Unmask and wait for this IRQ.
- l4_msgtag_t wait (l4_umword_t *label, l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept

Unmask IRQ and (open) wait for any message.

- l4_msgtag_t unmask (l4_utcb_t *utcb=l4_utcb()) noexcept

  Unmask IRQ.

Additional Inherited Members

15.151.1 Detailed Description

C++ Irq interface, see IRQs for the C interface.

Note

"IRQ" is short for "interrupt request". This is often used interchangeably for "interrupt"

The Irq class provides access to abstract interrupts provided by the microkernel. Interrupts may be

- hardware interrupts provided by the platform interrupt controller,
- virtual device interrupts provided by the microkernel's virtual devices (virtual serial or trace buffer) or
- virtual interrupts that can be triggered by user programs (IRQs) via the inherited method L4::Triggerable::trigger().

For hardware and virtual device interrupts the Irq object must be bound to an interrupt source, see L4::Icu. To receive interrupts, the Irq object must be bound to a thread, see L4::Rcv_endpoint.

Irq objects can be created using a factory, see the L4::Factory API (L4::Factory::create()).

Include File

```
#include <l4/sys/irq>
```

For the C interface refer to the IRQs API for an overview.

Examples

examples/libs/l4re/c++/shared_ds/ds_clnt.cc.

Definition at line 135 of file irq.

15.151.2 Member Function Documentation

15.151.2.1 detach()

```
l4_msgtag_t L4::Irq::detach (
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Detach from this interrupt.

Parameters

utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 147 of file irq.

References l4_irq_detach_u().

Here is the call graph for this function:

15.151.2.2 receive()

```
l4_msgtag_t L4::Irq::receive (
            l4_timeout_t timeout = L4_IPC_NEVER,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Unmask and wait for this IRQ.

Parameters

timeout	Timeout.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Note

If this is the function normally used for your IRQs consider using L4::Semaphore instead of L4::Irq.

Definition at line 162 of file irq.

References l4_irq_receive_u().

Here is the call graph for this function:

15.151.2.3 unmask()

```
l4_msgtag_t L4::Irq::unmask (
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Unmask IRQ.

Parameters

utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag for a send-only operation, use l4_ipc_error() to check for errors (**do not** use l4_error()).

Note

This function is a send-only operation, this means there is no return value except for a failed send operation. Use l4_ipc_error() to check for errors, **do not** use l4_error(), because l4_error() will always return an error.

Irq::wait() and Irq::receive() operations already include an unmask(), do not use an extra unmask() in these cases.

Deprecated Use L4::Irq_eoi::unmask()

Definition at line 198 of file irq.

References L4_IPC_NEVER.

Referenced by wait().

Here is the caller graph for this function:

15.151.2.4 wait()

```
l4_msgtag_t L4::Irq::wait (
            l4_umword_t * label,
            l4_timeout_t timeout = L4_IPC_NEVER,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Unmask IRQ and (open) wait for any message.

Parameters

label	The *protected label* shall be received here.
timeout	Timeout.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 175 of file irq.

References unmask().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/irq

15.152 L4::Irq_eoi Class Reference

Interface for sending an unmask message to an object.

Inheritance diagram for L4::Irq_eoi:

Collaboration diagram for L4::Irq_eoi:

Public Member Functions

- l4_msgtag_t unmask (unsigned irqnum, l4_umword_t ∗label=0, l4_timeout_t to=L4_IPC_NEVER, l4_utcb_t ∗utcb=l4_utcb()) noexcept

  Unmask the given interrupt line.

15.152.1 Detailed Description

Interface for sending an unmask message to an object.

The object is usually an ICU or an IRQ.

When the kernel receives an IRQ, it masks the interrupt line at the interrupt controller and immediately acknowledges the interrupt. This interface is used to let the kernel know that userspace has dealt with the IRQ. The kernel will unmask the interrupt line and further IRQs can then be delivered.

See also

L4::Icu, L4::Irq

Definition at line 48 of file irq.

15.152.2 Member Function Documentation

15.152.2.1 unmask()

```
l4_msgtag_t L4::Irq_eoi::unmask (
            unsigned irqnum,
            l4_umword_t * label = 0,
            l4_timeout_t to = L4_IPC_NEVER,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Unmask the given interrupt line.

When the object is an IRQ, the given interrupt line is ignored and instead the line which the IRQ is bound to (if any) is unmasked.

Its counterpart for explicitly masking an interrupt line is L4::Icu::mask().

Parameters

	irqnum	The interrupt line that shall be unmasked. Ignored if the object is an IRQ.
out	*label*	If NULL this is a send-only unmask, if not NULL then this operation enters an open wait and the *protected label* shall be received here.
	to	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL `label` only.
	utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag.

Note

If `label` is NULL this function is a send-only operation and there is no return value except for a failed send operation. In this case use l4_ipc_error() to check for errors, **do not** use l4_error(), because l4_error() will always return an error.

Definition at line 76 of file irq.

The documentation for this class was generated from the following file:

- l4/sys/irq

15.153 L4::Irq_handler_object Struct Reference

Server object base class for handling IRQ messages.

Inheritance diagram for L4::Irq_handler_object:

Collaboration diagram for L4::Irq_handler_object:

Additional Inherited Members

15.153.1 Detailed Description

Server object base class for handling IRQ messages.

This server object base class implements the empty interface (L4::Kobject). The implementation of Server_object::dispatch() must return -L4_ENOREPLY, because IRQ messages do not handle replies.

Examples

examples/libs/l4re/c++/shared_ds/ds_srv.cc.

Definition at line 172 of file ipc_server.

The documentation for this struct was generated from the following file:

- l4/cxx/ipc_server

15.154 L4::Irq_mux Struct Reference

IRQ multiplexer for shared IRQs.

Inheritance diagram for L4::Irq_mux:

Collaboration diagram for L4::Irq_mux:

Public Member Functions

- l4_msgtag_t chain (Cap< Triggerable > const &slave, l4_utcb_t *utcb=l4_utcb()) noexcept

  Attach an IRQ to this multiplexer.

Additional Inherited Members

15.154.1 Detailed Description

IRQ multiplexer for shared IRQs.

Deprecated IRQ muxer objects are no longer supported by the kernel.

This interface allows broadcasting of shared IRQs to multiple triggerables. The IRQ multiplexer is responsible for the correct mask and unmask logic for such shared IRQs.

The semantics are that each of the slave IRQs is triggered whenever the multiplexer IRQ is triggered. As shared IRQs are usually level-triggered, the real IRQ source will be masked automatically when an IRQ is delivered and shall be unmasked when all attached slave IRQs are unmasked.

Definition at line 217 of file irq.

15.154.2 Member Function Documentation

15.154.2.1 chain()

```
l4_msgtag_t L4::Irq_mux::chain (
            Cap< Triggerable > const & slave,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Attach an IRQ to this multiplexer.

Parameters

slave	The slave that shall be attached to the master.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be an IRQ multiplexer (Irq_mux) bound to the real IRQ source. This function allows to add slave IRQs to this multiplexer.

Definition at line 232 of file irq.

References L4::Cap_base::cap(), L4::Kobject::cap(), and l4_irq_mux_chain_u().

Here is the call graph for this function:

The documentation for this struct was generated from the following file:

- l4/sys/irq

15.155 L4::Irqep_t< Derived, BASE, bool > Struct Template Reference

Epiface implementation for interrupt handlers.

Inheritance diagram for L4::Irqep_t< Derived, BASE, bool >:

Collaboration diagram for L4::Irqep_t< Derived, BASE, bool >:

Public Member Functions

- l4_msgtag_t dispatch (l4_msgtag_t, unsigned, l4_utcb_t *) final

  The abstract handler for client requests to the object.
- Cap< L4::Irq > obj_cap () const

  Get the (typed) capability to this object.

Additional Inherited Members

15.155.1 Detailed Description

template<typename Derived, typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
struct L4::Irqep_t< Derived, BASE, bool >

Epiface implementation for interrupt handlers.

Template Parameters

Derived	Irq handler implementation class. The class must provide a single function handle_irq().
BASE	Base Epiface class.

Definition at line 290 of file ipc_epiface.

15.155.2 Member Function Documentation

15.155.2.1 dispatch()

```
template<typename Derived , typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
l4_msgtag_t L4::Irqep_t< Derived, BASE, bool >::dispatch (
            l4_msgtag_t tag,
            unsigned rights,
            l4_utcb_t * utcb )  [inline], [final], [virtual]
```

The abstract handler for client requests to the object.

Parameters

tag	The message tag for this invocation.
rights	The rights bits in the invoked capability.
utcb	The UTCB used for the invocation.

Return values

-L4_ENOREPLY	No reply message is send.
<0	Error, reply with error code.
>=0	Success, reply with return value.

This function must be implemented by application specific server objects.

Implements L4::Epiface.

Definition at line 292 of file ipc_epiface.

References L4_ENOREPLY, and l4_msgtag().

Here is the call graph for this function:

15.155.2.2 obj_cap()

```
template<typename Derived , typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
Cap<L4::Irq> L4::Irqep_t< Derived, BASE, bool >::obj_cap ( ) const  [inline]
```

Get the (typed) capability to this object.

Returns

Irq capability for the kernel object behind the server.

Definition at line 302 of file ipc_epiface.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_epiface

15.156 L4::Kip::Mem_desc Class Reference

Memory descriptors stored in the kernel interface page.

Collaboration diagram for L4::Kip::Mem_desc:

Public Types

- enum Mem_type {
  Undefined = 0x0 , Conventional = 0x1 , Reserved = 0x2 , Dedicated = 0x3 ,
  Shared = 0x4 , Info = 0xd , Bootloader = 0xe , Arch = 0xf }

  Memory types.
- enum Info_sub_type { Info_acpi_rsdp = 0 }

  Memory sub types for the Mem_type::Info type.

Public Member Functions

- Mem_desc (unsigned long start, unsigned long end, Mem_type t, unsigned char st=0, bool virt=false) noexcept

  Initialize memory descriptor.
- unsigned long start () const noexcept

  Return start address of memory descriptor.
- unsigned long end () const noexcept

  Return end address of memory descriptor.
- unsigned long size () const noexcept

  Return size of region described by the memory descriptor.
- Mem_type type () const noexcept

  Return type of the memory descriptor.
- unsigned char sub_type () const noexcept

  Return sub-type of the memory descriptor.
- unsigned is_virtual () const noexcept

  Return whether the memory descriptor describes a virtual or physical region.
- void set (unsigned long start, unsigned long end, Mem_type t, unsigned char st=0, bool virt=false) noexcept

  Set values of a memory descriptor.

Static Public Member Functions

- static Mem_desc ∗ first (void ∗kip) noexcept

  Get first memory descriptor.
- static unsigned long count (void const ∗kip) noexcept

  Return number of memory descriptors stored in the kernel info page.
- static void count (void ∗kip, unsigned count) noexcept

  Set number of memory descriptors.
- static cxx::static_vector< Mem_desc const > all (void const ∗kip)

  Return enumerable list of memory descriptors.
- static cxx::static_vector< Mem_desc > all (void ∗kip)

  Return enumerable list of memory descriptors.

15.156.1 Detailed Description

Memory descriptors stored in the kernel interface page.

Include File

```
#include <l4/sys/kip>
```

Definition at line 53 of file kip.

15.156.2 Member Enumeration Documentation

15.156.2.1 Info_sub_type

```
enum L4::Kip::Mem_desc::Info_sub_type
```

Memory sub types for the Mem_type::Info type.

Enumerator

Info_acpi_rsdp	Physical address of the ACPI root pointer.

Definition at line 75 of file kip.

15.156.2.2 Mem_type

```
enum L4::Kip::Mem_desc::Mem_type
```

Memory types.

Enumerator

Undefined	Undefined memory.
Conventional	Conventional memory.
Reserved	Reserved region, do not use this memory.
Dedicated	Dedicated.
Shared	Shared.
Info	Info by boot loader.
Bootloader	Memory belongs to the boot loader.
Arch	Architecture specific memory.

Definition at line 59 of file kip.

15.156.3 Constructor & Destructor Documentation

15.156.3.1 Mem_desc()

```
L4::Kip::Mem_desc::Mem_desc (
            unsigned long start,
            unsigned long end,
            Mem_type t,
            unsigned char st = 0,
            bool virt = false )  [inline], [noexcept]
```

Initialize memory descriptor.

Parameters

start	Start address
end	End address
t	Memory type
st	Memory subtype, defaults to 0
virt	True for virtual memory, false for physical memory, defaults to physical

Definition at line 166 of file kip.

15.156.4 Member Function Documentation

15.156.4.1 all() [1/2]

```
static cxx::static_vector<Mem_desc> L4::Kip::Mem_desc::all (
            void * kip )  [inline], [static]
```

Return enumerable list of memory descriptors.

Parameters

kip	Pointer to the kernel info page.

Definition at line 150 of file kip.

References count(), and first().

Here is the call graph for this function:

15.156.4.2 all() [2/2]

```
static cxx::static_vector<Mem_desc const> L4::Kip::Mem_desc::all (
            void const * kip )  [inline], [static]
```

Return enumerable list of memory descriptors.

Parameters

kip	Pointer to the kernel info page.

Definition at line 139 of file kip.

References count(), and first().

Here is the call graph for this function:

15.156.4.3 count() [1/2]

```
static void L4::Kip::Mem_desc::count (
            void * kip,
            unsigned count )  [inline], [static], [noexcept]
```

Set number of memory descriptors.

Parameters

kip	Pointer to the kernel info page
count	Number of memory descriptors

Definition at line 128 of file kip.

References count().

Here is the call graph for this function:

15.156.4.4 count() [2/2]

```
static unsigned long L4::Kip::Mem_desc::count (
            void const * kip )  [inline], [static], [noexcept]
```

Return number of memory descriptors stored in the kernel info page.

Parameters

kip	Pointer to the kernel info page

Returns

Number of memory descriptors in the kernel info page.

Definition at line 116 of file kip.

Referenced by all(), and count().

Here is the caller graph for this function:

15.156.4.5 end()

```
unsigned long L4::Kip::Mem_desc::end ( ) const  [inline], [noexcept]
```

Return end address of memory descriptor.

Returns

End address of memory descriptor

Definition at line 184 of file kip.

Referenced by set(), and size().

Here is the caller graph for this function:

15.156.4.6 first()

```
static Mem_desc* L4::Kip::Mem_desc::first (
            void * kip )  [inline], [static], [noexcept]
```

Get first memory descriptor.

Parameters

kip	Pointer to the kernel info page

Returns

First memory descriptor stored in the kernel info page

Definition at line 97 of file kip.

Referenced by all().

Here is the caller graph for this function:

15.156.4.7 is_virtual()

```
unsigned L4::Kip::Mem_desc::is_virtual ( ) const  [inline], [noexcept]
```

Return whether the memory descriptor describes a virtual or physical region.

Returns

True for virtual region, false for physical region.

Definition at line 213 of file kip.

15.156.4.8 set()

```
void L4::Kip::Mem_desc::set (
            unsigned long start,
            unsigned long end,
            Mem_type t,
            unsigned char st = 0,
            bool virt = false )  [inline], [noexcept]
```

Set values of a memory descriptor.

Parameters

start	Start address
end	End address
t	Memory type
st	Sub-type, defaults to 0
virt	Virtual or physical memory region, defaults to physical

Definition at line 224 of file kip.

References end(), and start().

Here is the call graph for this function:

15.156.4.9 size()

```
unsigned long L4::Kip::Mem_desc::size ( ) const  [inline], [noexcept]
```

Return size of region described by the memory descriptor.

Returns

Size of the region described by the memory descriptor

Definition at line 191 of file kip.

References end(), and start().

Here is the call graph for this function:

15.156.4.10 start()

```
unsigned long L4::Kip::Mem_desc::start ( ) const  [inline], [noexcept]
```

Return start address of memory descriptor.

Returns

Start address of memory descriptor

Definition at line 177 of file kip.

Referenced by set(), and size().

Here is the caller graph for this function:

15.156.4.11 sub_type()

```
unsigned char L4::Kip::Mem_desc::sub_type ( ) const  [inline], [noexcept]
```

Return sub-type of the memory descriptor.

Returns

Sub-type of the memory descriptor

Definition at line 205 of file kip.

15.156.4.12 type()

```
Mem_type L4::Kip::Mem_desc::type ( ) const  [inline], [noexcept]
```

Return type of the memory descriptor.

Returns

Type of the memory descriptor

Definition at line 198 of file kip.

The documentation for this class was generated from the following file:

- l4/sys/kip

15.157 L4::Kobject Class Reference

Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

Inheritance diagram for L4::Kobject:

Collaboration diagram for L4::Kobject:

Public Member Functions

- l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())

  Decrement the in kernel reference counter for the object.

Protected Member Functions

- l4_cap_idx_t cap () const noexcept

  Return capability selector.

15.157.1 Detailed Description

Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

Include File

```
#include <l4/sys/capability>
```

This is the base class for all remote objects accessible using RPC. However, subclasses doe not directly inherit from L4::Kobject but *must* use L4::Kobject_t (L4::Kobject_0t, L4::Kobject_2t, L4::Kobject_3t, or L4::Kobject_x) for inheritance, otherwise these classes cannot be used as RPC interfaces.

Attention

Objects derived from Kobject *must* never add any data to those objects. Kobjects can act only as proxy object for encapsulating object invocations.

Definition at line 46 of file kobject.

15.157.2 Member Function Documentation

15.157.2.1 cap()

```
l4_cap_idx_t L4::Kobject::cap ( ) const  [inline], [protected], [noexcept]
```

Return capability selector.

Returns

Capability selector.

This method is for derived classes to gain access to the actual capability selector.

Definition at line 79 of file kobject.

Referenced by L4::Icu::bind(), L4::Irq_mux::chain(), L4::Factory::create(), L4::Thread::ex_regs(), L4::Vcon::get_attr(), L4::Icu::info(), L4::Icu::mask(), L4::Thread::modify_senders(), L4::Vcon::read(), L4::Vcon::send(), L4::Vcon::set_attr(), L4::Icu::set_mode(), L4::Triggerable::trigger(), L4::Icu::unbind(), L4::Thread::vcpu_control(), L4::Thread::vcpu_control_ext(), and L4::Vcon::write().

Here is the caller graph for this function:

15.157.2.2 dec_refcnt()

```
l4_msgtag_t L4::Kobject::dec_refcnt (
            l4_mword_t diff,
            l4_utcb_t * utcb = l4_utcb() )  [inline]
```

Decrement the in kernel reference counter for the object.

Parameters

diff	The delta that shall be subtracted from the reference count.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

This function is intended for servers to be able to remove the servers own capability from the counted references. This leads to the semantics that the kernel will delete the object even if the capability of the server is valid. The server can detect the deletion by polling its capabilities or by using the IPC-gate deletion IRQs. And to cleanup if the clients dropped the last reference (capability) to the object.

This function only succeeds on a kernel object of type L4::Ipc_gate which has the server right (L4_FPAGE_C_IPCGATE_SVR). For other kernel objects, -L4_ENOSYS is returned.

Definition at line 110 of file kobject.

The documentation for this class was generated from the following file:

- l4/sys/kobject

15.158 L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference

Helper class to create an L4Re interface class that is derived from two base classes (see `L4::Kobject_t`).

`#include <l4/sys/capability>`

Inheritance diagram for L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >:

Collaboration diagram for L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >:

Protected Types

- typedef Derived Class

  The target interface type (inheriting from Kobject_t)
- typedef Typeid::Iface< PROTO, Derived > __Iface

  The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, Typeid::Merge_list< typename Base1::__Iface↩_list, typename Base2::__Iface_list > > __Iface_list

  The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions

- L4::Cap< Class > c () const noexcept

  Get the capability to ourselves.

Static Protected Member Functions

- static void __check_protocols__ () noexcept

  Helper to check for protocol conflicts.

15.158.1 Detailed Description

template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename S_DEMAND = Type_↩info::Demand_t<>>
class L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >

Helper class to create an L4Re interface class that is derived from two base classes (see `L4::Kobject_t`).

Template Parameters

Derived	is the name of the new interface.
Base1	is the name of the interface's first base class.
Base2	is the name of the interface's second base class.
PROTO	may be set to the statically assigned protocol number used to communicate with this interface.
S_DEMAND	type defining the demand of server-side resources for this interface, usually a L4::Type_info::Demand_t. This value must describe the server-side resources needed by the interface itself, the resource demand of the base interfaces (Base1 and Base2) are automatically included.

The typical usage pattern is shown in the following code snippet. The semantics of this example is an interface My_iface that is derived from L4::Icu and L4Re::Dataspace.

```
class My_iface : public L4::Kobject_2t<My_iface, L4::Icu, L4Re::Dataspace>
{
  ...
};
```

Definition at line 835 of file __typeinfo.h.

15.158.2 Member Typedef Documentation

15.158.2.1 __Iface

```
template<typename Derived , typename Base1 , typename Base2 , long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Iface<PROTO, Derived> L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND
>::__Iface  [protected]
```

The interface description for the derived class.

Definition at line 841 of file __typeinfo.h.

15.158.2.2 __Iface_list

```
template<typename Derived , typename Base1 , typename Base2 , long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Merge_list< Typeid::Iface_list<__Iface>, Typeid::Merge_list< typename Base1↩
::__Iface_list, typename Base2::__Iface_list > > L4::Kobject_2t< Derived, Base1, Base2, PROTO,
S_DEMAND >::__Iface_list  [protected]
```

The list of all RPC interfaces provided directly or through inheritance.

Definition at line 849 of file __typeinfo.h.

15.158.2.3 Class

```
template<typename Derived , typename Base1 , typename Base2 , long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Derived L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::Class  [protected]
```

The target interface type (inheriting from Kobject_t)

Definition at line 839 of file __typeinfo.h.

15.158.3 Member Function Documentation

15.158.3.1 __check_protocols__()

```
template<typename Derived , typename Base1 , typename Base2 , long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
static void L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::__check_protocols__ ( )
[inline], [static], [protected], [noexcept]
```

Helper to check for protocol conflicts.

Definition at line 852 of file __typeinfo.h.

15.158.3.2 c()

```
template<typename Derived , typename Base1 , typename Base2 , long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
L4::Cap<Class> L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::c ( ) const  [inline],
[protected], [noexcept]
```

Get the capability to ourselves.

Definition at line 871 of file __typeinfo.h.

The documentation for this class was generated from the following file:

- l4/sys/__typeinfo.h

15.159 L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference

Helper class to create an L4Re interface class that is derived from three base classes (see `L4::Kobject_t`).

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >:

Collaboration diagram for L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >:

Protected Types

- typedef Derived Class

  The target interface type (inheriting from Kobject_t)
- typedef Typeid::Iface< PROTO, Derived > __Iface

  The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, Typeid::Merge_list< typename Base1::__Iface↩_list, Typeid::Merge_list< typename Base2::__Iface_list, typename Base3::__Iface_list > > > __Iface_list

  The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions

- L4::Cap< Class > c () const noexcept

  Get the capability to ourselves.

Static Protected Member Functions

- static void __check_protocols__ () noexcept

  Helper to check for protocol conflicts.

15.159.1 Detailed Description

template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_ANY, typename S_↩DEMAND = Type_info::Demand_t<>>
struct L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >

Helper class to create an L4Re interface class that is derived from three base classes (see `L4::Kobject_t`).

Template Parameters

Derived	is the name of the new interface.
Base1	is the name of the interface's first base class.
Base2	is the name of the interface's second base class.
Base3	is the name of the interfaces third base class.
PROTO	may be set to the statically assigned protocol number used to communicate with this interface.
S_DEMAND	type defining the demand on server-side resources for this interface, usually a L4::Type_info::Demand_t. This value must describe the server-side resources needed by the interface itself, the resource demand of the base interfaces (Base1 and Base2) are automatically included.

See also

L4::Kobject_t, L4::Kobject_2t, L4::Kobject_0t, L4::Kobject_x

Definition at line 936 of file __typeinfo.h.

15.159.2 Member Typedef Documentation

15.159.2.1 __Iface

```
template<typename Derived , typename Base1 , typename Base2 , typename Base3 , long PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Iface<PROTO, Derived> L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO,
S_DEMAND >::__Iface [protected]
```

The interface description for the derived class.

Definition at line 942 of file __typeinfo.h.

15.159.2.2 __Iface_list

```
template<typename Derived , typename Base1 , typename Base2 , typename Base3 , long PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Merge_list< Typeid::Iface_list<__Iface>, Typeid::Merge_list< typename Base1↩
::__Iface_list, Typeid::Merge_list< typename Base2::__Iface_list, typename Base3::__Iface↩
_list > > > L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::__Iface_list
[protected]
```

The list of all RPC interfaces provided directly or through inheritance.

Definition at line 953 of file __typeinfo.h.

15.159.2.3 Class

```
template<typename Derived , typename Base1 , typename Base2 , typename Base3 , long PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Derived L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::Class  [protected]
```

The target interface type (inheriting from Kobject_t)

Definition at line 940 of file __typeinfo.h.

15.159.3 Member Function Documentation

15.159.3.1 __check_protocols__()

```
template<typename Derived , typename Base1 , typename Base2 , typename Base3 , long PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
static void L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::__check_protocols↩
__ ( )  [inline], [static], [protected], [noexcept]
```

Helper to check for protocol conflicts.

Definition at line 956 of file __typeinfo.h.

15.159.3.2 c()

```
template<typename Derived , typename Base1 , typename Base2 , typename Base3 , long PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
L4::Cap<Class> L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::c ( ) const
[inline], [protected], [noexcept]
```

Get the capability to ourselves.

Definition at line 984 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.160 L4::Kobject_demand< T > Struct Template Reference

Get the combined server-side resource requirements for all type T...

Collaboration diagram for L4::Kobject_demand< T >:

15.160.1 Detailed Description

template<typename ... T>
struct L4::Kobject_demand< T >

Get the combined server-side resource requirements for all type T...

Template Parameters

T	List of IPC interface types for which the combined server-side resource requirements shall be calculated.

Definition at line 1035 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.161 L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference

Helper class to create an L4Re interface class that is derived from a single base class.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >:

Collaboration diagram for L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >:

Protected Types

- typedef Derived Class

  The target interface type (inheriting from Kobject_t)
- typedef Typeid::Iface< PROTO, Derived > __Iface

  The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Base::__Iface_list > __Iface_list

  The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions

- L4::Cap< Class > c () const noexcept

  Get the capability to ourselves.

Static Protected Member Functions

- static void __check_protocols__ () noexcept

  Helper to check for protocol conflicts.

15.161.1 Detailed Description

template<typename Derived, typename Base, long PROTO = PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
class L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >

Helper class to create an L4Re interface class that is derived from a single base class.

Template Parameters

Derived	is the name of the new interface.
Base	is the name of the interfaces single base class.
PROTO	may be set to the statically assigned protocol number used to communicate with this interface.
S_DEMAND	type defining the demand on server-side resources for this interface, usually a L4::Type_info::Demand_t. This value must describe the server-side resources needed by the interface itself, the resource demand of the base interface `Base` is automatically included.

The typical usage pattern is shown in the following code snippet. The semantics of this example is an interface My_iface that is derived from L4::Kobject.

```
class My_iface : public L4::Kobject_t<My_iface, L4::Kobject>
{
  ...
};
```

Definition at line 759 of file __typeinfo.h.

The documentation for this class was generated from the following file:

- l4/sys/__typeinfo.h

15.162 L4::Kobject_typeid< T > Struct Template Reference

Meta object for handling access to type information of Kobjects.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Kobject_typeid< T >:

Public Types

- typedef T::__Kobject_typeid::Demand Demand

  Data type expressing the static demand of receive buffers in a server.

Static Public Member Functions

- static Type_info const ∗ id () noexcept

  Get a pointer to teh Kobject type information of T.
- static Type_info::Demand demand () noexcept

  Get the receive-buffer demand for the server providing the interface T.
- template<typename THIS , typename A1 , typename A2 >
  static int proto_dispatch (THIS ∗self, long proto, A1 a1, A2 &a2)

  Protocol based server-side dispatch function.

15.162.1 Detailed Description

template<typename T>
struct L4::Kobject_typeid< T >

Meta object for handling access to type information of Kobjects.

Template Parameters

T	The data type derived from Kobject, usually using Kobject_t.

Definition at line 620 of file __typeinfo.h.

15.162.2 Member Typedef Documentation

15.162.2.1 Demand

```
template<typename T >
typedef T::__Kobject_typeid::Demand L4::Kobject_typeid< T >::Demand
```

Data type expressing the static demand of receive buffers in a server.

This information is the combined demand of all base interfaces for T and the buffer demand of T itself. The buffer demand of T is usually specified as the S_DEMAND argument of the Kobject_t or Kobject_2t inheritance helpers. S_DEMAND is usually of type L4::Type_info::Demand_t, or L4::Type_info::Demand_union_t.

Definition at line 632 of file __typeinfo.h.

15.162.3 Member Function Documentation

15.162.3.1 demand()

```
template<typename T >
static Type_info::Demand L4::Kobject_typeid< T >::demand ( )  [inline], [static], [noexcept]
```

Get the receive-buffer demand for the server providing the interface T.

Returns

A demand value describing the minimum receive buffers needed for handling server side requests for interface T.

Definition at line 649 of file __typeinfo.h.

15.162.3.2 id()

```
template<typename T >
static Type_info const* L4::Kobject_typeid< T >::id ( )  [inline], [static], [noexcept]
```

Get a pointer to teh Kobject type information of T.

Returns

a pointer to the Kobject typeinfor of T.

Definition at line 640 of file __typeinfo.h.

Referenced by L4::kobject_typeid().

Here is the caller graph for this function:

15.162.3.3 proto_dispatch()

```
template<typename T >
template<typename THIS , typename A1 , typename A2 >
static int L4::Kobject_typeid< T >::proto_dispatch (
            THIS * self,
            long proto,
            A1 a1,
            A2 & a2 )  [inline], [static]
```

Protocol based server-side dispatch function.

Template Parameters

THIS	Data type of the server-side object implementing the interface T.
A1	Data type of second argument for p_dispatch()
A2	Data type of third argument for p_dispatch()

Parameters

self	The pointer to the server object
proto	The protocol number used by the caller
a1	The second argument passed to self->p_dispatch()
a2	The third argument passed to self->p_dispatch()

This function forwards the call to the overloaded p_dispatch() function of self. The data type of the first argument for p_dispatch is determined by the given protocol number.

Definition at line 670 of file __typeinfo.h.

Referenced by L4::Server_object_t< IFACE, BASE >::proto_dispatch().

Here is the caller graph for this function:

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.163 L4::Kobject_typeid< void > Struct Reference

Minimalistic ID for `void` interface.

`#include <__typeinfo.h>`

Collaboration diagram for L4::Kobject_typeid< void >:

15.163.1 Detailed Description

Minimalistic ID for `void` interface.

Definition at line 677 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.164 L4::Kobject_x< Derived, ARGS > Struct Template Reference

Generic Kobject inheritance template.

Inheritance diagram for L4::Kobject_x< Derived, ARGS >:

Collaboration diagram for L4::Kobject_x< Derived, ARGS >:

15.164.1 Detailed Description

template<typename Derived, typename ... ARGS>
struct L4::Kobject_x< Derived, ARGS >

Generic Kobject inheritance template.

Template Parameters

Derived	The class name that derives from Kobject_x.
ARGS	An optional protocol number via L4::Proto_t, followed by an optional server-side requirement passed as L4::Type_info::Demand_t, followed by the list of base classes.

Definition at line 1198 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.165 L4::Meta Class Reference

Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects.

Inheritance diagram for L4::Meta:

Collaboration diagram for L4::Meta:

Public Member Functions

- l4_msgtag_t num_interfaces ()

  Get the number of interfaces implemented by this object.
- l4_msgtag_t interface (l4_umword_t idx, long *proto, L4::Ipc::String< char > *name)

  Get the protocol number that must be used for the interface with the number `idx`.
- l4_msgtag_t supports (l4_mword_t protocol)

  Figure out if the object supports the given protocol (number).

Additional Inherited Members

15.165.1 Detailed Description

Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects.

Definition at line 37 of file meta.

15.165.2 Member Function Documentation

15.165.2.1 interface()

```
l4_msgtag_t L4::Meta::interface (
            l4_umword_t idx,
            long * proto,
            L4::Ipc::String< char > * name )
```

Get the protocol number that must be used for the interface with the number `idx`.

Parameters

	idx	The index of the interface to get the protocol number for. `idx` must be >= 0 and < the return value of num_interfaces().
`out`	*proto*	The protocol number for interface `idx`.
`out`	*name*	The protocol name for interface `idx`.

Return values

l4_msgtag_t::label()	== 0 Successful; see `proto` and `name`.
l4_msgtag_t::label()	< 0 Error code.

15.165.2.2 num_interfaces()

```
l4_msgtag_t L4::Meta::num_interfaces ( )
```

Get the number of interfaces implemented by this object.

Return values

l4_msgtag_t::label()	>= 0 The number of supported interfaces.
l4_msgtag_t::label()	< 0 Error code of the occurred error.

15.165.2.3 supports()

```
l4_msgtag_t L4::Meta::supports (
            l4_mword_t protocol )
```

Figure out if the object supports the given protocol (number).

Parameters

protocol	The protocol number to check for.

Return values

l4_msgtag_t::label()	== 1 protocol is supported.
l4_msgtag_t::label()	== 0 protocol is not supported.

This method is intended to be used for statically assigned protocol numbers.

The documentation for this class was generated from the following file:

- l4/sys/meta

15.166 L4::Out_of_memory Class Reference

Exception signalling insufficient memory.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Out_of_memory:

Collaboration diagram for L4::Out_of_memory:

Public Member Functions

- Out_of_memory (char const ∗extra="") throw ()

  Create an out-of-memory exception.
- ∼Out_of_memory () throw ()

  Destruction.

Additional Inherited Members

15.166.1 Detailed Description

Exception signalling insufficient memory.

Definition at line 188 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

15.167 L4::Pager Class Reference

Pager interface including the Io_pager interface.

Inheritance diagram for L4::Pager:

Collaboration diagram for L4::Pager:

Public Member Functions

- l4_msgtag_t page_fault (l4_umword_t pfa, l4_umword_t pc, L4::Ipc::Rcv_fpage rwin, L4::Ipc::Opt< L4::Ipc::Snd_fpage & > fp)

  Page-fault protocol message.

Additional Inherited Members

15.167.1 Detailed Description

Pager interface including the Io_pager interface.

This class defines the interface for handling page fault IPC. If a thread causes a page fault, the microkernel synthesises a page fault IPC message and sends it to the thread's page fault handler (pager). The pager can then handle the message, for example by establishing a suitable page mapping.

The page fault handler is set with the L4::Thread::control interface.

Definition at line 98 of file pager.

15.167.2 Member Function Documentation

15.167.2.1 page_fault()

```
l4_msgtag_t L4::Pager::page_fault (
            l4_umword_t pfa,
            l4_umword_t pc,
            L4::Ipc::Rcv_fpage rwin,
            L4::Ipc::Opt< L4::Ipc::Snd_fpage & > fp )
```

Page-fault protocol message.

Parameters

	pfa	Faulting address including failure reason: bits [0:2].
	pc	Faulting program counter.
	rwin	Receive window for a flex-page mapping resolving the page fault.
out	*fp*	Optional: flex-page descriptor to send to the task raising the page fault.

Returns

System call message tag; use l4_error() to check for errors.

Page-fault messages are usually generated by the kernel and need to be handled by an appropriate handler function, potentially filling in `fp` for the reply.

`pfa` encoding is as shown:

[63/31 .. 3]	2	1	0
PFA	X	W	r

- **PFA** Bits 63/31..3 of `pfa` are the page fault address bits 63/31 to 3, bits 2..0 are masked.
- **X** Bit 2 of `pfa` if set, indicates a page fault during instruction fetch. Note, this bit is implementation-defined and might always be clear. Therefore, if this bit is clear it does not imply that the page fault is not due to an instruction fetch.
- **W** Bit 1 of `pfa` is set to 1 for a page fault due to a write operation.
- **r** Bit0: reserved, undefined.

The documentation for this class was generated from the following file:

- l4/sys/pager

15.168 L4::Platform_control Class Reference

L4 C++ interface for controlling platform-wide properties, see Platform Control C API for the C interface.

Inheritance diagram for L4::Platform_control:

Collaboration diagram for L4::Platform_control:

Public Member Functions

- l4_msgtag_t system_suspend (l4_umword_t extras)

  Enter suspend to RAM.
- l4_msgtag_t system_shutdown (l4_umword_t reboot)

  Shutdown/Reboot the system.
- l4_msgtag_t cpu_allow_shutdown (l4_umword_t phys_id, l4_umword_t enable)

  Allow CPU shutdown.
- l4_msgtag_t cpu_enable (l4_umword_t phys_id)

  Enable an offline CPU.
- l4_msgtag_t cpu_disable (l4_umword_t phys_id)

  Disable an online CPU.

Additional Inherited Members

15.168.1 Detailed Description

L4 C++ interface for controlling platform-wide properties, see Platform Control C API for the C interface.

Add

```
#include <l4/sys/platform_control>
```

to your code to use the platform control functions. The API allows a client to suspend, reboot or shutdown the system.

For the C interface refer to the Platform Control C API.

Definition at line 47 of file platform_control.

15.168.2 Member Function Documentation

15.168.2.1 cpu_allow_shutdown()

```
l4_msgtag_t L4::Platform_control::cpu_allow_shutdown (
            l4_umword_t phys_id,
            l4_umword_t enable )
```

Allow CPU shutdown.

Parameters

phys↩ _id	Physical CPU id of CPU (e.g. local APIC id) to disable.
enable	Allow shutdown when 1, disallow when 0.

Sets or unsets a hint that a CPU that is not currently used may be powered down.

15.168.2.2 cpu_disable()

```
l4_msgtag_t L4::Platform_control::cpu_disable (
            l4_umword_t phys_id )
```

Disable an online CPU.

Parameters

phys↩ _id	Physical CPU id of CPU (e.g. local APIC id) to disable.

Returns

System call message tag

This function is currently only supported on the ARM EXYNOS platform.

15.168.2.3 cpu_enable()

```
l4_msgtag_t L4::Platform_control::cpu_enable (
            l4_umword_t phys_id )
```

Enable an offline CPU.

Parameters

phys↩ _id	Physical CPU id of CPU (e.g. local APIC id) to enable.

Returns

System call message tag

This function is currently only supported on the ARM EXYNOS platform.

15.168.2.4 system_shutdown()

```
l4_msgtag_t L4::Platform_control::system_shutdown (
            l4_umword_t reboot )
```

Shutdown/Reboot the system.

Parameters

reboot	1 for reboot, 0 for power off

15.168.2.5 system_suspend()

```
l4_msgtag_t L4::Platform_control::system_suspend (
            l4_umword_t extras )
```

Enter suspend to RAM.

Parameters

extras	Some extra platform-specific information needed to enter suspend to RAM. On x86 platforms and when using the Platform_control object provided by Fiasco, the value defines the sleep state. The sleep states are defined in the ACPI table. Other platforms as well as Io's Platform_control object don't make use of this value at the moment.

The documentation for this class was generated from the following file:

- l4/sys/platform_control

15.169 L4::Poll_timeout_kipclock Class Reference

A polling timeout based on the L4Re clock.

Collaboration diagram for L4::Poll_timeout_kipclock:

Public Member Functions

- Poll_timeout_kipclock (unsigned poll_time_us)

  Initialise relative timeout in microseconds.
- void set (unsigned poll_time_us)

  (Re-)Set relative timeout in microseconds
- bool test (bool expression=true)

  Test whether timeout has expired.
- bool timed_out () const

  Query whether timeout has expired.

15.169.1 Detailed Description

A polling timeout based on the L4Re clock.

This class allows to conveniently add a timeout to a polling loop.

The original
```
while (device.read(State) & Busy)
  ;
```

is converted to
```
Poll_timeout_kipclock timeout(10000);
while (timeout.test(device.read(State) & Busy))
  ;
if (timeout.timed_out())
  printf("ERROR: Device does not respond.\n");
```

Definition at line 38 of file poll_timeout_kipclock.

15.169.2 Constructor & Destructor Documentation

15.169.2.1 Poll_timeout_kipclock()

```
L4::Poll_timeout_kipclock::Poll_timeout_kipclock (
            unsigned poll_time_us )  [inline]
```

Initialise relative timeout in microseconds.

Parameters

poll_time_us	Polling timeout in microseconds.

Definition at line 45 of file poll_timeout_kipclock.

References set().

Here is the call graph for this function:

15.169.3 Member Function Documentation

15.169.3.1 set()

```
void L4::Poll_timeout_kipclock::set (
            unsigned poll_time_us )  [inline]
```

(Re-)Set relative timeout in microseconds

Parameters

poll_time_us	Polling timeout in microseconds.

Definition at line 54 of file poll_timeout_kipclock.

References l4_kip_clock(), and l4re_kip().

Referenced by Poll_timeout_kipclock().

Here is the call graph for this function:

Here is the caller graph for this function:

15.169.3.2 test()

```
bool L4::Poll_timeout_kipclock::test (
            bool expression = true )  [inline]
```

Test whether timeout has expired.

Parameters

expression	Optional expression.

Return values

false	The timeout has expired or the given expression returned false.
true	The timeout has not expired and the optionally given expression returns true.

Definition at line 68 of file poll_timeout_kipclock.

References l4_kip_clock(), and l4re_kip().

Here is the call graph for this function:

15.169.3.3 timed_out()

```
bool L4::Poll_timeout_kipclock::timed_out ( ) const  [inline]
```

Query whether timeout has expired.

Returns

Expiry state of timeout

Definition at line 80 of file poll_timeout_kipclock.

The documentation for this class was generated from the following file:

- l4/re/util/poll_timeout_kipclock

15.170 L4::Proto_t< P > Struct Template Reference

Data type for defining protocol numbers.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Proto_t< P >:

15.170.1 Detailed Description

template<long P = PROTO_EMPTY>
struct L4::Proto_t< P >

Data type for defining protocol numbers.

Template Parameters

P	The protocol number itself

This type must be used when specifying a protocol number with Kobject_x.

Definition at line 1182 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.171 L4::Rcv_endpoint Class Reference

Interface for kernel objects that allow to receive IPC from them.

Inheritance diagram for L4::Rcv_endpoint:

Collaboration diagram for L4::Rcv_endpoint:

Public Member Functions

- l4_msgtag_t bind_thread (lpc::Cap< Thread > t, l4_umword_t label)

  Bind a thread to an IPC receive endpoint.

Additional Inherited Members

15.171.1 Detailed Description

Interface for kernel objects that allow to receive IPC from them.

Such an object is for example an lpc_gate (with server rights) or an Irq_sender. Those objects allow to bind a thread that shall receive IPC from these object via bind_thread().

Definition at line 40 of file rcv_endpoint.

15.171.2 Member Function Documentation

15.171.2.1 bind_thread()

```
l4_msgtag_t L4::Rcv_endpoint::bind_thread (
            Ipc::Cap< Thread > t,
            l4_umword_t label )
```

Bind a thread to an IPC receive endpoint.

Parameters

t	Thread object that shall be bound to this receive endpoint.
label	Label to assign to `this` receive endpoint. The two least significant bits should usually be set to zero.

Returns

Syscall return tag containing one of the following return codes.

Return values

L4_EOK	Operation successful.
-L4_EINVAL	`t` is not a thread object or other arguments were malformed.
-L4_EPERM	No L4_CAP_FPAGE_S rights on `t` or the capability used to invoke this operation.

The specified `label` is passed to the receiver of the incoming IPC. It is possible to re-bind a receive endpoint to the same or a different thread. In this case, IPC already in flight will be delivered with the old label to the previously bound thread unless L4::Thread::modify_senders() is used to change these labels.

Referenced by L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/rcv_endpoint

15.172 L4::Registry_iface Class Reference

Abstract interface for object registries.

Inheritance diagram for L4::Registry_iface:

Collaboration diagram for L4::Registry_iface:

Public Member Functions

- virtual L4::Cap< void > register_obj (L4::Epiface *o, char const *service)=0

  Register an L4::Epiface for an IPC gate available in the applications environment under the name `service`.
- virtual L4::Cap< void > register_obj (L4::Epiface *o)=0

  Register o as server-side object for synchronous RPC.
- virtual L4::Cap< L4::Irq > register_irq_obj (L4::Epiface *o)=0

  Register o as server-side object for asynchronous IRQs.
- virtual L4::Cap< L4::Rcv_endpoint > register_obj (L4::Epiface *o, L4::Cap< L4::Rcv_endpoint > ep)=0

  Register o as server-side object for a pre-allocated capability.
- virtual void unregister_obj (L4::Epiface *o, bool unmap=true)=0

  Unregister the given object o from the server.

15.172.1 Detailed Description

Abstract interface for object registries.

An object registry allows to register L4::Epiface objects at a server loop either for synchronous RPC messages or for asynchronous IRQ messages.

Definition at line 330 of file ipc_epiface.

15.172.2 Member Function Documentation

15.172.2.1 register_irq_obj()

```
virtual L4::Cap<L4::Irq> L4::Registry_iface::register_irq_obj (
            L4::Epiface * o )  [pure virtual]
```

Register o as server-side object for asynchronous IRQs.

Parameters

o	Pointer to an Epiface object that shall be registered as server-side object for IRQs.

Return values

L4::Cap<L4::Irq>	Capability to a new IRQ object on success.
L4::Cap<L4::Irq>::Invalid	The allocation of the IRQ has failed.

After successful registration `o->obj_cap()` will be the capability of the allocated IRQ object.

The function may allocate a capability slot for the object. In that case unregister_obj() is responsible for freeing the slot as well.

Implemented in L4Re::Util::Object_registry.

15.172.2.2 register_obj() [1/3]

```
virtual L4::Cap<void> L4::Registry_iface::register_obj (
            L4::Epiface * o )  [pure virtual]
```

Register `o` as server-side object for synchronous RPC.

Parameters

o	Pointer to an Epiface object that shall be registered as server-side object for RPC.

Return values

L4::Cap<void>	A valid capability to a new IPC gate.
L4::Cap<void>::Invalid	The allocation of the IPC gate has failed.

After successful registration `o->obj_cap()` will be the capability of the allocated IPC gate.

The function may allocate a capability slot for the object. In that case unregister_obj() is responsible for freeing the slot as well.

Implemented in L4Re::Util::Object_registry.

15.172.2.3 register_obj() [2/3]

```
virtual L4::Cap<void> L4::Registry_iface::register_obj (
            L4::Epiface * o,
            char const * service )  [pure virtual]
```

Register an L4::Epiface for an IPC gate available in the applications environment under the name `service`.

Parameters

o	Pointer to an Epiface object that shall be registered.
service	Name of the capability that shall be used to connect `o` to as a server-side object.

Return values

L4::Cap<void>	The capability known as `service` on success.
L4::Cap<void>::Invalid	No capability with the given name found.

After a successful call to this function `o->obj_cap()` is equal to the capability in the environment with the name given by `service`.

Implemented in L4Re::Util::Object_registry.

15.172.2.4 register_obj() [3/3]

```
virtual L4::Cap<L4::Rcv_endpoint> L4::Registry_iface::register_obj (
            L4::Epiface * o,
            L4::Cap< L4::Rcv_endpoint > ep )  [pure virtual]
```

Register `o` as server-side object for a pre-allocated capability.

Parameters

o	Pointer to an Epiface object that shall be registered as server-side object.
ep	Capability to an already allocated capability where `o` shall be attached as server-side handler. The capability may point to an IPC gate or an IRQ.

Return values

L4::Cap<L4::Rcv_endpoint>	Capability `ep` on success.
L4::Cap<L4::Rcv_endpoint>::Invalid	The IRQ attach operation has failed.

After successful registration `o->obj_cap()` will be equal to `ep`.

Implemented in L4Re::Util::Object_registry.

15.172.2.5 unregister_obj()

```
virtual void L4::Registry_iface::unregister_obj (
            L4::Epiface * o,
            bool unmap = true )  [pure virtual]
```

Unregister the given object `o` from the server.

Parameters

o	Pointer to the Epiface object that shall be unregistered. The object must have been registered with any of the register methods if Registry_iface.
unmap	If true the capability `o->obj_cap()` shall be unmapped from the local object space.

The function always unmaps and frees the capability if it was allocated by either Registry_iface::register_irq_obj(L4::Epiface ∗), or by Registry_iface::register_obj(L4::Epiface ∗).

Implemented in L4Re::Util::Object_registry.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_epiface

15.173 L4::Runtime_error Class Reference

Exception for an abstract runtime error.

`#include <l4/cxx/exceptions>`

Inheritance diagram for L4::Runtime_error:

Collaboration diagram for L4::Runtime_error:

Public Member Functions

- Runtime_error (long err_no, char const ∗extra=0) throw ()

  Create a new Runtime_error.
- char const ∗ str () const throw ()

  Return a human readable string for the exception.
- char const ∗ extra_str () const

  Get the description text for this runtime error.
- long err_no () const throw ()

  Get the error value for this runtime error.

Additional Inherited Members

15.173.1 Detailed Description

Exception for an abstract runtime error.

This is the base class for a set of exceptions that cover all errors that have a C error value (see l4_error_code_t).

Definition at line 139 of file exceptions.

15.173.2 Constructor & Destructor Documentation

15.173.2.1 Runtime_error()

```
L4::Runtime_error::Runtime_error (
            long err_no,
            char const * extra = 0 ) throw ( )   [inline], [explicit]
```

Create a new Runtime_error.

Parameters

err_no	Error value for this runtime error.
extra	Description of what was happening while the error occured.

Definition at line 152 of file exceptions.

15.173.3 Member Function Documentation

15.173.3.1 err_no()

```
long L4::Runtime_error::err_no ( ) const throw ( )   [inline]
```

Get the error value for this runtime error.

Returns

Error value.

Definition at line 181 of file exceptions.

15.173.3.2 extra_str()

```
char const* L4::Runtime_error::extra_str ( ) const  [inline]
```

Get the description text for this runtime error.

Returns

Pointer to the description string.

Definition at line 173 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

15.174 L4::Scheduler Class Reference

C++ interface of the Scheduler kernel object, see Scheduler for the C interface.

Inheritance diagram for L4::Scheduler:

Collaboration diagram for L4::Scheduler:

Public Member Functions

- l4_msgtag_t info (l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes=nullptr, l4_utcb_t *utcb=l4_utcb()) const noexcept

  Get scheduler information.
- l4_msgtag_t run_thread (Ipc::Cap< Thread > thread, l4_sched_param_t const &sp)

  Run a thread on a Scheduler.

- l4_msgtag_t idle_time (l4_sched_cpu_set_t const &cpus, l4_kernel_clock_t *us)

  Query the idle time (in µs) of a CPU.
- bool is_online (l4_umword_t cpu, l4_utcb_t *utcb=l4_utcb()) const noexcept

  Query if a CPU is online.

Additional Inherited Members

15.174.1 Detailed Description

C++ interface of the Scheduler kernel object, see Scheduler for the C interface.

The Scheduler interface allows a client to manage CPU resources. The API provides functions to query scheduler information, check the online state of CPUs, query CPU idle time and to start threads on defined CPU sets.

The scheduler offers IRQ number 0, which triggers when the number of online cores changes, e.g. due to hotplug events.

The Scheduler interface inherits from L4::Icu and L4::Irq_eoi for managing the scheduler virtual device IRQ which, in contrast to hardware IRQs, implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

It depends on the platform, which hotplug events actually trigger the IRQ. Many platforms only support triggering the IRQ when a CPU core different from the boot CPU goes online.

Include File

```
#include <l4/sys/scheduler>
```

Definition at line 57 of file scheduler.

15.174.2 Member Function Documentation

15.174.2.1 idle_time()

```
l4_msgtag_t L4::Scheduler::idle_time (
            l4_sched_cpu_set_t const & cpus,
            l4_kernel_clock_t * us )
```

Query the idle time (in µs) of a CPU.

Parameters

	cpus	Set of CPUs to query. Only the idle time of the first selected CPU in `cpus.map` is queried.
out	*us*	Idle time of queried CPU in µs.

Return values

0	Success.
-L4_EINVAL	Invalid CPU requested in cpu set.

This function retrieves the idle time in µs of the first selected CPU in `cpus.map`. The idle time is the accumulated time a CPU has spent in the idle thread since its last reset. To calculate a load estimate `l` one has to retrieve the idle time at the beginning (`i1`) and the end (`i2`) of a known time interval `t`. The load is then calculated as l = 1 - (i2 - i1)/t.

The idle time is only defined for online CPUs. Reading the idle time from offline CPUs is undefined and may result in either getting -L4_EINVAL or calculating an estimated (incorrect) load of 1.

Note

The idle time statistics of remote CPUs is updated on context switch events only, hence may not be up-to-date when requested cross-CPU. To get up-to-date idle time you should use a thread running on the same CPU of which the idle time is requested.

15.174.2.2 info()

```
l4_msgtag_t L4::Scheduler::info (
            l4_umword_t * cpu_max,
            l4_sched_cpu_set_t * cpus,
            l4_umword_t * sched_classes = nullptr,
            l4_utcb_t * utcb = l4_utcb() ) const  [inline], [noexcept]
```

Get scheduler information.

Parameters

`out`	*cpu_max*	Maximum number of CPUs ever available. Optional, can be nullptr.
`in,out`	*cpus*	*cpus.offset* is first CPU of interest. *cpus.granularity* (see l4_sched_cpu_set_t). *cpus.map* Bitmap of online CPUs. Pass nullptr to omit this information.
`out`	*sched_classes*	A bitmap of available scheduling classes (see L4_scheduler_classes). Pass nullptr to omit this information.
	utcb	UTCB pointer of the calling thread. This defaults to the UTCB of the current thread.

Return values

0	Success.
-L4_ERANGE	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 86 of file scheduler.

References l4_sched_cpu_set_t::gran_offset, l4_sched_cpu_set_t::map, and cxx::max().

Here is the call graph for this function:

15.174.2.3 is_online()

```
bool L4::Scheduler::is_online (
            l4_umword_t cpu,
            l4_utcb_t * utcb = l4_utcb() ) const  [inline], [noexcept]
```

Query if a CPU is online.

Parameters

cpu	CPU number whose online status should be queried.
utcb	UTCB pointer of the calling thread. Defaults to *l4_utcb()*.

Return values

true	The CPU is online.
false	The CPU is offline

Definition at line 162 of file scheduler.

15.174.2.4 run_thread()

```
l4_msgtag_t L4::Scheduler::run_thread (
            Ipc::Cap< Thread > thread,
            l4_sched_param_t const & sp )
```

Run a thread on a Scheduler.

Parameters

thread	Capability of the thread to run.
sp	Scheduling parameters.

Return values

0	Success.
-L4_EINVAL	Invalid size of the scheduling parameter.

This function launches a thread on a CPU determined by the scheduling parameter `sp.affinity`. A thread can be intentionally stopped by migrating it on an offline or an invalid CPU. The thread is only guaranteed to run if the CPU it is migrated to is currently online.

Note

A scheduler may impose a policy with regard to selecting CPUs. However the scheduler is required to ensure the following two properties:

- Two threads with disjoint CPU sets must be scheduled to different physical CPUs.
- Two threads with a single identical CPU selected in the CPU set must be scheduled to the same physical CPU.

The documentation for this class was generated from the following file:

- l4/sys/scheduler

15.175 L4::Semaphore Struct Reference

C++ Kernel-provided semaphore interface, see Kernel-provided semaphore for the C interface.

Inheritance diagram for L4::Semaphore:

Collaboration diagram for L4::Semaphore:

Public Member Functions

- l4_msgtag_t up (l4_utcb_t *utcb=l4_utcb()) noexcept

  Semaphore up operation (wrapper for trigger()).
- l4_msgtag_t down (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept

  Semaphore down operation.

Additional Inherited Members

15.175.1 Detailed Description

C++ Kernel-provided semaphore interface, see Kernel-provided semaphore for the C interface.

This is the interface for kernel-provided semaphore objects. The object provides the classical functions `up()` and `down()` for counting the semaphore and blocking. The semaphore is a Triggerable with respect to the `up()` function, this means that a semaphore can be bound to an interrupt line at an ICU (L4::Icu) and incoming interrupts increment the semaphore counter.

The `down()` method decrements the semaphore counter and blocks if the counter is already zero. Blocking on a semaphore may—as all blocking operations—either return successfully, or be aborted due to an expired timeout provided to the `down()` operation, or due to an L4::Thread::ex_regs() operation with the L4_THREAD_EX_REGS_CANCEL flag set.

The main reason for using a semaphore instead of an L4::Irq is to ensure that incoming trigger signals do not interfere with any open-wait operations, as used for example in a server loop.

Definition at line 52 of file semaphore.

15.175.2 Member Function Documentation

15.175.2.1 down()

```
l4_msgtag_t L4::Semaphore::down (
            l4_timeout_t timeout = L4_IPC_NEVER,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Semaphore down operation.

Parameters

timeout	Timeout for blocking the semaphore down operation. Note: The receive timeout of this timeout-pair is significant for blocking, the send part is usually non-blocking.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag. Use l4_error() to check for errors.

Return values

-L4_EPERM	No L4_CAP_FPAGE_S right on invoked semaphore capability.

This method decrements the semaphore counter by one, or blocks if the counter is already zero, until either a timeout or cancel condition hits or the counter is increased by an `up()` operation.

Definition at line 86 of file semaphore.

15.175.2.2 up()

```
l4_msgtag_t L4::Semaphore::up (
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Semaphore up operation (wrapper for trigger()).

Parameters

utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Send-only IPC message return tag. Use l4_ipc_error() to check for errors, do **not** use l4_error().

Increases the semaphore counter by one if it is smaller than an unspecified limit. The unspecified limit is guaranteed to be at least 2^31-1.

Definition at line 66 of file semaphore.

References L4::Triggerable::trigger().

Here is the call graph for this function:

The documentation for this struct was generated from the following file:

- l4/sys/semaphore

15.176 L4::Server< LOOP_HOOKS > Class Template Reference

Basic server loop for handling client requests.

Inheritance diagram for L4::Server< LOOP_HOOKS >:

Collaboration diagram for L4::Server< LOOP_HOOKS >:

Public Member Functions

- Server (l4_utcb_t *)

  Initializes the server loop.
- Server ()

  Initializes the server loop.
- template<typename DISPATCH >
  L4_NORETURN void internal_loop (DISPATCH dispatch, l4_utcb_t *)

  The server loop.
- template<typename R >
  L4_NORETURN void loop_noexc (R r, l4_utcb_t *u=l4_utcb())

  Server loop without exception handling.
- template<typename EXC , typename R >
  L4_NORETURN void loop (R r, l4_utcb_t *u=l4_utcb())

  Server loop with internal exception handling.

Protected Member Functions

- l4_msgtag_t reply_n_wait (l4_msgtag_t reply, l4_umword_t *p, l4_utcb_t *)

  Internal implementation for reply and wait.

Additional Inherited Members

15.176.1 Detailed Description

template<typename LOOP_HOOKS = lpc_svr::Default_loop_hooks>
class L4::Server< LOOP_HOOKS >

Basic server loop for handling client requests.

Parameters

LOOP_HOOKS	the server inherits from LOOP_HOOKS and calls the hooks defined in LOOP_HOOKS in the server loop. See lpc_svr::Default_loop_hooks, lpc_svr::Ignore_errors, lpc_svr::Default_timeout, lpc_svr::Compound_reply, and lpc_svr::Br_manager_no_buffers.

This is basically a simple server loop that uses a single message buffer for receiving requests and sending replies. The dispatcher determines how incoming messages are handled.

Definition at line 258 of file ipc_server_loop.

15.176.2 Constructor & Destructor Documentation

15.176.2.1 Server()

```
template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
L4::Server< LOOP_HOOKS >::Server (
            l4_utcb_t * ) [inline], [explicit]
```

Initializes the server loop.

Parameters

utcb	The UTCB of the thread running the server loop.

Note that this variant is deprecated. The UTCB can be specified with the server loop function (l4_utcb() is the default). (2021-10)

Definition at line 271 of file ipc_server_loop.

15.176.3 Member Function Documentation

15.176.3.1 internal_loop()

```
template<typename L >
template<typename DISPATCH >
L4_NORETURN void L4::Server< L >::internal_loop (
            DISPATCH dispatch,
            l4_utcb_t * utcb )  [inline]
```

The server loop.

This function usually never returns, it waits for incoming messages calls the dispatcher, sends a reply and waits again.

Definition at line 345 of file ipc_server_loop.

References l4_msgtag_t::has_error(), L4_ENOREPLY, and l4_msgtag().

Referenced by L4::Server< LOOP_HOOKS >::loop(), and L4::Server< LOOP_HOOKS >::loop_noexc().

Here is the call graph for this function:

Here is the caller graph for this function:

15.176.3.2 loop()

```
template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
template<typename EXC , typename R >
L4_NORETURN void L4::Server< LOOP_HOOKS >::loop (
            R r,
            l4_utcb_t * u = l4_utcb() )  [inline]
```

Server loop with internal exception handling.

This server loop translates L4::Runtime_error exceptions into negative error return codes sent to the caller.

Definition at line 307 of file ipc_server_loop.

References L4::Server< LOOP_HOOKS >::internal_loop().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_server_loop

15.177 L4::Server_object Class Reference

Abstract server object to be used with L4::Server and L4::Basic_registry.

Inheritance diagram for L4::Server_object:

Collaboration diagram for L4::Server_object:

Public Member Functions

- virtual int dispatch (unsigned long rights, Ipc::Iostream &ios)=0

  The abstract handler for client requests to the object.
- l4_msgtag_t dispatch (l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb)

  The abstract handler for client requests to the object.

Additional Inherited Members

15.177.1 Detailed Description

Abstract server object to be used with L4::Server and L4::Basic_registry.

Note

Usually L4::Server_object_t is used as a base class when writing server objects.

This server object provides an abstract interface that is used by the L4::Registry_dispatcher model. You can derive subclasses from this interface and implement application specific server objects.

Definition at line 49 of file ipc_server.

15.177.2 Member Function Documentation

15.177.2.1 dispatch() [1/2]

```
l4_msgtag_t L4::Server_object::dispatch (
            l4_msgtag_t tag,
            unsigned rights,
            l4_utcb_t * utcb )  [inline], [virtual]
```

The abstract handler for client requests to the object.

Parameters

tag	The message tag for this invocation.
rights	The rights bits in the invoked capability.
utcb	The UTCB used for the invocation.

Return values

-L4_ENOREPLY	No reply message is send.
<0	Error, reply with error code.
>=0	Success, reply with return value.

This function must be implemented by application specific server objects.

Implements L4::Epiface.

Definition at line 71 of file ipc_server.

References dispatch().

Here is the call graph for this function:

15.177.2.2 dispatch() [2/2]

```
virtual int L4::Server_object::dispatch (
            unsigned long rights,
            Ipc::Iostream & ios )  [pure virtual]
```

The abstract handler for client requests to the object.

Parameters

rights	The rights bits in the invoked capability.
ios	The Ipc::Iostream for reading the request and writing the reply.

Return values

-L4_ENOREPLY	Instructs the server loop to not send a reply.
<0	Error, reply with error code.
>=0	Success, reply with return value.

This function must be implemented by application specific server objects. The implementation must unmarshall data from the stream (`ios`) and create a reply by marshalling to the stream (`ios`). For details about the IPC stream see IPC stream operators.

Note

You need to extract the complete message from the *ios* stream before inserting any reply data or before doing any function call that may use the UTCB. Otherwise, the incoming message may get lost.

Implemented in L4::Server_object_x< Derived, IFACE, BASE >.

Examples

examples/libs/l4re/c++/shared_ds/ds_srv.cc, and examples/libs/l4re/streammap/server.cc.

Referenced by dispatch().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/ipc_server

15.178 L4::Server_object_t< IFACE, BASE > Struct Template Reference

Base class (template) for server implementing server objects.

Inheritance diagram for L4::Server_object_t< IFACE, BASE >:

Collaboration diagram for L4::Server_object_t< IFACE, BASE >:

Public Types

- typedef IFACE Interface

  Data type of the IPC interface definition.

Public Member Functions

- BASE::Demand get_buffer_demand () const
- int dispatch_meta_request (L4::Ipc::Iostream &ios)

  Implementation of the meta protocol based on IFACE.

Static Public Member Functions

- template<typename THIS >
  static int proto_dispatch (THIS *self, l4_umword_t rights, L4::Ipc::Iostream &ios)

  Implementation of protocol-based dispatch for this server object.

15.178.1 Detailed Description

template<typename IFACE, typename BASE = L4::Server_object>
struct L4::Server_object_t< IFACE, BASE >

Base class (template) for server implementing server objects.

Template Parameters

IFACE	The IPC interface class that defines the interface that shall be implemented.
BASE	The server object base class (usually L4::Server_object).

Examples

examples/libs/l4re/c++/shared_ds/ds_srv.cc, and examples/libs/l4re/streammap/server.cc.

Definition at line 91 of file ipc_server.

15.178.2 Member Function Documentation

15.178.2.1 dispatch_meta_request()

```
template<typename IFACE , typename BASE = L4::Server_object>
int L4::Server_object_t< IFACE, BASE >::dispatch_meta_request (
            L4::Ipc::Iostream & ios )  [inline]
```

Implementation of the meta protocol based on *IFACE*.

Parameters

ios	The IO stream used for receiving the message.

This function can be used to handle incoming L4_PROTO_META protcol requests. The implementation uses the L4::Type_info of *IFACE* to handle the requests. Call this function in the implementation of Server_object::dispatch() when the received message tag has protocol L4_PROTO_META (L4::Meta::Protocol).

Definition at line 110 of file ipc_server.

15.178.2.2 get_buffer_demand()

```
template<typename IFACE , typename BASE = L4::Server_object>
BASE::Demand L4::Server_object_t< IFACE, BASE >::get_buffer_demand ( ) const  [inline], [virtual]
```

Returns

the server-side buffer demand based in *IFACE*.

Implements L4::Epiface.

Definition at line 97 of file ipc_server.

15.178.2.3 proto_dispatch()

```
template<typename IFACE , typename BASE = L4::Server_object>
template<typename THIS >
static int L4::Server_object_t< IFACE, BASE >::proto_dispatch (
            THIS * self,
            l4_umword_t rights,
            L4::Ipc::Iostream & ios )  [inline], [static]
```

Implementation of protocol-based dispatch for this server object.

Parameters

self	The this pointer for the object (inherits from Server_object_t).
rights	The rights from the received IPC (forwarded to p_dispatch()).
ios	The message stream for the incoming and the reply message.

Server objects may call this function from their dispatch() function. This function reads the protocol ID from the message tag and uses the p_dispatch code to dispatch to overloaded p_dispatch functions of self.

Definition at line 125 of file ipc_server.

References l4_msgtag_t::label(), and L4::Kobject_typeid< T >::proto_dispatch().

Referenced by L4::Server_object_x< Derived, IFACE, BASE >::dispatch().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this struct was generated from the following file:

- l4/cxx/ipc_server

15.179 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference

Helper class to implement p_dispatch based server objects.

Inheritance diagram for L4::Server_object_x< Derived, IFACE, BASE >:

Collaboration diagram for L4::Server_object_x< Derived, IFACE, BASE >:

Public Member Functions

- int dispatch (l4_umword_t r, L4::Ipc::Iostream &ios)

  Implementation forwarding to p_dispatch().

Additional Inherited Members

15.179.1 Detailed Description

template<typename Derived, typename IFACE, typename BASE = L4::Server_object>
struct L4::Server_object_x< Derived, IFACE, BASE >

Helper class to implement p_dispatch based server objects.

Template Parameters

Derived	The data type of your server object class.
IFACE	The data type providing the interface definition for the object.
BASE	Optional data-type of the base server object (usually L4::Server_object)

This class implements the standard dispatch() function of L4::Server_object and forwards incoming messages to a set of overloaded p_dispatch() functions. There must be a p_dispatch() function in Derived for each interface provided by IFACE with the signature

```
int p_dispatch(Iface *, unsigned rights, L4::Ipc::Iostream &)
```

that is called for messages with protocol == Iface::Protocol.

Example signature for L4Re::Dataspace is:

```
int p_dispatch(L4Re::Dataspace *, unsigned, L4::Ipc::Iostream &)
```

Definition at line 154 of file ipc_server.

The documentation for this struct was generated from the following file:

- l4/cxx/ipc_server

15.180 L4::Smart_cap< T, SMART > Class Template Reference

Smart capability class.

Inheritance diagram for L4::Smart_cap< T, SMART >:

Collaboration diagram for L4::Smart_cap< T, SMART >:

Public Member Functions

- template<typename O >
  Smart_cap (Cap< O > const &p) noexcept

  Internal constructor, use to generate a capability from a `this` pointer.
- Cap< T > operator-> () const noexcept

  Member access of a `T`.

Additional Inherited Members

15.180.1 Detailed Description

template<typename T, typename SMART>
class L4::Smart_cap< T, SMART >

Smart capability class.

Definition at line 36 of file smart_capability.

15.180.2 Constructor & Destructor Documentation

15.180.2.1 Smart_cap()

```
template<typename T , typename SMART >
template<typename O >
L4::Smart_cap< T, SMART >::Smart_cap (
            Cap< O > const & p )  [inline], [noexcept]
```

Internal constructor, use to generate a capability from a `this` pointer.

Attention

This constructor is only useful to generate a capability from the `this` pointer of an objected that is an L4::Kobject. Do `never` use this constructor for something else!

Parameters

p	The `this` pointer of the Kobject or derived object

Definition at line 73 of file smart_capability.

The documentation for this class was generated from the following file:

- l4/sys/smart_capability

15.181 L4::String Class Reference

A null-terminated string container class.

```
#include <string.h>
```

Collaboration diagram for L4::String:

15.181.1 Detailed Description

A null-terminated string container class.

Definition at line 33 of file string.h.

The documentation for this class was generated from the following file:

- l4/cxx/string.h

15.182 L4::Task Class Reference

C++ interface of the Task kernel object, see Task for the C interface.

Inheritance diagram for L4::Task:

Collaboration diagram for L4::Task:

Public Member Functions

- l4_msgtag_t map (Cap< Task > const &src_task, l4_fpage_t const &snd_fpage, l4_umword_t snd_base, l4_utcb_t *utcb=l4_utcb()) noexcept

  Map resources available in the source task to a destination task.
- l4_msgtag_t unmap (l4_fpage_t const &fpage, l4_umword_t map_mask, l4_utcb_t *utcb=l4_utcb()) noexcept

  Revoke rights from the task.
- l4_msgtag_t unmap_batch (l4_fpage_t const *fpages, unsigned num_fpages, l4_umword_t map_mask, l4_utcb_t *utcb=l4_utcb()) noexcept

  Revoke rights from a task.
- l4_msgtag_t delete_obj (L4::Cap< void > obj, l4_utcb_t *utcb=l4_utcb()) noexcept

  Release capability and delete object.

- l4_msgtag_t release_cap (L4::Cap< void > cap, l4_utcb_t *utcb=l4_utcb()) noexcept

  Release object capability.
- l4_msgtag_t cap_valid (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb()) noexcept

  Check whether a capability is present (refers to an object).
- l4_msgtag_t cap_equal (Cap< void > const &cap_a, Cap< void > const &cap_b, l4_utcb_t *utcb=l4_utcb()) noexcept

  Test whether two capabilities point to the same object with the same rights.
- l4_msgtag_t add_ku_mem (l4_fpage_t const &fpage, l4_utcb_t *utcb=l4_utcb()) noexcept

  Add kernel-user memory.

Additional Inherited Members

15.182.1 Detailed Description

C++ interface of the Task kernel object, see Task for the C interface.

The L4::Task class represents a combination of the address spaces provided by the L4Re micro kernel. A task consists of at least a memory address space and an object address space. On IA32 there is also an IO-port address space associated with an L4::Task.

L4::Task objects are created using the L4::Factory interface.

Include File

```
#include <l4/sys/task>
```

Definition at line 44 of file task.

15.182.2 Member Function Documentation

15.182.2.1 add_ku_mem()

```
l4_msgtag_t L4::Task::add_ku_mem (
            l4_fpage_t const & fpage,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Add kernel-user memory.

Parameters

fpage	Flexpage describing the virtual area the memory goes to.
utcb	UTCP pointer of the calling thread.

Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page (`L4_PAGESIZE`). A portable implementation should not depend on allocations greater than 16KiB to succeed.

Returns

Syscall return tag

Definition at line 215 of file task.

15.182.2.2 cap_equal()

```
l4_msgtag_t L4::Task::cap_equal (
            Cap< void > const & cap_a,
            Cap< void > const & cap_b,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Test whether two capabilities point to the same object with the same rights.

Parameters

cap↩_a	First capability selector to compare.
cap↩_b	Second capability selector to compare.
utcb	Optional: UTCB pointer of the calling thread.

Return values

tag.label()	= 1: `cap_a` and `cap_b` point to the same object.
tag.label()	= 0: The two caps do **not** point to the same object.

Definition at line 196 of file task.

15.182.2.3 cap_valid()

```
l4_msgtag_t L4::Task::cap_valid (
            Cap< void > const & cap,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Check whether a capability is present (refers to an object).

Parameters

cap	Valid capability to check for presence.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Return values

tag.label()	> 0 Capability is present (refers to an object).
tag.label()	== 0 No capability present (void object).

A capability is considered present when it refers to an existing kernel object.

Precondition

`cap` must be a valid capability (i.e. `cap.is_valid() == true`). If you are unsure about the validity of your capability use L4::Cap.validate() instead.

Definition at line 180 of file task.

15.182.2.4 delete_obj()

```
l4_msgtag_t L4::Task::delete_obj (
            L4::Cap< void > obj,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Release capability and delete object.

Parameters

obj	Capability selector of the object to delete.
utcb	UTCB pointer of the calling thread.

Returns

Syscall return tag

Initiates the deletion of the object if `obj` has the delete permission. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference. No error will be reported if the rights are insufficient, however, the capability is removed in all cases from the destination task.

Definition at line 141 of file task.

15.182.2.5 map()

```
l4_msgtag_t L4::Task::map (
            Cap< Task > const & src_task,
            l4_fpage_t const & snd_fpage,
            l4_umword_t snd_base,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Map resources available in the source task to a destination task.

Parameters

src_task	Capability selector of the source task.
snd_fpage	Send flexpage that describes an area in the address space or object space of the source task.
snd_base	Send base that describes an offset in the receive window of the destination task. The lower bits contain additional map control flags (see l4_fpage_cacheability_opt_t for memory mappings, L4_obj_fpage_ctl for object mappings, and L4_MAP_ITEM_GRANT; also see l4_map_control() and l4_map_obj_control()).
utcb	UTCB pointer of the calling thread.

Returns

Syscall return tag.

This method allows for asynchronous transfer of capabilities, memory mappings, and IO-port mappings (on IA32) from one task to another. The destination task is the task referenced by the capability on which the map is invoked, and the receive window is the whole address space of that task.

For more information on spaces and mappings, see Spaces and Mappings. The flexpage API is described in more detail at Flex pages.

Definition at line 75 of file task.

15.182.2.6 release_cap()

```
l4_msgtag_t L4::Task::release_cap (
            L4::Cap< void > cap,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Release object capability.

Parameters

cap	Capability selector of the object to release.
utcb	UTCB pointer of the calling thread.

Returns

Syscall return tag.

This operation unmaps the capability from `this` task.

Note

If the reference counter of the kernel object referenced by `cap` goes down to zero, the deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Definition at line 160 of file task.

15.182.2.7 unmap()

```
l4_msgtag_t L4::Task::unmap (
            l4_fpage_t const & fpage,
            l4_umword_t map_mask,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Revoke rights from the task.

Parameters

fpage	Flexpage that describes an area in one capability space of `this` task
map_mask	Unmap mask, see l4_unmap_flags_t
utcb	UTCB pointer of the calling thread.

Returns

Syscall return tag

This method allows to revoke rights from the destination task. For a flex page describing IO ports or memory, it also revokes rights from all the tasks that got the rights delegated from the destination task (i.e., this operation does a recursive rights revocation). If the set of rights is empty after the revocation, the capability is unmapped.

Note

If the reference counter of a kernel object referenced in `fpage` goes down to zero (as a result of deleting capabilities), the deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Definition at line 102 of file task.

15.182.2.8 unmap_batch()

```
l4_msgtag_t L4::Task::unmap_batch (
            l4_fpage_t const * fpages,
            unsigned num_fpages,
            l4_umword_t map_mask,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Revoke rights from a task.

Parameters

fpages	An array of flexpages. Each item describes an area in one capability space of `this` task.
num_fpages	Number of fpages in the `fpages` array.
map_mask	Unmap mask, see l4_unmap_flags_t.
utcb	UTCB pointer of the calling thread.

Revoke rights for an array of flexpages, see unmap for details.

Precondition

The caller needs to take care that `num_fpages` is not bigger than L4_UTCB_GENERIC_DATA_SIZE - 2.

Definition at line 121 of file task.

The documentation for this class was generated from the following file:

- l4/sys/task

15.183 L4::Thread Class Reference

C++ L4 kernel thread interface, see Thread for the C interface.

Inheritance diagram for L4::Thread:

Collaboration diagram for L4::Thread:

Data Structures

- class Attr

  Thread attributes used for control_commit().
- class Modify_senders

  Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

Public Member Functions

- l4_msgtag_t ex_regs (l4_addr_t ip, l4_addr_t sp, l4_umword_t flags, l4_utcb_t *utcb=l4_utcb()) noexcept

Exchange basic thread registers.

- l4_msgtag_t ex_regs (l4_addr_t *ip, l4_addr_t *sp, l4_umword_t *flags, l4_utcb_t *utcb=l4_utcb()) noexcept

  Exchange basic thread registers and return previous values.
- l4_msgtag_t control (Attr const &attr) noexcept

  Commit the given thread-attributes object.
- l4_msgtag_t switch_to (l4_utcb_t *utcb=l4_utcb()) noexcept

  Switch execution to this thread.
- l4_msgtag_t stats_time (l4_kernel_clock_t *us, l4_utcb_t *utcb=l4_utcb()) noexcept

  Get consumed time of thread in us.
- l4_msgtag_t vcpu_resume_start (l4_utcb_t *utcb=l4_utcb()) noexcept

  Resume from vCPU asynchronous IPC handler, start.
- l4_msgtag_t vcpu_resume_commit (l4_msgtag_t tag, l4_utcb_t *utcb=l4_utcb()) noexcept

  Resume from vCPU asynchronous IPC handler, commit.
- l4_msgtag_t vcpu_control (l4_addr_t vcpu_state, l4_utcb_t *utcb=l4_utcb()) noexcept

  Enable the vCPU feature for the thread.
- l4_msgtag_t vcpu_control_ext (l4_addr_t ext_vcpu_state, l4_utcb_t *utcb=l4_utcb()) noexcept

  Enable the extended vCPU feature for the thread.
- l4_msgtag_t register_del_irq (Cap< Irq > irq, l4_utcb_t *u=l4_utcb()) noexcept

  Register an IRQ that will trigger upon deletion events.
- l4_msgtag_t modify_senders (Modify_senders const &todo) noexcept

  Apply sender modification rules.

Additional Inherited Members

15.183.1 Detailed Description

C++ L4 kernel thread interface, see Thread for the C interface.

The Thread class defines a thread of execution in the L4 context. Usually user-level and kernel threads are mapped 1:1 to each other. Thread kernel objects are created using a factory, see the L4::Factory API (L4::Factory::create()).

Amongst other things an L4::Thread encapsulates:

- CPU state
  - General-purpose registers
  - Program counter
  - Stack pointer
- FPU state
- Scheduling parameters, see the L4::Scheduler API
- Execution state
  - Blocked, Runnable, Running

Thread objects provide an API for

- Thread configuration and manipulation
- Thread switching.

Include File

```
#include <l4/sys/thread>
```

For the C interface see the Thread API.

Definition at line 58 of file thread.

15.183.2 Member Function Documentation

15.183.2.1 control()

```
l4_msgtag_t L4::Thread::control (
            Attr const & attr )  [inline], [noexcept]
```

Commit the given thread-attributes object.

Parameters

attr	the attribute object to commit to the thread.

Definition at line 239 of file thread.

15.183.2.2 ex_regs() [1/2]

```
l4_msgtag_t L4::Thread::ex_regs (
            l4_addr_t * ip,
            l4_addr_t * sp,
            l4_umword_t * flags,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Exchange basic thread registers and return previous values.

Parameters

in,out	*ip*	New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer.
in,out	*sp*	New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer.
in,out	*flags*	Ex-regs flags, see L4_thread_ex_regs_flags, return previous CPU flags of the thread.
	utcb	UTCB to use for this operation.

Returns

System call return tag. [out] parameters are only valid if the function returns successfully. Use l4_error() to check.

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if L4_THREAD_EX_REGS_TRIGGER_EXCEPTION forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

The thread is started using L4::Scheduler::run_thread(). However, if at the time L4::Scheduler::run_thread() is called, the instruction pointer of the thread is invalid, a later call to ex_regs() with a valid instruction pointer might start the thread.

Definition at line 122 of file thread.

References L4::Kobject::cap(), and l4_thread_ex_regs_ret_u().

Here is the call graph for this function:

15.183.2.3 ex_regs() [2/2]

```
l4_msgtag_t L4::Thread::ex_regs (
            l4_addr_t ip,
            l4_addr_t sp,
            l4_umword_t flags,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Exchange basic thread registers.

Parameters

ip	New instruction pointer, use ~0UL to leave the instruction pointer unchanged.
sp	New stack pointer, use ~0UL to leave the stack pointer unchanged.
flags	Ex-regs flags, see L4_thread_ex_regs_flags.
utcb	UTCB to use for this operation.

Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if L4_THREAD_EX_REGS_TRIGGER_EXCEPTION forces an IPC then changes in IP and SP take effect directly after returning from this IPC.

The thread is started using L4::Scheduler::run_thread(). However, if at the time L4::Scheduler::run_thread() is called, the instruction pointer of the thread is invalid, a later call to ex_regs() with a valid instruction pointer might start the thread.

Definition at line 89 of file thread.

References L4::Kobject::cap(), and l4_thread_ex_regs_u().

Here is the call graph for this function:

15.183.2.4 modify_senders()

```
l4_msgtag_t L4::Thread::modify_senders (
            Modify_senders const & todo )  [inline], [noexcept]
```

Apply sender modification rules.

Parameters

todo	Prepared sender modification rules.

Returns

System call return tag.

The modification rules are applied to all IPCs to the thread (whether directly or by IPC gate) that are already in flight, that is that the sender is already blocking on.

See Modify_senders for a detailed description when applying sender modification rules is required.

See also

l4_thread_modify_sender_commit()

Definition at line 448 of file thread.

References L4::Kobject::cap(), l4_ipc_call(), L4_IPC_NEVER, l4_msgtag(), and L4_PROTO_THREAD.

Referenced by L4Re::Util::Object_registry::unregister_obj().

Here is the call graph for this function:

Here is the caller graph for this function:

15.183.2.5 register_del_irq()

```
l4_msgtag_t L4::Thread::register_del_irq (
            Cap< Irq > irq,
            l4_utcb_t * u = l4_utcb() )  [inline], [noexcept]
```

Register an IRQ that will trigger upon deletion events.

Parameters

irq	Capability selector for the IRQ object to be triggered.
u	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

System call return tag containing the return code.

Return values

L4_EPERM	L4_CAP_FPAGE_W missing on `irq`

In case the `irq` is already bound to an interrupt source, it is unbound first. When `irq` is deleted, it will be deregistered first. A registered deletion Irq can only be deregistered by deleting the Irq or the thread.

List of deletion events:

- deletion of one or several IPC gates bound to this thread.

When the deletion event is delivered, there is no indication about which IPC gate was deleted.

See also

l4_thread_register_del_irq

Definition at line 364 of file thread.

15.183.2.6 stats_time()

```
l4_msgtag_t L4::Thread::stats_time (
            l4_kernel_clock_t * us,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Get consumed time of thread in us.

Parameters

`out`	*us*	Consumed time in µs.
	utcb	UTCB of the current thread.

Returns

Syscall return tag.

Definition at line 260 of file thread.

15.183.2.7 switch_to()

```
l4_msgtag_t L4::Thread::switch_to (
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Switch execution to this thread.

Parameters

utcb	the UTCB of the current thread.

Note

The current time slice is inherited to this thread.

Definition at line 249 of file thread.

15.183.2.8 vcpu_control()

```
l4_msgtag_t L4::Thread::vcpu_control (
            l4_addr_t vcpu_state,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Enable the vCPU feature for the thread.

Parameters

vcpu_state	A virtual address pointing to a l4_vcpu_state_t. It must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()).
utcb	UTCB to use for this operation.

Returns

Syscall return tag.

This function enables the vCPU feature of `this` thread

The kernel-user memory starting at `vcpu_state` must be at least 128-byte aligned and must cover the size of l4_vcpu_state_t.

The asynchronous IPC handling is described at vCPU API.

Note

Disabling of the vCPU feature is optional and currently not supported.

Definition at line 305 of file thread.

References L4::Kobject::cap(), and l4_thread_vcpu_control_u().

Here is the call graph for this function:

15.183.2.9 vcpu_control_ext()

```
l4_msgtag_t L4::Thread::vcpu_control_ext (
            l4_addr_t ext_vcpu_state,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Enable the extended vCPU feature for the thread.

Parameters

ext_vcpu_state	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()).
utcb	UTCB to use for this operation.

Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT or AMD's SVM.

This function enables the extended vCPU feature of `this` thread. Enabling the extended vCPU feature also enables the vCPU feature.

The kernel-user memory area starting at `ext_vcpu_state` must be at least 4 KiB aligned and must cover a size of L4_PAGESIZE. It includes the data of l4_vcpu_state_t at offset 0, the extended vCPU state at offset L4_VCPU↩_OFFSET_EXT_STATE, and, on some platforms, the extended vCPU information at offset L4_VCPU_OFFSET_↩EXT_INFOS.

Note

Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.

Disabling of the extended vCPU feature is currently not supported.

Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 338 of file thread.

References L4::Kobject::cap(), and l4_thread_vcpu_control_ext_u().

Here is the call graph for this function:

15.183.2.10 vcpu_resume_commit()

```
l4_msgtag_t L4::Thread::vcpu_resume_commit (
            l4_msgtag_t tag,
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Resume from vCPU asynchronous IPC handler, commit.

The asynchronous IPC handling is described at vCPU API.

See also

l4_thread_vcpu_resume_commit

Definition at line 281 of file thread.

15.183.2.11 vcpu_resume_start()

```
l4_msgtag_t L4::Thread::vcpu_resume_start (
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Resume from vCPU asynchronous IPC handler, start.

The asynchronous IPC handling is described at vCPU API.

See also

l4_thread_vcpu_resume_start

Definition at line 271 of file thread.

The documentation for this class was generated from the following file:

- l4/sys/thread

15.184 L4::Thread::Attr Class Reference

Thread attributes used for control_commit().

Collaboration diagram for L4::Thread::Attr:

Public Member Functions

- Attr (l4_utcb_t *utcb=l4_utcb()) noexcept

  Create a thread-attribute object with the given UTCB.
- void pager (Cap< void > const &pager) noexcept

  Set the pager capability selector.
- Cap< void > pager () noexcept

  Get the capability selector used for page-fault messages.
- void exc_handler (Cap< void > const &exc_handler) noexcept

  Set the exception-handler capability selector.
- Cap< void > exc_handler () noexcept

  Get the capability selector used for exception messages.
- void bind (l4_utcb_t *thread_utcb, Cap< Task > const &task) noexcept

  Bind the thread to a task.
- void alien (int on) noexcept

  Enable alien mode.
- void ux_host_syscall (int on) noexcept

  Allow host system calls on Fiasco-UX.

Friends

- class **L4::Thread**

15.184.1 Detailed Description

Thread attributes used for control_commit().

This class is responsible for initializing various attributes of a thread in a UTCB for the control_commit() method.

Note

Instantiation of this class starts the preparation of the UTCB. Do not invoke any non-Attr functions between the instantiation and the call to `L4::Thread::control()`.

See also

Thread control for some more details.

Definition at line 140 of file thread.

15.184.2 Constructor & Destructor Documentation

15.184.2.1 Attr()

```
L4::Thread::Attr::Attr (
            l4_utcb_t * utcb = l4_utcb() )  [inline], [explicit], [noexcept]
```

Create a thread-attribute object with the given UTCB.

Parameters

utcb	The UTCB to use for the later L4::Thread::control_commit() function. Usually this is the UTCB of the calling thread.

Definition at line 153 of file thread.

15.184.3 Member Function Documentation

15.184.3.1 alien()

```
void L4::Thread::Attr::alien (
            int on )  [inline], [noexcept]
```

Enable alien mode.

Parameters

on	Boolean value defining the state of the feature.

For a thread in alien mode the kernel produces just an exception IPC for each IPC and exception caused by the alien thread instead of handling these events regularly. (Page faults of alien threads and interrupts occuring while the alien thread is running are always handled regularly.) While the alien thread is blocking, the execption handler can inspect and modify the state of the alien thread and potentially also the systemcall arguments. If the exception handler replies with L4_PROTO_ALLOW_SYSCALL as message tag, the kernel handles the next IPC or exception of the alien thread in a regular way. If the exception handler leaves certain thread state unchanged (in particular the instruction pointer), this will be the IPC or exception that caused the call of the exception handler. For a regularly processed IPC or exception of the alien thread the kernel also performs an exception IPC on kernel exit.

This feature can be used to attach a debugger to a thread and trace all object invocations and their results. It could also be used to handle other systems that use the same syscall instruction as L4Re.

Definition at line 221 of file thread.

15.184.3.2 bind()

```
void L4::Thread::Attr::bind (
            l4_utcb_t * thread_utcb,
            Cap< Task > const & task )  [inline], [noexcept]
```

Bind the thread to a task.

Parameters

thread_utcb	The thread's UTCB address within the task it shall be bound to. The address must be aligned (architecture dependent; at least word aligned) and it must point to at least L4_UTCB_OFFSET bytes of kernel-user memory.
task	The task the thread shall be bound to.

Precondition

The thread must not be bound to a task yet.

A thread may execute code in the context of a task if and only if the thread is bound to the task. To actually start execution, L4::Thread::ex_regs() needs to be used. Execution in the context of the task means that the code has access to all the task's resources (and only those). The executed code itself must be one of those resources. A thread can be bound at most once to a task.

Note

The UTCBs of different threads in the same task should not overlap in order to prevent data corruption.

Definition at line 215 of file thread.

15.184.3.3 exc_handler() [1/2]

```
Cap<void> L4::Thread::Attr::exc_handler ( ) [inline], [noexcept]
```

Get the capability selector used for exception messages.

Returns

The capability selector used to send exception messages. The selector is valid in the task the thread is bound to.

Definition at line 191 of file thread.

15.184.3.4 exc_handler() [2/2]

```
void L4::Thread::Attr::exc_handler (
            Cap< void > const & exc_handler ) [inline], [noexcept]
```

Set the exception-handler capability selector.

Parameters

exc_handler	The capability selector that shall be used for exception messages. This capability selector must be valid within the task the thread is bound to.

Definition at line 182 of file thread.

15.184.3.5 pager() [1/2]

```
Cap<void> L4::Thread::Attr::pager ( )  [inline], [noexcept]
```

Get the capability selector used for page-fault messages.

Returns

The capability selector used to send page-fault messages. The selector is valid in the task the thread is bound to.

Definition at line 172 of file thread.

15.184.3.6 pager() [2/2]

```
void L4::Thread::Attr::pager (
            Cap< void > const & pager )  [inline], [noexcept]
```

Set the pager capability selector.

Parameters

pager	The capability selector that shall be used for page-fault messages. This capability selector must be valid within the task the thread is bound to.

Definition at line 163 of file thread.

15.184.3.7 ux_host_syscall()

```
void L4::Thread::Attr::ux_host_syscall (
            int on )  [inline], [noexcept]
```

Allow host system calls on Fiasco-UX.

Precondition

Running on Fiasco-UX.

Definition at line 229 of file thread.

The documentation for this class was generated from the following file:

- l4/sys/thread

15.185 L4::Thread::Modify_senders Class Reference

Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

Collaboration diagram for L4::Thread::Modify_senders:

Public Member Functions

- int add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits) noexcept

  Add a rule.

15.185.1 Detailed Description

Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

Use the add() function to add modification rules, and use modify_senders() to commit. Do not use the UTCB in between as it is used by add() and modify_senders().

This mechanism shall be used to change the source object labels of every pending IPC of an IPC gate or an IRQ if the labels in such pending IPC become invalid for the receiving thread, potentially because:

- a thread was unbound from an IPC gate / IRQ, or
- an IPC gate / IRQ was removed, or
- the label of an IPC gate / IRQ bound to a thread was changed.

It is not required to perform the modify_sender mechanism after an IPC gate or an IRQ was bound to a thread for the first time.

Definition at line 385 of file thread.

15.185.2 Member Function Documentation

15.185.2.1 add()

```
int L4::Thread::Modify_senders::add (
            l4_umword_t match_mask,
            l4_umword_t match,
            l4_umword_t del_bits,
            l4_umword_t add_bits )  [inline], [noexcept]
```

Add a rule.

Parameters

match_mask	Bitmask of bits to match the label.
match	Bitmask that must be equal to the label after applying match_mask.
del_bits	Bits to be deleted from the label.
add_bits	Bits to be added to the label.

Returns

0 on success, <0 on error

In pseudo code: if ((sender_label & match_mask) == match) { sender_label = (sender_label & ~del_bits) | add_bits; }

Only the first match is applied.

See also

l4_thread_modify_sender_add()

Definition at line 418 of file thread.

Referenced by L4Re::Util::Object_registry::unregister_obj().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/thread

15.186 L4::Triggerable Struct Reference

Interface that allows an object to be triggered by some source.

Inheritance diagram for L4::Triggerable:

Collaboration diagram for L4::Triggerable:

Public Member Functions

- l4_msgtag_t trigger (l4_utcb_t *utcb=l4_utcb()) noexcept

  Trigger the object.

Additional Inherited Members

15.186.1 Detailed Description

Interface that allows an object to be triggered by some source.

The interface specifies no semantics for the trigger operation, this is defined by derived objects.

This interface is usually used in conjunction with L4::Icu.

Definition at line 91 of file irq.

15.186.2 Member Function Documentation

15.186.2.1 trigger()

```
l4_msgtag_t L4::Triggerable::trigger (
            l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Trigger the object.

Parameters

utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag for a send-only operation, use l4_ipc_error() to check for errors (**do not** use l4_error()).

Note

This function is a send-only operation, this means there is no return value except for a failed send operation. Use l4_ipc_error() to check for errors, **do not** use l4_error(), because l4_error() will always return an error.

Examples

examples/libs/l4re/c++/shared_ds/ds_clnt.cc.

Definition at line 106 of file irq.

References L4::Kobject::cap(), and l4_irq_trigger_u().

Referenced by L4::Semaphore::up().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this struct was generated from the following file:

- l4/sys/irq

15.187 L4::Type_info Struct Reference

Dynamic Type Information for L4Re Interfaces.

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Type_info:

Data Structures

- class Demand

  Data type for expressing the needed receive buffers at the server-side of an interface.
- struct Demand_t

  Template type statically describing demand of receive buffers.
- struct Demand_union_t

  Template type statically describing the combination of two Demand object.

15.187.1 Detailed Description

Dynamic Type Information for L4Re Interfaces.

This class represents the runtime-dynamic type information for L4Re interfaces, and is not intended to be used directly by applications.

Note

The interface of is subject to changes.

The main use for this info is to be used by the implementation of the L4::cap_dynamic_cast() function.

Definition at line 509 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.188 L4::Type_info::Demand Class Reference

Data type for expressing the needed receive buffers at the server-side of an interface.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Type_info::Demand:

Collaboration diagram for L4::Type_info::Demand:

Public Member Functions

- Demand (unsigned char caps=0, unsigned char flags=0, unsigned char mem=0, unsigned char ports=0) noexcept

  Make Demand object.
- bool no_demand () const noexcept
- Demand operator| (Demand const &rhs) const noexcept

  get the combined demand of this and rhs

Data Fields

- unsigned char caps

  number of capability receive buffers.
- unsigned char flags

  flags, such as the need for timeouts (TBD).
- unsigned char mem

  number of memory receive buffers.
- unsigned char ports

  number of IO-port receive buffers.

15.188.1 Detailed Description

Data type for expressing the needed receive buffers at the server-side of an interface.

Definition at line 516 of file __typeinfo.h.

15.188.2 Constructor & Destructor Documentation

15.188.2.1 Demand()

```
L4::Type_info::Demand::Demand (
            unsigned char caps = 0,
            unsigned char flags = 0,
            unsigned char mem = 0,
            unsigned char ports = 0 )  [inline], [explicit], [noexcept]
```

Make Demand object.

Parameters

caps	number of capability receive buffers
flags	flags, such as the need for timeouts (TBD).
mem	number of memory receive windows.
ports	number of IO-port receive windows.

Definition at line 537 of file __typeinfo.h.

15.188.3 Member Function Documentation

15.188.3.1 no_demand()

```
bool L4::Type_info::Demand::no_demand ( ) const  [inline], [noexcept]
```

Returns

true if there is no demand at all

Definition at line 542 of file __typeinfo.h.

Referenced by L4::Ipc_svr::Br_manager_no_buffers::alloc_buffer_demand().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/__typeinfo.h

15.189 L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference

Template type statically describing demand of receive buffers.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >:

Collaboration diagram for L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >:

Public Types

- enum { Caps = CAPS , Flags = FLAGS , Mem = MEM , Ports = PORTS }

Additional Inherited Members

15.189.1 Detailed Description

template<unsigned char CAPS = 0, unsigned char FLAGS = 0, unsigned char MEM = 0, unsigned char PORTS = 0>
struct L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >

Template type statically describing demand of receive buffers.

Template Parameters

CAPS	number of capability receive buffers needed.
FLAGS	flags, such as the need for timeouts (TBD).
MEM	number of memory receive windows needed.
PORTS	number of IO-port receive windwows needed.

Definition at line 563 of file __typeinfo.h.

15.189.2 Member Enumeration Documentation

15.189.2.1 anonymous enum

```
template<unsigned char CAPS = 0, unsigned char FLAGS = 0, unsigned char MEM = 0, unsigned char
PORTS = 0>
anonymous enum
```

Enumerator

Caps	number of capability receive buffers.
Flags	flags, such as the need for timeouts.
Mem	number of memory receive windows.
Ports	number of IO-port receive windows.

Definition at line 565 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.190 L4::Type_info::Demand_union_t< D1, D2 > Struct Template Reference

Template type statically describing the combination of two Demand object.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Type_info::Demand_union_t< D1, D2 >:

Collaboration diagram for L4::Type_info::Demand_union_t< D1, D2 >:

Additional Inherited Members

15.190.1 Detailed Description

template<typename D1, typename D2>
struct L4::Type_info::Demand_union_t< D1, D2 >

Template type statically describing the combination of two Demand object.

Template Parameters

D1	first demand object.
D2	second demand object.

Definition at line 583 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.191 L4::Typeid::Detail::_Rpcs< OPCODE, O, X > Struct Template Reference

Empty list of RPCs.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::_Rpcs< OPCODE, O, X >:

Collaboration diagram for L4::Typeid::Detail::_Rpcs< OPCODE, O, X >:

15.191.1 Detailed Description

template<typename OPCODE, unsigned O, typename ... X>
struct L4::Typeid::Detail::_Rpcs< OPCODE, O, X >

Empty list of RPCs.

Definition at line 375 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.192 L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::Detail::Rpc< OP, RPCS >.

Collaboration diagram for L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y >:

15.192.1 Detailed Description

template<typename OPCODE, unsigned O, typename R>
template<typename Y>
struct L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y >

Find the given RPC in the list.

Definition at line 409 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.193 L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... > Struct Template Reference

Non-empty list of RPCs.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >:

Data Structures

- struct Rpc

  Find the given RPC in the list.

Public Types

- enum

  The opcode value to use for this RPC, may be bogus if the opcode_type is void.
- typedef _Rpcs type

  The list element itself.
- typedef OPCODE opcode_type

  The data type for the opcode.
- typedef R rpc

  The RPC type L4::Ipc::Msg::Rpc_call or L4::Ipc::Msg::Rpc_inline_call.
- typedef _Rpcs< OPCODE, _Get_opcode< R, O >::value+1, X... >::type next

  The next RPC in the list or Rpcs_end if this is the last.

15.193.1 Detailed Description

template<typename OPCODE, unsigned O, typename R, typename ... X>
struct L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >

Non-empty list of RPCs.

Definition at line 379 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.194 L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::Detail::Rpc< OP, RPCS >.

Collaboration diagram for L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y >:

15.194.1 Detailed Description

template<typename OPCODE, unsigned O, typename R, typename ... X>
template<typename Y>
struct L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y >

Find the given RPC in the list.

Definition at line 392 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.195 L4::Typeid::Detail::Rpcs_end Struct Reference

Internal end-of-list marker.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::Rpcs_end:

Collaboration diagram for L4::Typeid::Detail::Rpcs_end:

15.195.1 Detailed Description

Internal end-of-list marker.

Definition at line 327 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.196 L4::Typeid::P_dispatch< LIST > Struct Template Reference

Use for protocol based dispatch stage.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::_P_dispatch< LIST >.

Collaboration diagram for L4::Typeid::P_dispatch< LIST >:

15.196.1 Detailed Description

template<typename LIST>
struct L4::Typeid::P_dispatch< LIST >

Use for protocol based dispatch stage.

Definition at line 318 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.197 L4::Typeid::Raw_ipc< CLASS > Struct Template Reference

RPCs list for passing raw incoming IPC to the server object.

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Typeid::Raw_ipc< CLASS >:

15.197.1 Detailed Description

template<typename CLASS>
struct L4::Typeid::Raw_ipc< CLASS >

RPCs list for passing raw incoming IPC to the server object.

Template Parameters

CLASS	The type of the interface (e.g., L4::Icu)

This template allows to have fully handcrafted IPC protocols.

Definition at line 422 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.198 L4::Typeid::Rpc_nocode< OPERATION > Struct Template Reference

List of RPCs of an interface using a single operation without an opcode.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc_nocode< OPERATION >:

Collaboration diagram for L4::Typeid::Rpc_nocode< OPERATION >:

15.198.1 Detailed Description

template<typename OPERATION>
struct L4::Typeid::Rpc_nocode< OPERATION >

List of RPCs of an interface using a single operation without an opcode.

Template Parameters

OPERATION	The RPC operation as defined by L4_RPC etc.

Definition at line 464 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.199 L4::Typeid::Rpcs< RPCS > Struct Template Reference

Standard list of RPCs of an interface.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpcs< RPCS >:

Collaboration diagram for L4::Typeid::Rpcs< RPCS >:

15.199.1 Detailed Description

template<typename ... RPCS>
struct L4::Typeid::Rpcs< RPCS >

Standard list of RPCs of an interface.

Template Parameters

RPCS	list of RPC types as defined by L4_RPC etc.

This is the default list for RPC functions of an interface, it uses L4::Opcode as opcode type and uses opcodes starting from 0.

Definition at line 438 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.200 L4::Typeid::Rpcs_code< OPCODE_TYPE > Struct Template Reference

List of RPCs of an interface using a special opcode type.

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Typeid::Rpcs_code< OPCODE_TYPE >:

Data Structures

- struct F

15.200.1 Detailed Description

template<typename OPCODE_TYPE>
struct L4::Typeid::Rpcs_code< OPCODE_TYPE >

List of RPCs of an interface using a special opcode type.

Template Parameters

OPCODE_TYPE	The data type of the opcode.

List for RPC functions of an interface, using OPCODE_TYPE as data type for the opcode, opcodes starting from 0.

Definition at line 449 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.201 L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS >:

Collaboration diagram for L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS >:

15.201.1 Detailed Description

template<typename OPCODE_TYPE>
template<typename ... RPCS>
struct L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS >

Template Parameters

RPCS	list of RPC types as defined by L4_RPC etc.

Definition at line 455 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.202 L4::Typeid::Rpcs_sys< ARG > Struct Template Reference

List of RPCs typically used for kernel interfaces.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpcs_sys< ARG >:

Collaboration diagram for L4::Typeid::Rpcs_sys< ARG >:

15.202.1 Detailed Description

template<typename ... ARG>
struct L4::Typeid::Rpcs_sys< ARG >

List of RPCs typically used for kernel interfaces.

Template Parameters

RPCS	list of RPC types as defined by L4_RPC etc.

This list of RPC functions uses l4_umword_t as type for the opcode as most kernel protocol do.

Definition at line 475 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

15.203 L4::Types::Bool< V > Struct Template Reference

Boolean meta type.

Inheritance diagram for L4::Types::Bool< V >:

Collaboration diagram for L4::Types::Bool< V >:

Public Types

- typedef Bool< V > type

  The meta type itself.

15.203.1 Detailed Description

template<bool V>
struct L4::Types::Bool< V >

Boolean meta type.

Template Parameters

V	The boolean value

Definition at line 300 of file types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/types

15.204 L4::Types::False Struct Reference

False meta value.

Inheritance diagram for L4::Types::False:

Collaboration diagram for L4::Types::False:

Additional Inherited Members

15.204.1 Detailed Description

False meta value.

Definition at line 308 of file types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/types

15.205 L4::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference

Template for defining typical Flags bitmaps.

Collaboration diagram for L4::Types::Flags< BITS_ENUM, UNDERLYING >:

Public Types

- enum None_type { None }

  The none type to get an empty bitmap.
- typedef UNDERLYING value_type

  type of the underlying value
- typedef BITS_ENUM bits_enum_type

  enum type defining a name for each bit
- typedef Flags< BITS_ENUM, UNDERLYING > type

  the Flags<> type itself

Public Member Functions

- Flags (None_type)

  Make an empty bitmap.
- Flags ()

  Make default Flags.
- Flags (BITS_ENUM e)

  Make flags from bit name.
- operator Private_bool * () const

  Support for `if (flags)` syntax (test for non-empty flags).
- bool operator! () const

  Support for `if (!flags)` syntax (test for empty flags).
- type & operator|= (type rhs)

  Support |= of two compatible Flags types.

- type & operator|= (bits_enum_type rhs)

  Support |= of Flags type and bit name.
- type & operator&= (type rhs)

  Support &= of two compatible Flags types.
- type & operator&= (bits_enum_type rhs)

  Support &= of Flags type and bit name.
- type operator~ () const

  Support ~ for Flags types.
- type & clear (bits_enum_type flag)

  Clear the given flag.
- value_type as_value () const

  Get the underlying value.

Static Public Member Functions

- static type from_raw (value_type v)

  Make flags from a raw value of value_type.

Friends

- type operator| (type lhs, type rhs)

  Support | of two compatible Flags types.
- type operator| (type lhs, bits_enum_type rhs)

  Support | of Flags type and bit name.
- type operator& (type lhs, type rhs)

  Support & of two compatible Flags types.
- type operator& (type lhs, bits_enum_type rhs)

  Support & of Flags type and bit name.

15.205.1 Detailed Description

template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
class L4::Types::Flags< BITS_ENUM, UNDERLYING >

Template for defining typical Flags bitmaps.

Template Parameters

BITS_ENUM	enum type that defines a name for each bit in the bitmap. The values of the enum members must be the number of the bit (*not* a mask).
UNDERLYING	The underlying data type used to represent the bitmap.

The resulting data type provides a type-safe version that allows bitwise `and` and `or` operations with the BITS_↩ENUM members. As well as, test for `0`or `!0`.

Example:

```
enum Test_flag
{
```

```
  Do_weak_tests,
  Do_strong_tests
};
typedef L4::Types::Flags<Test_flag> Test_flags;
Test_flags x = Do_weak_tests;
if (x & Do_strong_tests) { ... }
x |= Do_strong_tests;
if (x & Do_strong_tests) { ... }
```

Definition at line 63 of file types.

15.205.2 Member Enumeration Documentation

15.205.2.1 None_type

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
enum L4::Types::Flags::None_type
```

The none type to get an empty bitmap.

Enumerator

None	Use this to get an empty bitmap.

Definition at line 80 of file types.

15.205.3 Constructor & Destructor Documentation

15.205.3.1 Flags() [1/2]

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
L4::Types::Flags< BITS_ENUM, UNDERLYING >::Flags (
            None_type  )  [inline]
```

Make an empty bitmap.

Usually used for implicit conversion from `Flags::None`.

```
Flags x = Flags::None;
```

Definition at line 90 of file types.

15.205.3.2 Flags() [2/2]

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
L4::Types::Flags< BITS_ENUM, UNDERLYING >::Flags (
            BITS_ENUM e )  [inline]
```

Make flags from bit name.

Usually used for implicit conversion for a bit name.
`Test_flags f = Do_strong_tests;`

Definition at line 103 of file types.

15.205.4 Member Function Documentation

15.205.4.1 clear()

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
type& L4::Types::Flags< BITS_ENUM, UNDERLYING >::clear (
            bits_enum_type flag )  [inline]
```

Clear the given flag.

Parameters

flag	The flag that shall be cleared.

`flags.clear(The_flag)` is a shortcut for `flags &= ~Flags(The_flag)`.

Definition at line 154 of file types.

References L4::Types::Flags< BITS_ENUM, UNDERLYING >::operator&=().

Here is the call graph for this function:

15.205.4.2 from_raw()

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
static type L4::Types::Flags< BITS_ENUM, UNDERLYING >::from_raw (
            value_type v )  [inline], [static]
```

Make flags from a raw value of *value_type*.

This function may be used for example in C wrapper code.

Definition at line 110 of file types.

The documentation for this class was generated from the following file:

- l4/sys/cxx/types

15.206 L4::Types::Flags_ops_t< DT > Struct Template Reference

Mixin class to define a set of friend bitwise operators on `DT`.

Inheritance diagram for L4::Types::Flags_ops_t< DT >:

Collaboration diagram for L4::Types::Flags_ops_t< DT >:

Public Member Functions

- DT operator|= (DT r)

  bitwise or assignment for DT
- DT operator&= (DT r)

  bitwise and assignment for DT
- constexpr operator bool () const

  explicit conversion to bool for tests
- constexpr DT operator~ () const

  bitwise negation for DT

Friends

- constexpr friend DT operator| (DT l, DT r)

  bitwise or for DT
- constexpr friend DT operator& (DT l, DT r)

  bitwise and for DT
- constexpr friend bool operator== (DT l, DT r)

  equality for DT
- constexpr friend bool operator!= (DT l, DT r)

  inequality for DT

15.206.1 Detailed Description

template<typename DT>
struct L4::Types::Flags_ops_t< DT >

Mixin class to define a set of friend bitwise operators on `DT`.

Template Parameters

DT	The type usually inheriting from Flags_ops_t with a member *raw* of enum or integral type.

Definition at line 232 of file types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/types

15.207 L4::Types::Flags_t< DT, T > Struct Template Reference

Template type to define a flags type with bitwise operations.

Inheritance diagram for L4::Types::Flags_t< DT, T >:

Collaboration diagram for L4::Types::Flags_t< DT, T >:

Public Member Functions

- Flags_t ()=default

  Default (uninitializing) costructor.
- constexpr Flags_t (T f)

  Explicit initialization from the underlying type.

Data Fields

- T raw

  Raw integral value.

15.207.1 Detailed Description

template<typename DT, typename T>
struct L4::Types::Flags_t< DT, T >

Template type to define a flags type with bitwise operations.

Template Parameters

DT	determinator type to make the resulting type unique (unused).
T	underlying type used to store the bits, usually an integral type.

Definition at line 284 of file types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/types

15.208 L4::Types::Int_for_size< SIZE, bool > Struct Template Reference

Metafunction to get an unsigned integral type for the given size.

Collaboration diagram for L4::Types::Int_for_size< SIZE, bool >:

15.208.1 Detailed Description

template<unsigned SIZE, bool = true>
struct L4::Types::Int_for_size< SIZE, bool >

Metafunction to get an unsigned integral type for the given size.

Template Parameters

SIZE	The size of the integer in bytes.

Definition at line 165 of file types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/types

15.209 L4::Types::Int_for_type< T > Struct Template Reference

Metafunction to get an integral type of the same size as `T`.

Collaboration diagram for L4::Types::Int_for_type< T >:

Public Types

- typedef Int_for_size< sizeof(T)>::type type

  The resulting unsigned integer type with the size like `T`.

15.209.1 Detailed Description

template<typename T>
struct L4::Types::Int_for_type< T >

Metafunction to get an integral type of the same size as `T`.

Template Parameters

T	The type for which an unsigned integral type with the same size is needed.

Definition at line 192 of file types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/types

15.210 L4::Types::Same< A, B > Struct Template Reference

Compare two data types for equality.

Inheritance diagram for L4::Types::Same< A, B >:

Collaboration diagram for L4::Types::Same< A, B >:

Additional Inherited Members

15.210.1 Detailed Description

template<typename A, typename B>
struct L4::Types::Same< A, B >

Compare two data types for equality.

Template Parameters

A	The first data type
B	The second data type

The result is the boolean True if A and B are the same types.

Definition at line 324 of file types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/types

15.211 L4::Types::True Struct Reference

True meta value.

Inheritance diagram for L4::Types::True:

Collaboration diagram for L4::Types::True:

Additional Inherited Members

15.211.1 Detailed Description

True meta value.

Definition at line 312 of file types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/types

15.212 L4::Unknown_error Class Reference

Exception for an unknown condition.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Unknown_error:

Collaboration diagram for L4::Unknown_error:

Public Member Functions

- char const ∗ str () const throw ()

  Return a human readable string for the exception.

Additional Inherited Members

15.212.1 Detailed Description

Exception for an unknown condition.

This error is usually used when a server returns an unknown return state to the client, this may indicate incompatible messages used by the client and the server.

Definition at line 219 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

15.213 L4::Vcon Class Reference

C++ L4 Vcon interface, see Virtual Console for the C interface.

Inheritance diagram for L4::Vcon:

Collaboration diagram for L4::Vcon:

Public Member Functions

- l4_msgtag_t send (char const ∗buf, unsigned size, l4_utcb_t ∗utcb=l4_utcb()) const noexcept

  Send data to `this` virtual console.
- long write (char const ∗buf, unsigned size, l4_utcb_t ∗utcb=l4_utcb()) const noexcept

  Write data to `this` virtual console.
- int read (char ∗buf, unsigned size, l4_utcb_t ∗utcb=l4_utcb()) const noexcept

Read data from `this` virtual console.

- int read_with_flags (char *buf, unsigned size, l4_utcb_t *utcb=l4_utcb()) const noexcept

  Read data from `this` virtual console which also returns flags.

- l4_msgtag_t set_attr (l4_vcon_attr_t const *attr, l4_utcb_t *utcb=l4_utcb()) const noexcept

  Set the attributes of `this` virtual console.

- l4_msgtag_t get_attr (l4_vcon_attr_t *attr, l4_utcb_t *utcb=l4_utcb()) const noexcept

  Get attributes of `this` virtual console.

Additional Inherited Members

15.213.1 Detailed Description

C++ L4 Vcon interface, see Virtual Console for the C interface.

L4::Vcon is a virtual console for simple character-based input and output. The interrupt for read events is provided by the virtual key interrupt.

The Vcon interface inherits from L4::Icu and L4::Irq_eoi for managing the virtual key interrupt which, in contrast to hardware IRQs, implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

A server implementing the virtual console protocol has a queue for input events. When the first input event is added to the empty queue, the virtual key interrupt is triggered. Further events are added to the queue without generating further interrupts. The queue is emptied when a client reads all queued input events.

Include File

```
#include <l4/sys/vcon>
```

See the Virtual Console for the C interface.

Definition at line 56 of file vcon.

15.213.2 Member Function Documentation

15.213.2.1 get_attr()

```
l4_msgtag_t L4::Vcon::get_attr (
            l4_vcon_attr_t * attr,
            l4_utcb_t * utcb = l4_utcb() ) const  [inline], [noexcept]
```

Get attributes of `this` virtual console.

Parameters

out	*attr*	Attribute structure. Contains the attributes after a successful call of this function.
	utcb	UTCB pointer of the calling thread.

Returns

Syscall return tag.

Definition at line 158 of file vcon.

References L4::Kobject::cap(), and l4_vcon_get_attr_u().

Here is the call graph for this function:

15.213.2.2 read()

```
int L4::Vcon::read (
            char * buf,
            unsigned size,
            l4_utcb_t * utcb = l4_utcb() ) const  [inline], [noexcept]
```

Read data from `this` virtual console.

Parameters

out	*buf*	Pointer to data buffer.
	size	Size of the data buffer in bytes.
	utcb	UTCB pointer of the calling thread.

Return values

<0	Error code.
>size	More bytes to read, `size` bytes are in the buffer `buf`.
<=size	Number of bytes read.

Note

Size must not exceed L4_VCON_READ_SIZE.

Definition at line 107 of file vcon.

References L4::Kobject::cap(), and l4_vcon_read_u().

Here is the call graph for this function:

15.213.2.3 read_with_flags()

```
int L4::Vcon::read_with_flags (
            char * buf,
            unsigned size,
            l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]
```

Read data from `this` virtual console which also returns flags.

Parameters

out	*buf*	Pointer to data buffer.
	size	Size of the data buffer in bytes.
	utcb	UTCB pointer of the calling thread.

Return values

<0	Error code.
>size	More bytes to read, `size` bytes are in the buffer `buf`.
<=size	Number of bytes read.

If this function returns a positive value the caller can check the L4_VCON_READ_STAT_BREAK flag bit for a break condition. The bytes read can be obtained by masking the return value with L4_VCON_READ_SIZE_MASK.

If a break condition is signaled, it is always the first event in the transmitted content, i.e. all characters supplied by this read call follow the break condition.

Note

Size must not exceed L4_VCON_READ_SIZE.

Definition at line 132 of file vcon.

15.213.2.4 send()

```
l4_msgtag_t L4::Vcon::send (
            char const * buf,
            unsigned size,
            l4_utcb_t * utcb = l4_utcb() ) const  [inline], [noexcept]
```

Send data to `this` virtual console.

Parameters

buf	Pointer to the data buffer.
size	Size of the data buffer in bytes.
utcb	UTBC pointer of the calling thread.

Returns

Syscall return tag

Note

Size must not exceed L4_VCON_WRITE_SIZE, a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use l4_ipc_error() to check for send errors, do not use l4_error(), as l4_error() will always return an error.

Definition at line 76 of file vcon.

References L4::Kobject::cap(), and l4_vcon_send_u().

Here is the call graph for this function:

15.213.2.5 set_attr()

```
l4_msgtag_t L4::Vcon::set_attr (
            l4_vcon_attr_t const * attr,
            l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]
```

Set the attributes of `this` virtual console.

Parameters

attr	Attribute structure with the attributes for the virtual console.
utcb	UTCB pointer of the calling thread.

Returns

Syscall return tag.

Definition at line 145 of file vcon.

References L4::Kobject::cap(), and l4_vcon_set_attr_u().

Here is the call graph for this function:

15.213.2.6 write()

```
long L4::Vcon::write (
            char const * buf,
            unsigned size,
            l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]
```

Write data to `this` virtual console.

Parameters

buf	Pointer to the data buffer.
size	Size of the data buffer in bytes.
utcb	UTCB pointer of the calling thread.

Return values

<0	Error.
>=0	Number of bytes written to the virtual console.

Definition at line 90 of file vcon.

References L4::Kobject::cap(), and l4_vcon_write_u().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/vcon

15.214 L4::Vm Class Reference

Virtual machine host address space.

Inheritance diagram for L4::Vm:

Collaboration diagram for L4::Vm:

Additional Inherited Members

15.214.1 Detailed Description

Virtual machine host address space.

L4::Vm is a specialisation of L4::Task, used for virtual machines. The microkernel employs an appropriate page-table format for hosting VMs, such as ePT on VT-x. On Arm, it offers a call to make the virtual GICC area available to the VM.

Definition at line 28 of file __vm-arm.h.

The documentation for this class was generated from the following files:

- l4/sys/__vm-arm.h
- l4/sys/vm

15.215 l4_buf_regs_t Struct Reference

Encapsulation of the buffer-registers block in the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for l4_buf_regs_t:

Data Fields

- l4_umword_t bdr

  Buffer descriptor.
- l4_umword_t br [L4_UTCB_GENERIC_BUFFERS_SIZE]

  Buffer registers.

15.215.1 Detailed Description

Encapsulation of the buffer-registers block in the UTCB.

Definition at line 93 of file utcb.h.

The documentation for this struct was generated from the following file:

- l4/sys/utcb.h

15.216 l4_exc_regs_t Struct Reference

UTCB structure for exceptions.

```
#include <utcb.h>
```

Collaboration diagram for l4_exc_regs_t:

Data Fields

- l4_umword_t pfa

  page fault address
- l4_umword_t err

  error code
- l4_umword_t r [13]

  registers
- l4_umword_t sp

  stack pointer
- l4_umword_t ulr

  ulr
- l4_umword_t _dummy1

  dummy
- l4_umword_t pc

  pc
- l4_umword_t cpsr

  cpsr
- l4_umword_t tpidruro

  Thread-ID register.
- l4_umword_t tpidrurw

  Thread-ID register.

- l4_umword_t r15

  r15
- l4_umword_t r14

  r14
- l4_umword_t r13

  r13
- l4_umword_t r12

  r12
- l4_umword_t r11

  r11
- l4_umword_t r10

  r10
- l4_umword_t r9

  r9
- l4_umword_t r8

  r8
- l4_umword_t rdi

  rdi
- l4_umword_t rsi

  rsi
- l4_umword_t rbp

  rbp
- l4_umword_t rbx

  rbx
- l4_umword_t rdx

  rdx
- l4_umword_t rcx

  rcx
- l4_umword_t rax

  rax
- l4_umword_t trapno

  trap number
- l4_umword_t ip

  instruction pointer
- l4_umword_t dummy1

  dummy
- l4_umword_t flags

  rflags
- l4_umword_t ss

  stack segment register
- l4_umword_t es

  es register
- l4_umword_t ds

  ds register
- l4_umword_t gs

  gs register
- l4_umword_t fs

  fs register
- l4_umword_t edi

  edi register
- l4_umword_t esi

esi register

- l4_umword_t ebp

  ebp register
- l4_umword_t ebx

  ebx register
- l4_umword_t edx

  edx register
- l4_umword_t ecx

  ecx register
- l4_umword_t eax

  eax register

15.216.1 Detailed Description

UTCB structure for exceptions.

Examples

examples/sys/aliens/main.c, examples/sys/singlestep/main.c, and examples/sys/start-with-exc/main.c.

Definition at line 38 of file utcb.h.

15.216.2 Field Documentation

15.216.2.1 flags

```
l4_umword_t l4_exc_regs_t::flags
```

rflags

eflags

Definition at line 80 of file utcb.h.

15.216.2.2 ss

```
l4_umword_t l4_exc_regs_t::ss
```

stack segment register

ss register

Definition at line 82 of file utcb.h.

The documentation for this struct was generated from the following file:

- arm/l4/sys/utcb.h

15.217 l4_fpage_t Union Reference

L4 flexpage type.

```
#include <__l4_fpage.h>
```

Collaboration diagram for l4_fpage_t:

Data Fields

- l4_umword_t fpage

  Raw value.
- l4_umword_t raw

  Raw value.

15.217.1 Detailed Description

L4 flexpage type.

Definition at line 83 of file __l4_fpage.h.

The documentation for this union was generated from the following file:

- l4/sys/__l4_fpage.h

15.218 l4_icu_info_t Struct Reference

Info structure for an ICU.

```
#include <icu.h>
```

Inheritance diagram for l4_icu_info_t:

Collaboration diagram for l4_icu_info_t:

Data Fields

- unsigned features

  Feature flags.
- unsigned nr_irqs

  The number of IRQ lines supported by the ICU,.
- unsigned nr_msis

  The number of MSI vectors supported by the ICU,.

15.218.1 Detailed Description

Info structure for an ICU.

This structure contains information about the features of an ICU.

See also

l4_icu_info().

Definition at line 172 of file icu.h.

15.218.2 Field Documentation

15.218.2.1 features

`unsigned l4_icu_info_t::features`

Feature flags.

If L4_ICU_FLAG_MSI is set the ICU supports MSIs.

Definition at line 179 of file icu.h.

Referenced by L4::Icu::Info::supports_msi().

The documentation for this struct was generated from the following file:

- l4/sys/icu.h

15.219 l4_icu_msi_info_t Struct Reference

Info to use for a specific MSI.

`#include <icu.h>`

Collaboration diagram for l4_icu_msi_info_t:

Data Fields

- l4_uint64_t msi_addr

  Value to use as address when sending this MSI.
- l4_uint32_t msi_data

  Value to use as data written to msi_addr, when sending this MSI.

15.219.1 Detailed Description

Info to use for a specific MSI.

Definition at line 193 of file icu.h.

The documentation for this struct was generated from the following file:

- l4/sys/icu.h

15.220 l4_kernel_info_mem_desc_t Struct Reference

Memory descriptor data structure.

```
#include <memdesc.h>
```

Collaboration diagram for l4_kernel_info_mem_desc_t:

15.220.1 Detailed Description

Memory descriptor data structure.

Note

This data type is opaque, and must be accessed by the accessor functions defined in this module.

Definition at line 74 of file memdesc.h.

The documentation for this struct was generated from the following file:

- l4/sys/memdesc.h

15.221 l4_kernel_info_t Struct Reference

L4 Kernel Interface Page.

```
#include <__kip-32bit.h>
```

Collaboration diagram for l4_kernel_info_t:

Data Fields

- l4_uint32_t magic

  Kernel Info Page identifier ("L4µK").
- l4_uint32_t version

  Kernel version.
- l4_uint8_t offset_version_strings

  offset to version string
- l4_uint8_t fill0 [3]

  reserved
- l4_uint8_t kip_sys_calls

  pointer to system calls
- l4_uint8_t fill1 [3]

  reserved
- l4_umword_t scheduler_granularity

  for rounding time slices
- l4_umword_t _res00 [3]

  default_kdebug_end
- l4_umword_t sigma0_esp

  Sigma0 start stack pointer.
- l4_umword_t sigma0_eip

  Sigma0 instruction pointer.

- l4_umword_t _res01 [2]

  reserved
- l4_umword_t sigma1_esp

  Sigma1 start stack pointer.
- l4_umword_t sigma1_eip

  Sigma1 instruction pointer.
- l4_umword_t _res02 [2]

  reserved
- l4_umword_t root_esp

  Root task stack pointer.
- l4_umword_t root_eip

  Root task instruction pointer.
- l4_umword_t _res03 [2]

  reserved
- l4_umword_t _res50 [1]

  reserved
- l4_umword_t mem_info

  memory information
- l4_umword_t _res58 [2]

  reserved
- l4_umword_t _res04 [16]

  reserved
- l4_umword_t _res05 [2]

  reserved
- l4_umword_t frequency_cpu

  CPU frequency in kHz.
- l4_umword_t frequency_bus

  Bus frequency.
- l4_umword_t _res06 [10]

  reserved
- l4_umword_t user_ptr

  user_ptr
- l4_umword_t vhw_offset

  offset to vhw structure
- l4_uint64_t magic

  Kernel Info Page identifier ("L4µK").
- l4_uint64_t version

  Kernel version.
- l4_uint8_t fill2 [7]

  reserved
- l4_uint8_t fill3 [7]

  reserved
- l4_umword_t _res_a0 [1]

  reserved
- l4_umword_t _res_b0 [2]

  reserver

15.221.1 Detailed Description

L4 Kernel Interface Page.

Examples

examples/sys/ux-vhw/main.c.

Definition at line 38 of file __kip-32bit.h.

The documentation for this struct was generated from the following files:

- l4/sys/__kip-32bit.h
- l4/sys/__kip-64bit.h

15.222 l4_msg_regs_t Union Reference

Encapsulation of the message-register block in the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for l4_msg_regs_t:

Data Fields

- l4_umword_t mr [L4_UTCB_GENERIC_DATA_SIZE]

  Message registers.
- l4_uint64_t mr64 [L4_UTCB_GENERIC_DATA_SIZE/(sizeof(l4_uint64_t)/sizeof(l4_umword_t))]

  Message registers 64bit alias.

15.222.1 Detailed Description

Encapsulation of the message-register block in the UTCB.

Examples

examples/sys/utcb-ipc/main.c.

Definition at line 78 of file utcb.h.

The documentation for this union was generated from the following file:

- l4/sys/utcb.h

15.223 l4_msgtag_t Struct Reference

Message tag data structure.

```
#include <types.h>
```

Collaboration diagram for l4_msgtag_t:

Public Member Functions

- long label () const L4_NOTHROW

  Get the protocol value.
- void label (long v) L4_NOTHROW

  Set the protocol value.
- unsigned words () const L4_NOTHROW

  Get the number of untyped words.
- unsigned items () const L4_NOTHROW

  Get the number of typed items.
- unsigned flags () const L4_NOTHROW

  Get the flags value.
- bool is_page_fault () const L4_NOTHROW

  Test if protocol indicates page-fault protocol.
- bool is_preemption () const L4_NOTHROW

  Test if protocol indicates preemption protocol.
- bool is_sys_exception () const L4_NOTHROW

  Test if protocol indicates system-exception protocol.
- bool is_exception () const L4_NOTHROW

  Test if protocol indicates exception protocol.
- bool is_sigma0 () const L4_NOTHROW

  Test if protocol indicates sigma0 protocol.
- bool is_io_page_fault () const L4_NOTHROW

  Test if protocol indicates IO-page-fault protocol.
- unsigned has_error () const L4_NOTHROW

  Test if flags indicate an error.

Data Fields

- l4_mword_t raw

  raw value

15.223.1 Detailed Description

Message tag data structure.

Include File

```
#include <l4/sys/types.h>
```

Describes the details of an IPC operation, in particular which parts of the UTCB have to be transmitted, and also flags to enable real-time and FPU extensions.

The message tag also contains a user-defined label that could be used to specify a protocol ID. Some negative values are reserved for kernel protocols such as page faults and exceptions.

The type must be treated completely opaque.

Examples

examples/libs/l4re/c++/shared_ds/ds_srv.cc, examples/libs/l4re/streammap/server.cc, examples/sys/aliens/main.c, examples/sys/ipc/ipc_example.c, examples/sys/isr/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 162 of file types.h.

15.223.2 Member Function Documentation

15.223.2.1 flags()

```
unsigned l4_msgtag_t::flags ( ) const  [inline]
```

Get the flags value.

The flags are a combination of the flags defined by l4_msgtag_flags.

Definition at line 180 of file types.h.

References raw.

The documentation for this struct was generated from the following file:

- l4/sys/types.h

15.224 l4_sched_cpu_set_t Struct Reference

CPU sets.

```
#include <scheduler.h>
```

Collaboration diagram for l4_sched_cpu_set_t:

Public Member Functions

- unsigned char granularity () const
- unsigned offset () const
- void set (unsigned char granularity, unsigned offset)

  Set offset and granularity.

Data Fields

- l4_umword_t gran_offset

  Combination of granularity and offset.
- l4_umword_t map

  Bitmap of CPUs.

15.224.1 Detailed Description

CPU sets.

Examples

examples/sys/migrate/thread_migrate.cc.

Definition at line 69 of file scheduler.h.

15.224.2 Member Function Documentation

15.224.2.1 granularity()

```
unsigned char l4_sched_cpu_set_t::granularity ( ) const  [inline]
```

Returns

Get granularity value

Definition at line 92 of file scheduler.h.

References gran_offset.

Referenced by set().

Here is the caller graph for this function:

15.224.2.2 offset()

```
unsigned l4_sched_cpu_set_t::offset ( ) const  [inline]
```

Returns

Get offset value

Definition at line 94 of file scheduler.h.

References gran_offset.

Referenced by set().

Here is the caller graph for this function:

15.224.2.3 set()

```
void l4_sched_cpu_set_t::set (
            unsigned char granularity,
            unsigned offset )  [inline]
```

Set offset and granularity.

Parameters

granularity	Granularity in log2 notation.
offset	Offset. Must be a multiple of 2^granularity.

Definition at line 101 of file scheduler.h.

References gran_offset, granularity(), and offset().

Here is the call graph for this function:

15.224.3 Field Documentation

15.224.3.1 gran_offset

```
l4_umword_t l4_sched_cpu_set_t::gran_offset
```

Combination of granularity and offset.

The granularity defines how many CPUs each bit in map describes. And the offset is the number of the first CPU described by the first bit in the bitmap.

Precondition

offset must be a multiple of 2^granularity.

MSB	LSB
8bit granularity	24bit offset ..

Definition at line 83 of file scheduler.h.

Referenced by granularity(), L4::Scheduler::info(), l4_sched_cpu_set(), offset(), and set().

The documentation for this struct was generated from the following file:

- l4/sys/scheduler.h

15.225 l4_sched_param_t Struct Reference

Scheduler parameter set.

```
#include <scheduler.h>
```

Collaboration diagram for l4_sched_param_t:

Data Fields

- l4_sched_cpu_set_t affinity

  CPU affinity.
- l4_umword_t prio

  Priority for scheduling.
- l4_umword_t quantum

  Timeslice in micro seconds.

15.225.1 Detailed Description

Scheduler parameter set.

Examples

examples/sys/aliens/main.c, examples/sys/migrate/thread_migrate.cc, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 153 of file scheduler.h.

The documentation for this struct was generated from the following file:

- l4/sys/scheduler.h

15.226 l4_snd_fpage_t Struct Reference

Send-flex-page types.

```
#include <__l4_fpage.h>
```

Collaboration diagram for l4_snd_fpage_t:

Data Fields

- l4_umword_t snd_base

  Offset in receive window (send base)
- l4_fpage_t fpage

  Source flex-page descriptor.

15.226.1 Detailed Description

Send-flex-page types.

Definition at line 106 of file __l4_fpage.h.

The documentation for this struct was generated from the following file:

- l4/sys/__l4_fpage.h

15.227 l4_thread_regs_t Struct Reference

Encapsulation of the thread-control-register block of the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for l4_thread_regs_t:

Data Fields

- l4_umword_t error

  System call error codes.
- l4_timeout_t xfer

  Message transfer timeout.
- l4_umword_t user [3]

  User values (ignored and preserved by the kernel)

15.227.1 Detailed Description

Encapsulation of the thread-control-register block of the UTCB.

Definition at line 110 of file utcb.h.

The documentation for this struct was generated from the following file:

- l4/sys/utcb.h

15.228 l4_timeout_s Struct Reference

Basic timeout specification.

```
#include <__timeout.h>
```

Collaboration diagram for l4_timeout_s:

Data Fields

- l4_uint16_t t

  timeout value

15.228.1 Detailed Description

Basic timeout specification.

Basically a floating point number with 10 bits mantissa and 5 bits exponent (t = m∗2^e).

If bit 15 == 1 the timeout is absolute and the lower 6 bits encode the index of the UTCB buffer register(s) holding the absolute 64-bit timeout value. On 32-bit systems, two consecutive UTCB buffer registers are used.

Definition at line 47 of file __timeout.h.

The documentation for this struct was generated from the following file:

- l4/sys/__timeout.h

15.229 l4_timeout_t Union Reference

Timeout pair.

`#include <__timeout.h>`

Collaboration diagram for l4_timeout_t:

Data Fields

- l4_uint32_t raw

  raw value

- struct {
  - l4_timeout_s rcv

    receive timeout
  - l4_timeout_s snd

    send timeout

  } p

  combined timeout

15.229.1 Detailed Description

Timeout pair.

For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

Definition at line 59 of file __timeout.h.

The documentation for this union was generated from the following file:

- l4/sys/__timeout.h

15.230 l4_vcon_attr_t Struct Reference

Vcon attribute structure.

```
#include <vcon.h>
```

Collaboration diagram for l4_vcon_attr_t:

Public Member Functions

- void set_raw ()

  Set terminal attributes to disable all special processing.

Data Fields

- l4_umword_t i_flags

  input flags
- l4_umword_t o_flags

  output flags
- l4_umword_t l_flags

  local flags

15.230.1 Detailed Description

Vcon attribute structure.

Definition at line 189 of file vcon.h.

15.230.2 Member Function Documentation

15.230.2.1 set_raw()

```
void l4_vcon_attr_t::set_raw ( ) [inline]
```

Set terminal attributes to disable all special processing.

Removes all flags that would mangle the read or written characters. Also disables echoing and any special processing of characters.

Definition at line 452 of file vcon.h.

References l4_vcon_set_attr_raw().

Here is the call graph for this function:

The documentation for this struct was generated from the following file:

- l4/sys/vcon.h

15.231 l4_vcpu_ipc_regs_t Struct Reference

vCPU message registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4_vcpu_ipc_regs_t:

15.231.1 Detailed Description

vCPU message registers.

Definition at line 64 of file __vcpu-arch.h.

The documentation for this struct was generated from the following file:

- arm/l4/sys/__vcpu-arch.h

15.232 l4_vcpu_regs_t Struct Reference

vCPU registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4_vcpu_regs_t:

Data Fields

- l4_umword_t pfa

  page fault address
- l4_umword_t err

  error code
- l4_umword_t sp

  stack pointer
- l4_umword_t ip

  instruction pointer
- l4_umword_t flags

  eflags
- l4_umword_t tpidruro

  Thread-ID register.
- l4_umword_t tpidrurw

  Thread-ID register.
- l4_umword_t r15

  r15 register
- l4_umword_t r14

  r14 register
- l4_umword_t r13

  r13 register

- l4_umword_t r12

  r12 register
- l4_umword_t r11

  r11 register
- l4_umword_t r10

  r10 register
- l4_umword_t r9

  r9 register
- l4_umword_t r8

  r8 register
- l4_umword_t di

  rdi register
- l4_umword_t si

  rsi register
- l4_umword_t bp

  rbp register
- l4_umword_t bx

  rbx register
- l4_umword_t dx

  rdx register
- l4_umword_t cx

  rcx register
- l4_umword_t ax

  rax register
- l4_umword_t trapno

  trap number
- l4_umword_t cs

  dummy
- l4_umword_t ss

  ss register
- l4_umword_t es

  gs register
- l4_umword_t ds

  fs register
- l4_umword_t gs

  gs register
- l4_umword_t fs

  fs register
- l4_umword_t dummy1

  dummy

15.232.1 Detailed Description

vCPU registers.

Definition at line 39 of file __vcpu-arch.h.

15.232.2 Field Documentation

15.232.2.1 ax

```
l4_umword_t l4_vcpu_regs_t::ax
```

rax register

eax register

Definition at line 68 of file __vcpu-arch.h.

15.232.2.2 bp

```
l4_umword_t l4_vcpu_regs_t::bp
```

rbp register

ebp register

Definition at line 63 of file __vcpu-arch.h.

15.232.2.3 bx

```
l4_umword_t l4_vcpu_regs_t::bx
```

rbx register

ebx register

Definition at line 65 of file __vcpu-arch.h.

15.232.2.4 cx

```
l4_umword_t l4_vcpu_regs_t::cx
```

rcx register

ecx register

Definition at line 67 of file __vcpu-arch.h.

15.232.2.5 di

```
l4_umword_t l4_vcpu_regs_t::di
```

rdi register

edi register

Definition at line 61 of file __vcpu-arch.h.

15.232.2.6 dx

```
l4_umword_t l4_vcpu_regs_t::dx
```

rdx register

edx register

Definition at line 66 of file __vcpu-arch.h.

15.232.2.7 si

```
l4_umword_t l4_vcpu_regs_t::si
```

rsi register

esi register

Definition at line 62 of file __vcpu-arch.h.

The documentation for this struct was generated from the following file:

- arm/l4/sys/__vcpu-arch.h

15.233 l4_vcpu_state_t Struct Reference

State of a vCPU.

```
#include <vcpu.h>
```

Inheritance diagram for l4_vcpu_state_t:

Collaboration diagram for l4_vcpu_state_t:

Data Fields

- l4_vcpu_regs_t r

  Register state.
- l4_vcpu_ipc_regs_t i

  IPC state.
- l4_uint16_t state

Current vCPU state. See L4_vcpu_state_flags.

- l4_uint16_t saved_state

  Saved vCPU state. See L4_vcpu_state_flags.
- l4_uint16_t sticky_flags

  Pending flags. See L4_vcpu_sticky_flags.
- l4_cap_idx_t user_task

  User task to use.
- l4_umword_t entry_sp

  Stack pointer for entry (when coming from user task)
- l4_umword_t entry_ip

  IP for entry.

15.233.1 Detailed Description

State of a vCPU.

Definition at line 80 of file vcpu.h.

The documentation for this struct was generated from the following file:

- l4/sys/vcpu.h

15.234 l4_vhw_descriptor Struct Reference

Virtual hardware devices description.

```
#include <vhw.h>
```

Collaboration diagram for l4_vhw_descriptor:

Data Fields

- l4_uint32_t magic

  Magic.
- l4_uint8_t version

  Version of the descriptor.
- l4_uint8_t count

  Number of entries.
- l4_uint8_t pad1

  padding
- l4_uint8_t pad2

  padding
- struct l4_vhw_entry descs [ ]

  Array of device descriptions.

15.234.1 Detailed Description

Virtual hardware devices description.

Examples

examples/sys/ux-vhw/main.c.

Definition at line 70 of file vhw.h.

The documentation for this struct was generated from the following file:

- l4/sys/vhw.h

15.235 l4_vhw_entry Struct Reference

Description of a device.

```
#include <vhw.h>
```

Collaboration diagram for l4_vhw_entry:

Data Fields

- enum l4_vhw_entry_type type

  Type of virtual hardware.
- l4_uint32_t provider_pid

  Host PID of the VHW provider.
- l4_addr_t mem_start

  Start of memory region.
- l4_addr_t mem_size

  Size of memory region.
- l4_uint32_t irq_no

  IRQ number.
- l4_uint32_t fd

  File descriptor.

15.235.1 Detailed Description

Description of a device.

Examples

examples/sys/ux-vhw/main.c.

Definition at line 55 of file vhw.h.

The documentation for this struct was generated from the following file:

- l4/sys/vhw.h

15.236 l4_vm_svm_vmcb_control_area Struct Reference

VMCB structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4_vm_svm_vmcb_control_area:

15.236.1 Detailed Description

VMCB structure for SVM VMs.

Definition at line 39 of file __vm-svm.h.

The documentation for this struct was generated from the following file:

- l4/sys/__vm-svm.h

15.237 l4_vm_svm_vmcb_state_save_area Struct Reference

State save area structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4_vm_svm_vmcb_state_save_area:

15.237.1 Detailed Description

State save area structure for SVM VMs.

Definition at line 96 of file __vm-svm.h.

The documentation for this struct was generated from the following file:

- l4/sys/__vm-svm.h

15.238 l4_vm_svm_vmcb_state_save_area_seg Struct Reference

State save area segment selector struct.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4_vm_svm_vmcb_state_save_area_seg:

15.238.1 Detailed Description

State save area segment selector struct.

Definition at line 84 of file __vm-svm.h.

The documentation for this struct was generated from the following file:

- l4/sys/__vm-svm.h

15.239 l4_vm_svm_vmcb_t Struct Reference

Control structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4_vm_svm_vmcb_t:

15.239.1 Detailed Description

Control structure for SVM VMs.

Definition at line 165 of file __vm-svm.h.

The documentation for this struct was generated from the following file:

- l4/sys/__vm-svm.h

15.240 l4_vm_tz_state Struct Reference

state structure for TrustZone VMs

```
#include <vm.h>
```

Collaboration diagram for l4_vm_tz_state:

15.240.1 Detailed Description

state structure for TrustZone VMs

Definition at line 52 of file vm.h.

The documentation for this struct was generated from the following file:

- arm/l4/sys/vm.h

15.241 L4Re::Cap_alloc Class Reference

Capability allocator interface.

Inherited by L4Re::Cap_alloc_t< ALLOC >.

Collaboration diagram for L4Re::Cap_alloc:

Public Member Functions

- virtual L4::Cap< void > alloc () noexcept=0

  Allocate a capability.
- template<typename T >
  L4::Cap< T > alloc () noexcept

  Allocate a capability.
- virtual void free (L4::Cap< void > cap, l4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_↩flags=L4_FP_ALL_SPACES) noexcept=0

  Free a capability.
- virtual ~Cap_alloc ()=0

  Destructor.

Static Public Member Functions

- template<typename CAP_ALLOC >
  static L4Re::Cap_alloc * get_cap_alloc (CAP_ALLOC &ca)

  Construct an instance of a capability allocator.

15.241.1 Detailed Description

Capability allocator interface.

Definition at line 41 of file cap_alloc.

15.241.2 Member Function Documentation

15.241.2.1 alloc() [1/2]

```
template<typename T >
L4::Cap<T> L4Re::Cap_alloc::alloc ( ) [inline], [noexcept]
```

Allocate a capability.

Returns

Capability of type T

Definition at line 64 of file cap_alloc.

References alloc().

Here is the call graph for this function:

15.241.2.2 alloc() [2/2]

```
virtual L4::Cap<void> L4Re::Cap_alloc::alloc ( ) [pure virtual], [noexcept]
```

Allocate a capability.

Returns

Capability of type void

Referenced by alloc(), get_cap_alloc(), L4Re::make_shared_cap(), L4Re::make_shared_del_cap(), L4Re::make_unique_cap(), and L4Re::make_unique_del_cap().

Here is the caller graph for this function:

15.241.2.3 free()

```
virtual void L4Re::Cap_alloc::free (
            L4::Cap< void > cap,
            l4_cap_idx_t task = L4_INVALID_CAP,
            unsigned unmap_flags = L4_FP_ALL_SPACES )  [pure virtual], [noexcept]
```

Free a capability.

Parameters

cap	Capability to free.
task	If set, task to unmap the capability from.
unmap_flags	Flags for unmap, see l4_unmap_flags_t.

Referenced by get_cap_alloc().

Here is the caller graph for this function:

15.241.2.4 get_cap_alloc()

```
template<typename CAP_ALLOC >
static L4Re::Cap_alloc* L4Re::Cap_alloc::get_cap_alloc (
            CAP_ALLOC & ca )  [inline], [static]
```

Construct an instance of a capability allocator.

Parameters

ca	Capability allocator

Returns

Instance of a capability allocator.

Definition at line 90 of file cap_alloc.

References alloc(), free(), L4_FP_ALL_SPACES, and L4_INVALID_CAP.

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/re/cap_alloc

15.242 L4Re::Console Class Reference

Console class.

Inheritance diagram for L4Re::Console:

Collaboration diagram for L4Re::Console:

Additional Inherited Members

15.242.1 Detailed Description

Console class.

Definition at line 39 of file console.

The documentation for this class was generated from the following file:

- l4/re/console

15.243 L4Re::Dataspace Class Reference

Interface for memory-like objects.

Inheritance diagram for L4Re::Dataspace:

Collaboration diagram for L4Re::Dataspace:

Data Structures

- struct F

  Dataspace flags definitions.
- struct Stats

  Information about the dataspace.

Public Member Functions

- long map (Offset offset, Flags flags, Map_addr local_addr, Map_addr min_addr, Map_addr max_addr) const noexcept

  Request a flex-page mapping from the dataspace.
- long map_region (Offset offset, Flags flags, Map_addr min_addr, Map_addr max_addr) const noexcept

  Map a part of a dataspace into a local memory area.

- long clear (Offset offset, Size size)

  Clear parts of a dataspace.
- long allocate (Offset offset, Size size)

  Allocate a range in the dataspace.
- long copy_in (Offset dst_offs, L4::Ipc::Cap< Dataspace > src, Offset src_offs, Size size)

  Copy contents from another dataspace.
- Size size () const noexcept

  Get size of a dataspace.
- Flags flags () const noexcept

  Get flags of the dataspace.
- long info (Stats *stats)

  Get information on the dataspace.

Additional Inherited Members

15.243.1 Detailed Description

Interface for memory-like objects.

Dataspaces are a central abstraction provided by L4Re. A dataspace is an abstraction for any thing that is available via usual memory access instructions. A dataspace can be a file, as well as the memory-mapped registers of a device, or anonymous memory, such as a heap.

The dataspace interface defines a set of methods that allow any kind of dataspace to be attached (mapped) to the virtual address space of an L4 task and then be accessed via memory-access instructions. The L4Re::Rm interface can be used to attach a dataspace to a virtual address space of a task paged by a certain instance of a region map.

Include File

```
#include <l4/re/dataspace>
```

Examples

examples/libs/l4re/c++/mem_alloc/ma+rm.cc, examples/libs/l4re/c++/shared_ds/ds_clnt.cc, and examples/libs/l4re/c++/shared_

Definition at line 60 of file dataspace.

15.243.2 Member Function Documentation

15.243.2.1 allocate()

```
long L4Re::Dataspace::allocate (
            Offset offset,
            Size size )
```

Allocate a range in the dataspace.

Parameters

offset	Offset in the dataspace, in bytes.
size	Size of the range, in bytes.

Return values

L4_EOK	Success
-L4_ERANGE	Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.)
-L4_ENOMEM	Not enough memory available.
<0	IPC errors

On success, at least the given range is guaranteed to be allocated. The dataspace manager may also allocate more memory due to page granularity.

The memory is allocated with the same rights as the dataspace capability.

15.243.2.2 clear()

```
long L4Re::Dataspace::clear (
            Offset offset,
            Size size )
```

Clear parts of a dataspace.

Parameters

offset	Offset within dataspace (in bytes).
size	Size of region to clear (in bytes).

Return values

>=0	Success.
-L4_ERANGE	Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.)
-L4_EACCESS	Dataspace is read-only.
<0	IPC errors

Zeroes out the memory. Depending on the type of memory the memory could also be deallocated and replaced by a shared zero-page.

15.243.2.3 copy_in()

```
long L4Re::Dataspace::copy_in (
            Offset dst_offs,
            L4::Ipc::Cap< Dataspace > src,
```

```
Offset src_offs,
Size size )
```

Copy contents from another dataspace.

Parameters

dst_offs	Offset in destination dataspace.
src	Source dataspace to copy from.
src_offs	Offset in the source dataspace.
size	Size to copy (in bytes).

Return values

L4_EOK	Success
-L4_EACCESS	Destination dataspace not writable.
-L4_EINVAL	Invalid parameter supplied.
<0	IPC errors

The copy operation may use copy-on-write mechanisms. The operation may also fail if both dataspaces are not from the same dataspace manager or the dataspace managers do not cooperate.

15.243.2.4 flags()

```
Dataspace::Flags L4Re::Dataspace::flags ( ) const  [noexcept]
```

Get flags of the dataspace.

Return values

>=0	Flags of the dataspace
<0	IPC errors

See also

L4Re::Dataspace::F::Flags

Definition at line 119 of file dataspace_impl.h.

Referenced by L4vbus::Vbus::assign_dma_domain().

Here is the caller graph for this function:

15.243.2.5 info()

```
long L4Re::Dataspace::info (
            Stats * stats )
```

Get information on the dataspace.

Parameters

out	*stats*	Dataspace information

Return values

0	Success
<0	IPC errors

15.243.2.6 map()

```
long L4Re::Dataspace::map (
            Dataspace::Offset offset,
            Dataspace::Flags flags,
            Dataspace::Map_addr local_addr,
            Dataspace::Map_addr min_addr,
            Dataspace::Map_addr max_addr ) const [noexcept]
```

Request a flex-page mapping from the dataspace.

Parameters

offset	Offset to start within dataspace
flags	Dataspace flags, see L4Re::Dataspace::F::Flags.
local_addr	Local address to map to.
min_addr	Defines start of receive window. (Rounded down to page size.)
max_addr	Defines end of receive window. (Rounded up to page size.)

Return values

L4_EOK	Success
-L4_ERANGE	Invalid offset.
-L4_EPERM	Insufficient permission to map with requested rights.
<0	IPC errors

The map call will attempt to map the largest possible flexpage that covers the given local address and still fits into the region defined by `min_addr` and `max_addr`. If the given region is invalid or does not overlap the local address, the smallest valid page size is used.

Definition at line 94 of file dataspace_impl.h.

References L4_LOG2_PAGESIZE.

15.243.2.7 map_region()

```
long L4Re::Dataspace::map_region (
            Dataspace::Offset offset,
            Dataspace::Flags flags,
            Dataspace::Map_addr min_addr,
            Dataspace::Map_addr max_addr ) const  [noexcept]
```

Map a part of a dataspace into a local memory area.

Parameters

offset	Offset to start within dataspace.
flags	Dataspace flags, see L4Re::Dataspace::F::Flags.
min_addr	(Inclusive) start of the receive area.
max_addr	(Exclusive) end of receive area.

Return values

L4_EOK	Success
-L4_ERANGE	Invalid offset or receive area larger than the dataspace.
-L4_EPERM	Insufficient permission to map with requested rights.
<0	IPC errors

This is a convenience function which maps flex-pages consecutively into the given memory area in the local task. The area is expected to be filled completely. If the dataspace is not large enough to provide the mappings for the entire size of the area, then an error is returned. Mappings may or may not have been already established at that point.

`offset` and `min_addr` are rounded down to the next `L4_PAGESIZE` boundary when necessary. `max_addr` is rounded up to the page boundary. If the resulting maximum address is less or equal than the minimum address, then the function is a noop.

Definition at line 55 of file dataspace_impl.h.

References L4_LOG2_PAGESIZE, L4_UNLIKELY, L4::round_order(), and L4::trunc_order().

Here is the call graph for this function:

15.243.2.8 size()

```
Dataspace::Size L4Re::Dataspace::size ( ) const  [noexcept]
```

Get size of a dataspace.

Returns

Size of the dataspace in bytes.

Examples

examples/libs/l4re/c++/shared_ds/ds_clnt.cc.

Definition at line 109 of file dataspace_impl.h.

The documentation for this class was generated from the following files:

- l4/re/dataspace
- l4/re/impl/dataspace_impl.h

15.244 L4Re::Dataspace::F Struct Reference

Dataspace flags definitions.

Collaboration diagram for L4Re::Dataspace::F:

Public Types

- enum { Caching_shift = 4 }
- enum Flags {
  R = L4_FPAGE_RO , Ro = L4_FPAGE_RO , RW = L4_FPAGE_RW , W = L4_FPAGE_W ,
  X = L4_FPAGE_X , **RX** = L4_FPAGE_RX , **RWX** = L4_FPAGE_RWX , Rights_mask = 0x0f ,
  Normal = 0x00 , Cacheable = Normal , Bufferable = 0x10 , Uncacheable = 0x20 ,
  Caching_mask = 0x30 }

  Flags for map operations.

15.244.1 Detailed Description

Dataspace flags definitions.

Definition at line 67 of file dataspace.

15.244.2 Member Enumeration Documentation

15.244.2.1 anonymous enum

`anonymous enum`

Enumerator

Caching_shift	shift value for caching flags

Definition at line 69 of file dataspace.

15.244.2.2 Flags

`enum L4Re::Dataspace::F::Flags`

Flags for map operations.

Enumerator

R	Request read-only mapping.
Ro	Request read-only mapping.
RW	Request writable mapping (R + W)
W	Request write-only memory.
Rights_mask	All rights bits available for mappings.
Normal	request normal memory mapping
Cacheable	request normal memory mapping
Bufferable	request bufferable (write buffered) mappings
Uncacheable	request uncacheable memory mappings
Caching_mask	mask for caching flags

Definition at line 77 of file dataspace.

The documentation for this struct was generated from the following file:

- l4/re/dataspace

15.245 L4Re::Dataspace::Stats Struct Reference

Information about the dataspace.

Collaboration diagram for L4Re::Dataspace::Stats:

Data Fields

- Size size

  size
- Flags flags

  flags

15.245.1 Detailed Description

Information about the dataspace.

Definition at line 129 of file dataspace.

The documentation for this struct was generated from the following file:

- l4/re/dataspace

15.246 L4Re::Debug_obj Class Reference

Debug interface.

Inheritance diagram for L4Re::Debug_obj:

Collaboration diagram for L4Re::Debug_obj:

Public Member Functions

- long debug (unsigned long function)

  Debug call.

Additional Inherited Members

15.246.1 Detailed Description

Debug interface.

See also

Debugging API .

Definition at line 51 of file debug.

15.246.2 Member Function Documentation

15.246.2.1 debug()

```
long L4Re::Debug_obj::debug (
            unsigned long function )
```

Debug call.

Parameters

function	Function to call.

Returns

- L4_EOK
  - IPC errors

An object can provide a number of debug functions, each identified by some integer. This method is used to fan out to these functions from a common entry point.

The documentation for this class was generated from the following file:

- l4/re/debug

15.247 L4Re::Dma_space Class Reference

DMA Address Space.

Inherits L4::Kobject_0t< Derived, PROTO, S_DEMAND >.

Collaboration diagram for L4Re::Dma_space:

Public Types

- enum Direction { Bidirectional , To_device , From_device , None }

  Direction of the DMA transfers.
- enum Attribute { No_sync }

  Attributes used for the memory region during the transfer.
- enum Space_attrib { Coherent , Phys_space }

  Attributes assigned to the DMA space when associated with a specific device.
- typedef l4_uint64_t Dma_addr

  Data type for DMA addresses.
- typedef L4::Types::Flags< Attribute > Attributes

  Attributes for DMA mappings.
- typedef L4::Types::Flags< Space_attrib > Space_attribs

  Attributes used when configuring the DMA space.

Public Member Functions

- long map (L4::Ipc::Cap< L4Re::Dataspace > src, L4Re::Dataspace::Offset offset, L4::Ipc::In_out< l4_size_t * > size, Attributes attrs, Direction dir, Dma_addr *dma_addr)

  Map the given part of this data space into the DMA address space.
- long unmap (Dma_addr dma_addr, l4_size_t size, Attributes attrs, Direction dir)

  Unmap the given part of this data space from the DMA address space.
- long associate (L4::Ipc::Opt< L4::Ipc::Cap< L4::Task > > dma_task, Space_attribs attr)

  Associate a DMA task for a device to this Dma_space.
- long disassociate ()

  Disassociate the DMA task from this Dma_space.

15.247.1 Detailed Description

DMA Address Space.

A Dma_space represents the DMA address space of a device. Whenever a device needs direct access to parts of an L4Re::Dataspace, that part of the data space must be mapped to the DMA address space that is assigned to that device. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a DMA address space, using map(), makes the given parts of the data space visible to the associated device at the returned DMA address. As long as the memory is mapped into a DMA space it is 'pinned' and cannot be subject to dynamic memory management such as swapping. Additionally, map() is responsible for the necessary syncing operations before the DMA.

unmap() is the reverse operation to map() and unmaps the given data-space part for the DMA address space. unmap() is responsible for the necessary sync operations after the DMA.

Definition at line 57 of file dma_space.

15.247.2 Member Typedef Documentation

15.247.2.1 Attributes

```
typedef L4::Types::Flags<Attribute> L4Re::Dma_space::Attributes
```

Attributes for DMA mappings.

See also

Attribute

Definition at line 102 of file dma_space.

15.247.3 Member Enumeration Documentation

15.247.3.1 Attribute

```
enum L4Re::Dma_space::Attribute
```

Attributes used for the memory region during the transfer.

See also

Attributes

Enumerator

No_sync	Do not sync the memory hierarchy. When this flag is *not set* (default) the memory region shall be made coherent to the point-of-coherency of the device associated with this Dma_space. When using this attribute the client is responsible for syncing the memory hierarchy for DMA. This can either be done using the cache API or by another map() or unmap() operation of the same part of the data space (without the No_sync attribute).

Definition at line 81 of file dma_space.

15.247.3.2 Direction

```
enum L4Re::Dma_space::Direction
```

Direction of the DMA transfers.

Enumerator

Bidirectional	device reads and writes to the memory
To_device	device reads the memory
From_device	device writes to the memory
None	device is coherently connected to the memory

Definition at line 69 of file dma_space.

15.247.3.3 Space_attrib

```
enum L4Re::Dma_space::Space_attrib
```

Attributes assigned to the DMA space when associated with a specific device.

See also

Space_attribs

Enumerator

Coherent	The device is connected coherently with the cache. This means that the map() and unmap() do not need to sync CPU caches before and after DMA.
Phys_space	The DMA space has no DMA task assigned and uses the CPUs physical memory.

Definition at line 109 of file dma_space.

15.247.4 Member Function Documentation

15.247.4.1 associate()

```
long L4Re::Dma_space::associate (
            L4::Ipc::Opt< L4::Ipc::Cap< L4::Task > > dma_task,
            Space_attribs attr )
```

Associate a DMA task for a device to this Dma_space.

Precondition

requires capability rights: {RW}

Parameters

in	*dma_task*	The DMA task used for the device that shall be associated with this DMA space. The dma_task might be an invalid capability when L4Re::Dma_space::Phys_space is set in `attr`, in this case the CPUs physical memory is used as DMA address space.
in	*attr*	Attributes for this DMA space. See L4Re::Dma_space::Space_attrib.

15.247.4.2 disassociate()

```
long L4Re::Dma_space::disassociate ( )
```

Disassociate the DMA task from this Dma_space.

Precondition

requires capability rights: {RW}

15.247.4.3 map()

```
long L4Re::Dma_space::map (
            L4::Ipc::Cap< L4Re::Dataspace > src,
            L4Re::Dataspace::Offset offset,
            L4::Ipc::In_out< l4_size_t * > size,
            Attributes attrs,
            Direction dir,
            Dma_addr * dma_addr )
```

Map the given part of this data space into the DMA address space.

Precondition

requires capability rights: {R}

Parameters

in	*src*	Source data space (that describes the memory). Caller needs write rights to the data space.
in	*offset*	The offset (bytes) within `src`.
in,out	*size*	The size (bytes) of the region to be mapped for DMA, after successful mapping the size returned is the size mapped for DMA as a single block. This size might be smaller than the original input size, in this case the caller might call map() again with a new offset and the remaining size.
in	*attrs*	The attributes used for this DMA mapping (a combination of Dma_space::Attribute values).
in	*dir*	The direction of the DMA transfer issued with this mapping. The same value must later be passed to unmap().
out	*dma_addr*	The DMA address to use for DMA with the associated device.

Returns

0 in the case of success, a negative error code otherwise.

15.247.4.4 unmap()

```
long L4Re::Dma_space::unmap (
            Dma_addr dma_addr,
            l4_size_t size,
            Attributes attrs,
            Direction dir )
```

Unmap the given part of this data space from the DMA address space.

Precondition

requires capability rights: {R}

Parameters

dma_addr	The DMA address (returned by Dma_space::map()).
size	The size (bytes) of the memory region to unmap.
attrs	The attributes for the unmap (currently none).
dir	The direction of the finished DMA operation.

The documentation for this class was generated from the following file:

- l4/re/dma_space

15.248 L4Re::Env Class Reference

C++ interface of the initial environment that is provided to an L4 task.

Collaboration diagram for L4Re::Env:

Public Types

- typedef l4re_env_cap_entry_t Cap_entry

  C++ type for an entry in the initial objects array.

Public Member Functions

- L4::Cap< Parent > parent () const noexcept

  Object-capability to the parent.
- L4::Cap< Mem_alloc > mem_alloc () const noexcept

  Object-capability to the memory allocator.
- L4::Cap< L4::Factory > user_factory () const noexcept

  Object-capability to the user-level object factory.
- L4::Cap< Rm > rm () const noexcept

  Object-capability to the region map.
- L4::Cap< Log > log () const noexcept

  Object-capability to the logging service.
- L4::Cap< L4::Thread > main_thread () const noexcept

  Object-capability of the first user thread.
- L4::Cap< L4::Task > task () const noexcept

  Object-capability of the user task.
- L4::Cap< L4::Factory > factory () const noexcept

  Object-capability to the factory object available to the task.
- l4_cap_idx_t first_free_cap () const noexcept

  First available capability selector.
- l4_fpage_t utcb_area () const noexcept

  UTCB area of the task.
- l4_addr_t first_free_utcb () const noexcept

  First free UTCB.
- Cap_entry const * initial_caps () const noexcept

  Get a pointer to the first entry in the initial objects array.
- Cap_entry const * get (char const *name, unsigned l) const noexcept

  Get the Cap_entry for the object named name.
- template<typename T >
  L4::Cap< T > get_cap (char const *name, unsigned l) const noexcept

  Get the capability selector for the object named name.
- template<typename T >
  L4::Cap< T > get_cap (char const *name) const noexcept

  Get the capability selector for the object named name.
- void parent (L4::Cap< Parent > const &c) noexcept

  Set parent object-capability.
- void mem_alloc (L4::Cap< Mem_alloc > const &c) noexcept

  Set memory allocator object-capability.
- void rm (L4::Cap< Rm > const &c) noexcept

  Set region map object-capability.
- void log (L4::Cap< Log > const &c) noexcept

  Set log object-capability.
- void main_thread (L4::Cap< L4::Thread > const &c) noexcept

  Set object-capability of first user thread.
- void factory (L4::Cap< L4::Factory > const &c) noexcept

Set factory object-capability.

- void first_free_cap (l4_cap_idx_t c) noexcept

  Set first available capability selector.
- void utcb_area (l4_fpage_t utcbs) noexcept

  Set UTCB area of the task.
- void first_free_utcb (l4_addr_t u) noexcept

  Set first free UTCB.
- L4::Cap< L4::Scheduler > scheduler () const noexcept

  Get the scheduler capability for the task.
- void scheduler (L4::Cap< L4::Scheduler > const &c) noexcept

  Set the scheduler capability.
- void initial_caps (Cap_entry *first) noexcept

  Set the pointer to the first Cap_entry in the initial objects array.

Static Public Member Functions

- static Env const * env () noexcept

  Returns the initial environment for the current task.

15.248.1 Detailed Description

C++ interface of the initial environment that is provided to an L4 task.

The initial environment is provided to each L4 task that is started by an L4Re conform loader, such as the Moe root task. The initial environment provides access to a set of initial capabilities and some additional information about the available resources, such as free UTCBs (see Virtual Registers) and available entries in capability table (provided by the micro kernel).

Each of the initial capabilities is stored at a fixed index in the task's capability table and the L4 runtime environment provides convenience functions to retrieve the capabilities. See the table below for an comprehensive overview.

Name	Object Type	Convenience Function
parent	L4Re::Parent	L4Re::Env::parent()
user_factory	L4::Factory	L4Re::Env::user_factory()
log	L4Re::Log	L4Re::Env::log()
main_thread	L4::Thread	L4Re::Env::main_thread()
rm	L4Re::Rm	L4Re::Env::rm()
factory	L4::Factory	L4Re::Env::factory()
task	L4::Task	L4Re::Env::task()
scheduler	L4::Scheduler	L4Re::Env::scheduler()

Additional information found in the initial environment is:

- First free entry in capability table
- The UTCB area (as flex page)
- First free UTCB (address in the UTCB area)

Include File

```
#include <l4/re/env>
```

For an explanation of the default task capabilites see l4_default_caps_t.

For the C interface refer to Initial Environment.

Definition at line 85 of file env.

15.248.2 Member Function Documentation

15.248.2.1 env()

```
static Env const* L4Re::Env::env ( )  [inline], [static], [noexcept]
```

Returns the initial environment for the current task.

Returns

Pointer to the initial environment class.

A typical use of this function is L4Re::Env::env()-><member>()

Examples

examples/clntsrv/client.cc, examples/libs/l4re/c++/mem_alloc/ma+rm.cc, examples/libs/l4re/c++/shared_ds/ds_clnt.cc, examples/libs/l4re/c++/shared_ds/ds_srv.cc, examples/libs/l4re/streammap/client.cc, and examples/sys/migrate/thread_migrate

Definition at line 103 of file env.

Referenced by L4virtio::Svr::Dev_config::Dev_config(), L4virtio::Driver::Device::driver_connect(), and L4virtio::Driver::Block_device::se

Here is the caller graph for this function:

15.248.2.2 factory() [1/2]

```
L4::Cap<L4::Factory> L4Re::Env::factory ( ) const  [inline], [noexcept]
```

Object-capability to the factory object available to the task.

Returns

Factory object-capability

Definition at line 151 of file env.

References l4re_env_t::factory.

15.248.2.3 factory() [2/2]

```
void L4Re::Env::factory (
            L4::Cap< L4::Factory > const & c )  [inline], [noexcept]
```

Set factory object-capability.

Parameters

c	Factory object-capability

Definition at line 256 of file env.

References l4re_env_t::factory.

15.248.2.4 first_free_cap() [1/2]

```
l4_cap_idx_t L4Re::Env::first_free_cap ( ) const  [inline], [noexcept]
```

First available capability selector.

Returns

First capability selector.

First capability selector available for use for in the application.

Definition at line 159 of file env.

References l4re_env_t::first_free_cap.

15.248.2.5 first_free_cap() [2/2]

```
void L4Re::Env::first_free_cap (
            l4_cap_idx_t c )  [inline], [noexcept]
```

Set first available capability selector.

Parameters

c	First capability selector available to the application.

Definition at line 262 of file env.

References l4re_env_t::first_free_cap.

15.248.2.6 first_free_utcb() [1/2]

```
l4_addr_t L4Re::Env::first_free_utcb ( ) const  [inline], [noexcept]
```

First free UTCB.

Returns

object-capability

First free UTCB within the UTCB area available for the application to use.

Definition at line 174 of file env.

References l4re_env_t::first_free_utcb.

15.248.2.7 first_free_utcb() [2/2]

```
void L4Re::Env::first_free_utcb (
            l4_addr_t u )  [inline], [noexcept]
```

Set first free UTCB.

Parameters

u	First UTCB available for the application to use.

Definition at line 274 of file env.

References l4re_env_t::first_free_utcb.

15.248.2.8 get()

```
Cap_entry const* L4Re::Env::get (
            char const * name,
            unsigned l ) const  [inline], [noexcept]
```

Get the Cap_entry for the object named *name*.

Parameters

name	is the name of the object.
l	is the length of the name, thus *name* might not be zero terminated.

Returns

A pointer to the Cap_entry for the object named *name*, or NULL if no such object was found.

Definition at line 192 of file env.

References l4re_env_get_cap_l().

Here is the call graph for this function:

15.248.2.9 get_cap() [1/2]

```
template<typename T >
L4::Cap<T> L4Re::Env::get_cap (
            char const * name ) const  [inline], [noexcept]
```

Get the capability selector for the object named *name*.

Parameters

name	is the name of the object (zero terminated).

Returns

A capability selector for the object named *name*, or an invalid capability selector if no such object was found.

Definition at line 219 of file env.

15.248.2.10 get_cap() [2/2]

```
template<typename T >
L4::Cap<T> L4Re::Env::get_cap (
            char const * name,
            unsigned l ) const  [inline], [noexcept]
```

Get the capability selector for the object named *name*.

Parameters

name	is the name of the object.
l	is the length of the name, thus *name* might not be zero terminated.

Returns

A capability selector for the object named *name*, or an invalid capability selector if no such object was found.

Examples

examples/clntsrv/client.cc, examples/libs/l4re/c++/shared_ds/ds_clnt.cc, and examples/libs/l4re/streammap/client.cc.

Definition at line 204 of file env.

References L4_ENOENT.

15.248.2.11 initial_caps() [1/2]

```
Cap_entry const* L4Re::Env::initial_caps ( ) const  [inline], [noexcept]
```

Get a pointer to the first entry in the initial objects array.

Returns

A pointer to the first entry in the initial objects array.

Definition at line 181 of file env.

15.248.2.12 initial_caps() [2/2]

```
void L4Re::Env::initial_caps (
            Cap_entry * first )  [inline], [noexcept]
```

Set the pointer to the first Cap_entry in the initial objects array.

Parameters

first	is the first element in the array.

Definition at line 296 of file env.

15.248.2.13 log() [1/2]

```
L4::Cap<Log> L4Re::Env::log ( ) const  [inline], [noexcept]
```

Object-capability to the logging service.

Returns

Log object-capability

Definition at line 133 of file env.

References l4re_env_t::log.

15.248.2.14 log() [2/2]

```
void L4Re::Env::log (
            L4::Cap< Log > const & c )  [inline], [noexcept]
```

Set log object-capability.

Parameters

c	Log object-capability

Definition at line 244 of file env.

References l4re_env_t::log.

15.248.2.15 main_thread() [1/2]

```
L4::Cap<L4::Thread> L4Re::Env::main_thread ( ) const  [inline], [noexcept]
```

Object-capability of the first user thread.

Returns

Object-capability of the first user thread.

Definition at line 139 of file env.

References l4re_env_t::main_thread.

15.248.2.16 main_thread() [2/2]

```
void L4Re::Env::main_thread (
            L4::Cap< L4::Thread > const & c )  [inline], [noexcept]
```

Set object-capability of first user thread.

Parameters

c	First thread's object-capability

Definition at line 250 of file env.

References l4re_env_t::main_thread.

15.248.2.17 mem_alloc() [1/2]

```
L4::Cap<Mem_alloc> L4Re::Env::mem_alloc ( ) const  [inline], [noexcept]
```

Object-capability to the memory allocator.

Returns

Memory allocator object-capability

Examples

examples/libs/l4re/c++/shared_ds/ds_srv.cc.

Definition at line 116 of file env.

References l4re_env_t::mem_alloc.

15.248.2.18 mem_alloc() [2/2]

```
void L4Re::Env::mem_alloc (
            L4::Cap< Mem_alloc > const & c )  [inline], [noexcept]
```

Set memory allocator object-capability.

Parameters

c	Memory allocator object-capability

Definition at line 232 of file env.

References l4re_env_t::mem_alloc.

15.248.2.19 parent() [1/2]

```
L4::Cap<Parent> L4Re::Env::parent ( ) const  [inline], [noexcept]
```

Object-capability to the parent.

Returns

Parent object-capability

Definition at line 110 of file env.

References l4re_env_t::parent.

15.248.2.20 parent() [2/2]

```
void L4Re::Env::parent (
            L4::Cap< Parent > const & c )  [inline], [noexcept]
```

Set parent object-capability.

Parameters

c	Parent object-capability

Definition at line 226 of file env.

References l4re_env_t::parent.

15.248.2.21 rm() [1/2]

```
L4::Cap<Rm> L4Re::Env::rm ( ) const  [inline], [noexcept]
```

Object-capability to the region map.

Returns

Region map object-capability

Examples

examples/libs/l4re/c++/shared_ds/ds_clnt.cc, and examples/libs/l4re/c++/shared_ds/ds_srv.cc.

Definition at line 127 of file env.

References l4re_env_t::rm.

15.248.2.22 rm() [2/2]

```
void L4Re::Env::rm (
            L4::Cap< Rm > const & c )  [inline], [noexcept]
```

Set region map object-capability.

Parameters

c	Region map object-capability

Definition at line 238 of file env.

References l4re_env_t::rm.

15.248.2.23 scheduler() [1/2]

```
L4::Cap<L4::Scheduler> L4Re::Env::scheduler ( ) const  [inline], [noexcept]
```

Get the scheduler capability for the task.

Returns

The capability selector for the default scheduler used for this task.

Examples

examples/sys/migrate/thread_migrate.cc.

Definition at line 282 of file env.

References l4re_env_t::scheduler.

15.248.2.24 scheduler() [2/2]

```
void L4Re::Env::scheduler (
            L4::Cap< L4::Scheduler > const & c )  [inline], [noexcept]
```

Set the scheduler capability.

Parameters

c	is the capability to be set as scheduler.

Definition at line 289 of file env.

References l4re_env_t::scheduler.

15.248.2.25 task()

```
L4::Cap<L4::Task> L4Re::Env::task ( ) const  [inline], [noexcept]
```

Object-capability of the user task.

Returns

Object-capability of the user task.

Definition at line 145 of file env.

15.248.2.26 utcb_area() [1/2]

```
l4_fpage_t L4Re::Env::utcb_area ( ) const  [inline], [noexcept]
```

UTCB area of the task.

Returns

UTCB area

Definition at line 165 of file env.

References l4re_env_t::utcb_area.

15.248.2.27 utcb_area() [2/2]

```
void L4Re::Env::utcb_area (
            l4_fpage_t utcbs )  [inline], [noexcept]
```

Set UTCB area of the task.

Parameters

utcbs	UTCB area

Definition at line 268 of file env.

References l4re_env_t::utcb_area.

The documentation for this class was generated from the following file:

- l4/re/env

15.249 L4Re::Event Class Reference

Event class.

Inheritance diagram for L4Re::Event:

Collaboration diagram for L4Re::Event:

Public Member Functions

- long get_buffer (L4::Ipc::Out< L4::Cap< Dataspace > > ds)

  Get event signal buffer.

Additional Inherited Members

15.249.1 Detailed Description

Event class.

See also

L4Re Event API

Definition at line 144 of file event.

15.249.2 Member Function Documentation

15.249.2.1 get_buffer()

```
long L4Re::Event::get_buffer (
            L4::Ipc::Out< L4::Cap< Dataspace > > ds )
```

Get event signal buffer.

Return values

ds	Event buffer.

Returns

0 on success, negative error code otherwise.

The documentation for this class was generated from the following file:

- l4/re/event

15.250 L4Re::Event_buffer_t< PAYLOAD > Class Template Reference

Event buffer class.

Inheritance diagram for L4Re::Event_buffer_t< PAYLOAD >:

Collaboration diagram for L4Re::Event_buffer_t< PAYLOAD >:

Data Structures

- struct Event

  Event structure used in buffer.

Public Member Functions

- Event_buffer_t (void *buffer, l4_addr_t size)

  Initialize event buffer.
- Event * next () noexcept

  Next event in buffer.
- bool put (Event const &ev) noexcept

  Put event into buffer at current position.

15.250.1 Detailed Description

template<typename PAYLOAD = Default_event_payload>
class L4Re::Event_buffer_t< PAYLOAD >

Event buffer class.

Definition at line 198 of file event.

15.250.2 Constructor & Destructor Documentation

15.250.2.1 Event_buffer_t()

```
template<typename PAYLOAD = Default_event_payload>
L4Re::Event_buffer_t< PAYLOAD >::Event_buffer_t (
            void * buffer,
            l4_addr_t size )  [inline]
```

Initialize event buffer.

Parameters

buffer	Pointer to buffer.
size	Size of buffer in bytes.

Definition at line 245 of file event.

15.250.3 Member Function Documentation

15.250.3.1 next()

```
template<typename PAYLOAD = Default_event_payload>
Event* L4Re::Event_buffer_t< PAYLOAD >::next ( )  [inline], [noexcept]
```

Next event in buffer.

Returns

0 if no event available, event otherwise.

Definition at line 255 of file event.

References L4Re::Event_buffer_t< PAYLOAD >::Event::time.

15.250.3.2 put()

```
template<typename PAYLOAD = Default_event_payload>
bool L4Re::Event_buffer_t< PAYLOAD >::put (
            Event const & ev )  [inline], [noexcept]
```

Put event into buffer at current position.

Parameters

ev	Event to put into the buffer.

Returns

false if buffer is full and entry could not be added.

Definition at line 272 of file event.

References l4_wmb(), and L4Re::Event_buffer_t< PAYLOAD >::Event::time.

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/re/event

15.251 L4Re::Event_buffer_t< PAYLOAD >::Event Struct Reference

Event structure used in buffer.

Collaboration diagram for L4Re::Event_buffer_t< PAYLOAD >::Event:

Public Member Functions

- void free () noexcept

  Free the entry.

Data Fields

- long long time

  Event time stamp.

15.251.1 Detailed Description

template<typename PAYLOAD = Default_event_payload>
struct L4Re::Event_buffer_t< PAYLOAD >::Event

Event structure used in buffer.

Definition at line 205 of file event.

The documentation for this struct was generated from the following file:

- l4/re/event

15.252 L4Re::Inhibitor Class Reference

Set of inhibitor locks, which inhibit specific actions when held.

Inheritance diagram for L4Re::Inhibitor:

Collaboration diagram for L4Re::Inhibitor:

Public Types

- enum { Name_max = 20 }

Public Member Functions

- long acquire (l4_umword_t id, L4::Ipc::String<> reason)

  Acquire a specific inhibitor lock.
- long release (l4_umword_t id)

  Release a specific inhibitor lock.
- long next_lock_info (char *name, unsigned len, l4_mword_t current_id=-1, l4_utcb_t *utcb=l4_utcb())

  Get information for the next available inhibitor lock.

Additional Inherited Members

15.252.1 Detailed Description

Set of inhibitor locks, which inhibit specific actions when held.

This interface provides access to a set of inhibitor locks, each determined by an ID that is specific to the Inhibitor object. Each individual lock shall prevent, a specific (implementation defined) action to be executed, as long as the lock is held.

For example there can be an inhibitor lock to prevent a transition to suspend-to-RAM state and a different one to prevent shutdown.

A client shall take an inhibitor lock if it needs to execute code before the action is taken. For example a lock-screen application shall grab an inhibitor lock for the suspend action to be able to lock the screen before the system goes to sleep.

Inhibitor locks are usually closely related to specific events. Usually a server automatically subscribes a client holding a lock to the corresponding event. The server shall send the event to inform the client that an action is pending. Upon reception of the event, the client is supposed to release the corresponding inhibitor lock.

Definition at line 40 of file inhibitor.

15.252.2 Member Enumeration Documentation

15.252.2.1 anonymous enum

```
anonymous enum
```

Enumerator

Name_max	The maximum length of a lock's name.

Definition at line 44 of file inhibitor.

15.252.3 Member Function Documentation

15.252.3.1 acquire()

```
long L4Re::Inhibitor::acquire (
            l4_umword_t id,
            L4::Ipc::String<> reason )
```

Acquire a specific inhibitor lock.

Parameters

id	ID of the inhibitor lock that the client intends to acquire
reason	The reason why you need the lock. Used for informing the user or debugging.

Return values

0	Success
-L4_ENODEV	The specified `id` does not exist.

15.252.3.2 next_lock_info()

```
long L4Re::Inhibitor::next_lock_info (
            char * name,
            unsigned len,
            l4_mword_t current_id = -1,
            l4_utcb_t * utcb = l4_utcb() )  [inline]
```

Get information for the next available inhibitor lock.

Parameters

name	A pointer to a buffer for the name of the lock.
len	The length of the available buffer (usually Name_max is used)
current_id	The ID of the last available lock, use -1 to get the first lock.
utcb	The UTCB to use for the message.

Return values

>0	The ID of the next available lock if there is one (in this case `name` shall contain the name of the inhibitor lock).
-L4_ENODEV	There are no more locks.

Definition at line 86 of file inhibitor.

References L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >::c().

Here is the call graph for this function:

15.252.3.3 release()

```
long L4Re::Inhibitor::release (
            l4_umword_t id )
```

Release a specific inhibitor lock.

Parameters

id	The ID of the inhibitor lock to release.

Return values

0	Success
-L4_ENODEV	Lock with the given `id` does not exist.

The documentation for this class was generated from the following file:

- l4/re/inhibitor

15.253 L4Re::Log Class Reference

Log interface class.

Inheritance diagram for L4Re::Log:

Collaboration diagram for L4Re::Log:

Public Member Functions

- void printn (char const ∗string, int len) const noexcept

  Print string with length len, NULL characters don't matter.
- void print (char const ∗string) const noexcept

  Print NULL-terminated string.

Additional Inherited Members

15.253.1 Detailed Description

Log interface class.

Definition at line 44 of file log.

15.253.2 Member Function Documentation

15.253.2.1 print()

```
void L4Re::Log::print (
            char const * string ) const  [noexcept]
```

Print NULL-terminated string.

Parameters

string	string to print

15.253.2.2 printn()

```
void L4Re::Log::printn (
            char const * string,
            int len ) const  [noexcept]
```

Print string with length len, NULL characters don't matter.

Parameters

string	string to print
len	length of string

The documentation for this class was generated from the following file:

- l4/re/log

15.254 L4Re::Mem_alloc Class Reference

Memory allocation interface.

Inheritance diagram for L4Re::Mem_alloc:

Collaboration diagram for L4Re::Mem_alloc:

Public Types

- enum Mem_alloc_flags { Continuous = 0x01 , Pinned = 0x02 , Super_pages = 0x04 }

  Flags for the allocator.

Public Member Functions

- long alloc (long size, L4::Cap< Dataspace > mem, unsigned long flags=0, unsigned long align=0) const noexcept

  Allocate anonymous memory.

Additional Inherited Members

15.254.1 Detailed Description

Memory allocation interface.

The memory-allocator API is the basic API to allocate memory from the L4Re subsystem. The memory is allocated in terms of dataspaces (see L4Re::Dataspace). The provided dataspaces have at least the property that data written to such a dataspace is available as long as the dataspace is not freed or the data is not overwritten. In particular, the memory backing a dataspace from an allocator need not be allocated instantly, but may be allocated lazily on demand.

A memory allocator can provide dataspaces with additional properties, such as physically contiguous memory, pre-allocated memory, or pinned memory. To request memory with an additional property the L4Re::Mem_alloc::alloc() method provides a flags parameter. If the concrete implementation of a memory allocator does not support or allow allocation of memory with a certain property, the allocation may be refused.

Definition at line 61 of file mem_alloc.

15.254.2 Member Enumeration Documentation

15.254.2.1 Mem_alloc_flags

`enum L4Re::Mem_alloc::Mem_alloc_flags`

Flags for the allocator.

They describe requested properties of the allocated memory. Support of these properties by the dataspace provider is optional.

Enumerator

Continuous	Allocate physically contiguous memory.
Pinned	Deprecated, use L4Re::Dma_space instead.
Super_pages	Allocate super pages.

Definition at line 71 of file mem_alloc.

15.254.3 Member Function Documentation

15.254.3.1 alloc()

```
long L4Re::Mem_alloc::alloc (
            long size,
            L4::Cap< Dataspace > mem,
            unsigned long flags = 0,
            unsigned long align = 0 ) const  [noexcept]
```

Allocate anonymous memory.

Parameters

	size	Size in bytes to be requested. Allocation granularity is (super)pages, however, the allocator will store the byte-granular given size as the size of the dataspace and consecutively will use this byte-granular size for servicing the dataspace. Allocators may optionally also implement a maximum allocation strategy: if `size` is a negative value and `flags` set the Mem_alloc_flags::Continuous bit, the allocator tries to allocate as much memory as possible leaving an amount of at least `-size` bytes within the associated quota.
`out`	*mem*	Capability slot where the capability to the dataspace is received.
	flags	Special dataspace properties, see Mem_alloc_flags
	align	Log2 alignment of dataspace if supported by allocator, will be at least L4_PAGESHIFT, with Super_pages flag set at least L4_SUPERPAGESHIFT

Return values

0	Success
-L4_ERANGE	Given size not supported.
-L4_ENOMEM	Not enough memory available.
<0	IPC error

Definition at line 35 of file mem_alloc_impl.h.

References L4::Factory::create(), and l4_error().

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- l4/re/mem_alloc
- l4/re/impl/mem_alloc_impl.h

15.255 L4Re::Mmio_space Struct Reference

Interface for memory-like address space accessible via IPC.

Inheritance diagram for L4Re::Mmio_space:

Collaboration diagram for L4Re::Mmio_space:

Public Types

- enum Access_width { Wd_8bit = 0 , Wd_16bit = 1 , Wd_32bit = 2 , Wd_64bit = 3 }

  Actual size of the value to read or write.
- typedef l4_uint64_t Addr

  Device address.

Public Member Functions

- long mmio_read (Addr addr, char width, l4_uint64_t *value)

  Read a value from the given address.
- long mmio_write (Addr addr, char width, l4_uint64_t value)

  Write a value to the given address.

Additional Inherited Members

15.255.1 Detailed Description

Interface for memory-like address space accessible via IPC.

This interface defines methods for indirect access to MMIO regions.

Memory mapped IO (MMIO) is used by device drivers to control hardware devices. Access to MMIO regions is assigned to user-level device drivers via mappings of memory pages.

However, there are hardware platforms where MMIO regions for different devices share the same memory page. With respect to security and safety, it is often not allowed to map a memory page to multiple device drivers because the driver of one device could then influence operation of another device, which violates security boundaries.

A solution to that problem is to implement a third (trusted) component that gets exclusive access to the shared memory page, and that drivers can access via IPC with the Mmio_space protocol. This proxy-component can then enforce an access policy.

Include File

```
#include <l4/re/mmio_space>
```

Definition at line 46 of file mmio_space.

15.255.2 Member Enumeration Documentation

15.255.2.1 Access_width

```
enum L4Re::Mmio_space::Access_width
```

Actual size of the value to read or write.

Enumerator

Wd_8bit	Value is a byte.
Wd_16bit	Value is a 2-byte word.
Wd_32bit	Value is a 4-byte word.
Wd_64bit	Value is a 8-byte word.

Definition at line 50 of file mmio_space.

15.255.3 Member Function Documentation

15.255.3.1 mmio_read()

```
long L4Re::Mmio_space::mmio_read (
            Addr addr,
            char width,
            l4_uint64_t * value )
```

Read a value from the given address.

Parameters

	addr	Device virtual address to read from. The address must be aligned relative to the access width.
	width	Access width of value to be read, see Access_width.
out	*value*	Return value. If width is smaller than 64 bit,the upper bits are guaranteed to be 0.

Return values

L4_EOK	Success.
-L4_EPERM	Insufficient read rights.
-L4_EINVAL	Address does not exist or cannot be accessed with the given width.

15.255.3.2 mmio_write()

```
long L4Re::Mmio_space::mmio_write (
            Addr addr,
            char width,
            l4_uint64_t value )
```

Write a value to the given address.

Parameters

addr	Device virtual address to write to. The address must be aligned relative to the access width.
width	Access width of value to write, see Access_width.
value	Value to write. If width is smaller than 64 bit, the upper bits are ignored.

Return values

L4_EOK	Success.
-L4_EPERM	Insufficient write rights.
-L4_EINVAL	Address does not exist or cannot be accessed with the given width.

The documentation for this struct was generated from the following file:

- l4/re/mmio_space

15.256 L4Re::Namespace Class Reference

Name-space interface.

Inheritance diagram for L4Re::Namespace:

Collaboration diagram for L4Re::Namespace:

Public Types

- enum Register_flags {
  Ro = L4_CAP_FPAGE_RO , Rw = L4_CAP_FPAGE_RW , Rs = L4_CAP_FPAGE_RS , Rws = L4_CAP_↩ FPAGE_RWS ,
  Strong = L4_CAP_FPAGE_S , Trusted = 0x008 , **Cap_flags** = Ro | Rw | Strong | Trusted , Link = 0x100 , Overwrite = 0x200 }

  Flags for registering name spaces.
- enum Query_result_flags { Partly_resolved = 0x020 }

  Flags returned by query IPC, only used internally.

Public Member Functions

- long query (char const ∗name, L4::Cap< void > const &cap, int timeout=To_default, l4_umword_t ∗local_↩ id=0, bool iterate=true) const noexcept

  Query the name space for a named object.

- long query (char const *name, unsigned len, L4::Cap< void > const &cap, int timeout=To_default, l4_umword_t *local_id=0, bool iterate=true) const noexcept

  Query the name space for a named object.
- long register_obj (char const *name, L4::Ipc::Cap< void > obj, unsigned flags=Rw) const noexcept

  Register an object with a name.
- long unlink (char const *name)

  Remove an entry from the name space.

Additional Inherited Members

15.256.1 Detailed Description

Name-space interface.

All name space objects must provide this interface. However, it is not mandatory that a name space object allows to register new capabilities.

The name lookup is done iteratively, this means the hierarchical names are resolved component wise by the client itself.

Definition at line 60 of file namespace.

15.256.2 Member Enumeration Documentation

15.256.2.1 Query_result_flags

`enum L4Re::Namespace::Query_result_flags`

Flags returned by query IPC, only used internally.

Enumerator

Partly_resolved	Name was only partly resolved.

Definition at line 88 of file namespace.

15.256.2.2 Register_flags

`enum L4Re::Namespace::Register_flags`

Flags for registering name spaces.

Enumerator

Ro	Read-only.
Rw	Read-write.
Rs	Read-only + strong.
Rws	Read-write + strong.
Strong	Strong.
Trusted	Obsolete, do not use.
Link	Obsolete, do not use.
Overwrite	If entry already exists, overwrite it.

Definition at line 68 of file namespace.

15.256.3 Member Function Documentation

15.256.3.1 query() [1/2]

```
long L4Re::Namespace::query (
            char const * name,
            L4::Cap< void > const & cap,
            int timeout = To_default,
            l4_umword_t * local_id = 0,
            bool iterate = true ) const  [noexcept]
```

Query the name space for a named object.

Parameters

in	*name*	String to query (without any leading slashes).
out	*cap*	Capability slot where the received capability will be put.
in	*timeout*	Timeout of query in milliseconds. The client will only wait if a name has already been registered with the server but no object has yet been attached.
out	*local↩ _id*	If given, L4_RCV_ITEM_LOCAL_ID will be set for the IPC from the name space, so that if the capability that was received is a local item, the capability ID will be returned with this parameter.
in	*iterate*	If true, the client will try to resolve names by iteratively calling the name spaces until the name is fully resolved.

Return values

0	Name could be fully resolved.
>0	Name could only be partly resolved. The number of remaining characters is returned.
-L4_ENOENT	Entry could not be found.
-L4_EAGAIN	Entry exists but no object is yet attached. Try again later.
<0	IPC errors, see l4_error_code_t.

Definition at line 125 of file namespace_impl.h.

15.256.3.2 query() [2/2]

```
long L4Re::Namespace::query (
            char const * name,
            unsigned len,
            L4::Cap< void > const & cap,
            int timeout = To_default,
            l4_umword_t * local_id = 0,
            bool iterate = true ) const  [noexcept]
```

Query the name space for a named object.

The query string does not necessarily need to be null-terminated.

Parameters

in	*len*	Length of the string to query without any terminating null characters.
in	*name*	String to query (without any leading slashes).
out	*cap*	Capability slot where the received capability will be put.
in	*timeout*	Timeout of query in milliseconds. The client will only wait if a name has already been registered with the server but no object has yet been attached.
out	*local↩ _id*	If given, L4_RCV_ITEM_LOCAL_ID will be set for the IPC from the name space, so that if the capability that was received is a local item, the capability ID will be returned with this parameter.
in	*iterate*	If true, the client will try to resolve names by iteratively calling the name spaces until the name is fully resolved.

Return values

0	Name could be fully resolved.
>0	Name could only be partly resolved. The number of remaining characters is returned.
-L4_ENOENT	Entry could not be found.
-L4_EAGAIN	Entry exists but no object is yet attached. Try again later.
<0	IPC errors, see l4_error_code_t.

Definition at line 77 of file namespace_impl.h.

References L4_EAGAIN, L4_EINVAL, l4_sleep(), and L4_UNLIKELY.

Here is the call graph for this function:

15.256.3.3 register_obj()

```
long L4Re::Namespace::register_obj (
            char const * name,
            L4::Ipc::Cap< void > obj,
            unsigned flags = Rw ) const  [inline], [noexcept]
```

Register an object with a name.

Parameters

name	Name under which the object should be registered.
obj	Capability to object to register. An invalid capability may be given to only reserve the name for later use.
flags	Flags to assign to the entry, see L4Re::Namespace::Register_flags. Note that the rights that are assigned to a capability are not only determined by the rights given in these flags but also by the rights with which the `obj` capability was mapped to the name space.

Return values

0	Object was successfully registered with *name*.
-L4_EEXIST	Name already registered.
-L4_EPERM	Caller doesn't have necessary permissions.
-L4_ENOMEM	Server has insufficient resources.
-L4_EINVAL	Invalid parameter.
<0	IPC errors, see l4_error_code_t.

Precondition

requires capability rights: {RW}

Definition at line 176 of file namespace.

15.256.3.4 unlink()

```
long L4Re::Namespace::unlink (
            char const * name )  [inline]
```

Remove an entry from the name space.

Parameters

name	Name of the entry to remove.

Return values

0	Entry successfully removed.
-L4_ENOENT	Given name does not exist.
-L4_EPERM	Caller does not have write permission.
-L4_EACCESS	Name cannot be removed.
<0	IPC errors, see l4_error_code_t.

Precondition

requires capability rights: {RW}

Definition at line 202 of file namespace.

The documentation for this class was generated from the following files:

- l4/re/namespace
- l4/re/impl/namespace_impl.h

15.257 L4Re::Ned::Cmd_control Class Reference

Direct control interface for Ned.

Inherits L4::Kobject_0t< Derived, PROTO, S_DEMAND >.

Collaboration diagram for L4Re::Ned::Cmd_control:

Public Member Functions

- long execute (L4::Ipc::String<> cmd) noexcept

  Execute the given Lua code.
- long execute (L4::Ipc::String<> cmd, L4::Ipc::String< char > *result) noexcept

  Execute the given Lua code.

15.257.1 Detailed Description

Direct control interface for Ned.

Definition at line 20 of file cmd_control.

15.257.2 Member Function Documentation

15.257.2.1 execute() [1/2]

```
long L4Re::Ned::Cmd_control::execute (
            L4::Ipc::String<> cmd )  [inline], [noexcept]
```

Execute the given Lua code.

Parameters

in	*cmd*	String with Lua code to execute.

Return values

L4_EOK	Code was successfully executed.
-L4_EINVAL	Code could not be parsed.
-L4_EIO	Error during code execution.

The code is executed using the global Lua state of ned which is retained between successive calls to execute. Thus you may define data in one call to execute and use it in a subsequent call.

This function does not return any results from the execution of the Lua code itself.

Definition at line 43 of file cmd_control.

15.257.2.2 execute() [2/2]

```
long L4Re::Ned::Cmd_control::execute (
            L4::Ipc::String<> cmd,
            L4::Ipc::String< char > * result )  [inline], [noexcept]
```

Execute the given Lua code.

Parameters

in	*cmd*	String with Lua code to execute.
out	*result*	The first return value of the Lua code block as string.

Return values

L4_EOK	Code was successfully executed.
-L4_EINVAL	Code could not be parsed.
-L4_EIO	Error during code execution.

The code is executed using the global Lua state of ned which is retained between successive calls to execute. Thus you may define data in one call to execute and use it in a subsequent call.

Definition at line 65 of file cmd_control.

The documentation for this class was generated from the following file:

- pkg/l4re-core/ned/lib/include/cmd_control

15.258 L4Re::Parent Class Reference

Parent interface.

Inheritance diagram for L4Re::Parent:

Collaboration diagram for L4Re::Parent:

Public Member Functions

- long signal (unsigned long sig, unsigned long val)

  Send a signal to the parent.

Additional Inherited Members

15.258.1 Detailed Description

Parent interface.

See also

Parent API for more details about the purpose.

Definition at line 53 of file parent.

15.258.2 Member Function Documentation

15.258.2.1 signal()

```
long L4Re::Parent::signal (
            unsigned long sig,
            unsigned long val )
```

Send a signal to the parent.

Parameters

sig	Signal to send
val	Value of the signal

Returns

0 on success, <0 on error

- -L4_ENOREPLY
- IPC errors

The documentation for this class was generated from the following file:

- l4/re/parent

15.259 L4Re::Random Struct Reference

Low-bandwidth interface for random number generators.

Inheritance diagram for L4Re::Random:

Collaboration diagram for L4Re::Random:

Public Member Functions

- long get_random (l4_size_t size, L4::Ipc::Array< char, unsigned long > *buffer)

  Get a random number.

Additional Inherited Members

15.259.1 Detailed Description

Low-bandwidth interface for random number generators.

The interface offers an ICU interface where a client can register an interrupt to get notified when entropy is available. Support for notifications is optional. If a service does not implement notification, it must return 0 for the number of interrupts in the info() call. The notification interrupt must have index 0.

Include File

```
#include <l4/re/random>
```

Definition at line 33 of file random.

15.259.2 Member Function Documentation

15.259.2.1 get_random()

```
long L4Re::Random::get_random (
            l4_size_t size,
            L4::Ipc::Array< char, unsigned long > * buffer )
```

Get a random number.

Parameters

	size	Number of bytes of entropy requested.
out	*buffer*	Buffer containing the random number. Each byte in the buffer contains 8 bits of randomness.

Return values

>=0	Actual size of the returned random number in bytes. This may be less than the requested size. The return value may also be 0 if temporarily no entropy is available.
-L4_EIO	Source of randomness permanently unavailable.
<0	IPC error.

This function should never block. It should immediately return as much entropy as is available. If the call returns less than the requested bytes and a notification interrupt was installed, then the service triggers an interrupt as soon as the remaining entropy is available. That means that when an interrupt is triggered, the service must guarantee that the next call to get_random() returns at least the number of missing bytes for the call that initially triggered the notification.

If get_random() is called while a notification is pending, then the behaviour is implementation-defined.

The documentation for this struct was generated from the following file:

- l4/re/random

15.260 L4Re::Rm Class Reference

Region map.

```
#include <l4/re/rm>
```

Inheritance diagram for L4Re::Rm:

Collaboration diagram for L4Re::Rm:

Data Structures

- struct F

  Rm flags definitions.

- struct Range

  A range of virtual addresses.

Public Types

- enum Detach_result { Detached_ds = 0 , Kept_ds = 1 , Split_ds = 2 , **Detach_result_mask** = 3 , Detach_again = 4 }

  Result values for detach operation.
- enum Region_flag_shifts { Caching_shift = Dataspace::F::Caching_shift }
- enum Detach_flags { Detach_exact = 1 , Detach_overlap = 2 , Detach_keep = 4 }

  Flags for detach operation.
- using Region = Range

  A region is a range of virtual addresses which is backed by a dataspace.
- using Area = Range

  An area is a range of virtual addresses which is reserved, see L4Re::Rm::reserve_area().

Public Member Functions

- long reserve_area (l4_addr_t *start, unsigned long size, Flags flags=Flags(0), unsigned char align=L4_PAGESHIFT) const noexcept

  Reserve the given area in the region map.
- template<typename T >
  long reserve_area (T **start, unsigned long size, Flags flags=Flags(0), unsigned char align=L4_PAGESHIFT) const noexcept

  Reserve the given area in the region map.
- long free_area (l4_addr_t addr)

  Free an area from the region map.
- long attach (l4_addr_t *start, unsigned long size, Flags flags, L4::Ipc::Cap< Dataspace > mem, Offset offs=0, unsigned char align=L4_PAGESHIFT) const noexcept

  Attach a data space to a region.
- template<typename T >
  long attach (T **start, unsigned long size, Flags flags, L4::Ipc::Cap< Dataspace > mem, Offset offs=0, unsigned char align=L4_PAGESHIFT) const noexcept

  Attach a data space to a region.
- int detach (l4_addr_t addr, L4::Cap< Dataspace > *mem, L4::Cap< L4::Task > const &task=This_task) const noexcept

  Detach and unmap a region from the address space.
- int detach (void *addr, L4::Cap< Dataspace > *mem, L4::Cap< L4::Task > const &task=This_task) const noexcept

  Detach and unmap a region from the address space.
- int detach (l4_addr_t start, unsigned long size, L4::Cap< Dataspace > *mem, L4::Cap< L4::Task > const &task) const noexcept

  Detach and unmap all parts of the regions within the specified interval.
- int find (l4_addr_t *addr, unsigned long *size, Offset *offset, L4Re::Rm::Flags *flags, L4::Cap< Dataspace > *m) noexcept

  Find a region given an address and size.
- long get_regions (l4_addr_t start, L4::Ipc::Ret_array< Range > regions)

  Return the list of regions whose starting addresses are higher or equal to `start` in the address space managed by this region map.
- long get_areas (l4_addr_t start, L4::Ipc::Ret_array< Range > areas)

  Return the list of areas whose starting addresses are higher or equal to `start` in the address space managed by this region map.

Additional Inherited Members

15.260.1 Detailed Description

Region map.

See also

Region map API .

Definition at line 80 of file rm.

15.260.2 Member Typedef Documentation

15.260.2.1 Area

```
using L4Re::Rm::Area = Range
```

An area is a range of virtual addresses which is reserved, see L4Re::Rm::reserve_area().

See also

Region map API

Definition at line 608 of file rm.

15.260.2.2 Region

```
using L4Re::Rm::Region = Range
```

A region is a range of virtual addresses which is backed by a dataspace.

See also

Region map API

Definition at line 600 of file rm.

15.260.3 Member Enumeration Documentation

15.260.3.1 Detach_flags

```
enum L4Re::Rm::Detach_flags
```

Flags for detach operation.

Enumerator

Detach_exact	Do an unmap of the exact region given.
Detach_overlap	Do an unmap of all overlapping regions.
Detach_keep	Do not free the detached data space, ignore the F::Detach_free.

Definition at line 213 of file rm.

15.260.3.2 Detach_result

`enum L4Re::Rm::Detach_result`

Result values for detach operation.

Enumerator

Detached_ds	Detached data sapce.
Kept_ds	Kept data space.
Split_ds	Splitted data space, and done.
Detach_again	Detached data space, more to do.

Definition at line 88 of file rm.

15.260.3.3 Region_flag_shifts

`enum L4Re::Rm::Region_flag_shifts`

Enumerator

Caching_shift	Start of Rm cache bits.

Definition at line 99 of file rm.

15.260.4 Member Function Documentation

15.260.4.1 attach() [1/2]

```
long L4Re::Rm::attach (
            l4_addr_t * start,
```

```
unsigned long size,
Rm::Flags flags,
L4::Ipc::Cap< Dataspace > mem,
Rm::Offset offs = 0,
unsigned char align = L4_PAGESHIFT ) const  [noexcept]
```

Attach a data space to a region.

Parameters

in,out	*start*	Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If L4Re::Rm::F::Search_addr is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If L4Re::Rm::F::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) to attach the data space to.
	size	Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size.
	flags	The flags control how and with which rights the dataspace is attached to the region. See L4Re::Rm::F::Attach_flags and L4Re::Rm::F::Region_flags. The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the `F::Eager_map` flag is set this function may also return L4Re::Dataspace::map error codes if the mapping fails.
	mem	Data space.
	offs	Offset into the data space to use.
	align	Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used.

Return values

0	Success
-L4_ENOENT	No area could be found (see L4Re::Rm::F::In_area)
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAVAIL	The given address is not available.
<0	IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see L4Re::Rm::find).

Definition at line 43 of file rm_impl.h.

15.260.4.2 attach() [2/2]

```
template<typename T >
long L4Re::Rm::attach (
            T ** start,
            unsigned long size,
            Flags flags,
            L4::Ipc::Cap< Dataspace > mem,
            Offset offs = 0,
            unsigned char align = L4_PAGESHIFT ) const  [inline], [noexcept]
```

Attach a data space to a region.

Parameters

in,out	*start*	Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If L4Re::Rm::F::Search_addr is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If L4Re::Rm::F::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) to attach the data space to.
	size	Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size.
	flags	The flags control how and with which rights the dataspace is attached to the region. See L4Re::Rm::F::Attach_flags and L4Re::Rm::F::Region_flags. The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the `F::Eager_map` flag is set this function may also return L4Re::Dataspace::map error codes if the mapping fails.
	mem	Data space.
	offs	Offset into the data space to use.
	align	Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used.

Return values

0	Success
-L4_ENOENT	No area could be found (see L4Re::Rm::F::In_area)
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAVAIL	The given address is not available.
<0	IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see L4Re::Rm::find).

Definition at line 385 of file rm.

15.260.4.3 detach() [1/3]

```
int L4Re::Rm::detach (
            l4_addr_t addr,
            L4::Cap< Dataspace > * mem,
            L4::Cap< L4::Task > const & task = This_task ) const  [inline], [noexcept]
```

Detach and unmap a region from the address space.

Parameters

	addr	Virtual address of region, any address within the region is valid.
out	*mem*	Dataspace that is affected. Give 0 if not interested.
	task	This argument specifies the task where the pages are unmapped. Provide L4::Cap<L4::Task>::Invalid for none. The default is the current task.

Return values

L4Re::Rm::Detach_result	On success.
-L4_ENOENT	No region found.
<0	IPC errors

Frees a region in the virtual address space given by addr (address type). The corresponding part of the address space is now available again.

Definition at line 653 of file rm.

15.260.4.4 detach() [2/3]

```
int L4Re::Rm::detach (
            l4_addr_t start,
            unsigned long size,
            L4::Cap< Dataspace > * mem,
            L4::Cap< L4::Task > const & task ) const  [inline], [noexcept]
```

Detach and unmap all parts of the regions within the specified interval.

Parameters

	start	Start of area to detach, must be within region.
	size	Size of of area to detach (in bytes).
out	*mem*	Dataspace that is affected. Give 0 if not interested.
	task	This argument specifies the task where the pages are unmapped. Provide L4::Cap<L4::Task>::Invalid for none. The default is the current task.

Return values

L4Re::Rm::Detach_result	On success.
-L4_ENOENT	No region found.

Return values

<0	IPC errors

Frees all regions within the interval given by start and size. If a region overlaps the start or the end of the interval this region is only detached partly. If the interval is within one region the original region is split up into two separate regions.

Definition at line 663 of file rm.

15.260.4.5 detach() [3/3]

```
int L4Re::Rm::detach (
            void * addr,
            L4::Cap< Dataspace > * mem,
            L4::Cap< L4::Task > const & task = This_task ) const  [inline], [noexcept]
```

Detach and unmap a region from the address space.

Parameters

	addr	Virtual address of region, any address within the region is valid.
out	*mem*	Dataspace that is affected. Give 0 if not interested.
	task	This argument specifies the task where the pages are unmapped. Provide L4::Cap<L4::Task>::Invalid for none. The default is the current task.

Return values

L4Re::Rm::Detach_result	On success.
-L4_ENOENT	No region found.
<0	IPC errors

Frees a region in the virtual address space given by addr (address type). The corresponding part of the address space is now available again.

Definition at line 658 of file rm.

15.260.4.6 find()

```
int L4Re::Rm::find (
            l4_addr_t * addr,
            unsigned long * size,
            Offset * offset,
            L4Re::Rm::Flags * flags,
            L4::Cap< Dataspace > * m )  [inline], [noexcept]
```

Find a region given an address and size.

Parameters

in,out	*addr*	Address to look for. Returns the start address of the found region.
in,out	*size*	Size of the area to look for (in bytes). Returns the size of the found region (in bytes).
out	*offset*	Offset at the beginning of the region within the associated dataspace.
out	*flags*	Region flags, see F::Region_flags (and F::In_area).
out	*m*	Associated dataspace or paging service.

Return values

0	Success
-L4_EPERM	Operation not allowed.
-L4_ENOENT	No region found.
<0	IPC errors

This function returns the properties of the region that contains the area described by the addr and size parameter. If no such region is found but a reserved area, the area is returned and F::In_area is set in `flags`. Note, in the case of an area the `offset` and `m` return values are invalid.

```
                size-out
               /         \
              /           \
         addr-out          \
            ^_______________\
------------|----------------|------------------
|           | Region         |       Dataspace |
|           |_______|___|____|                 |
------------|-------|---|-----------------------
 \         /        |\ /
  \       /         | |> size-in
  offset-out        |
                    |> addr-in
```

Note

The value of the size input parameter should be 1 to assure that a region can be determined unambiguously.

Definition at line 575 of file rm.

15.260.4.7 free_area()

```
long L4Re::Rm::free_area (
            l4_addr_t addr )
```

Free an area from the region map.

Parameters

addr	An address within the area to free.

Return values

0	Success
-L4_ENOENT	No area found.
<0	IPC errors

Note

The data spaces that are attached to that area are not detached by this operation.

See also

reserve_area() for more information about areas.

15.260.4.8 get_areas()

```
long L4Re::Rm::get_areas (
            l4_addr_t start,
            L4::Ipc::Ret_array< Range > areas )
```

Return the list of areas whose starting addresses are higher or equal to `start` in the address space managed by this region map.

Parameters

	start	Virtual address from where to start searching.
`out`	*areas*	List of areas found in this region map.

Return values

>=0	Number of returned areas in the `areas` array.
<0	IPC errors

Note

The returned list of areas might not be complete and the caller shall use the function repeatedly with a start address one larger than the end address of the last area from the previous call.

15.260.4.9 get_regions()

```
long L4Re::Rm::get_regions (
            l4_addr_t start,
            L4::Ipc::Ret_array< Range > regions )
```

Return the list of regions whose starting addresses are higher or equal to `start` in the address space managed by this region map.

Parameters

	start	Virtual address from where to start searching.
out	*regions*	List of regions found in this region map.

Return values

>=0	Number of returned regions in the `regions` array.
<0	IPC errors

Note

The returned list of regions might not be complete and the caller shall use the function repeatedly with a start address one larger that the end address of the last region from the previous call.

15.260.4.10 reserve_area() [1/2]

```
long L4Re::Rm::reserve_area (
            l4_addr_t * start,
            unsigned long size,
            Flags flags = Flags(0),
            unsigned char align = L4_PAGESHIFT ) const  [inline], [noexcept]
```

Reserve the given area in the region map.

Parameters

in,out	*start*	The virtual start address of the area to reserve. Returns the start address of the area.
	size	The size of the area to reserve (in bytes).
	flags	Flags for the reserved area (see L4Re::Rm::F::Region_flags and L4Re::Rm::F::Attach_flags).
	align	Alignment of area if searched as bits (log2 value).

Return values

0	Success
-L4_EADDRNOTAVAIL	The given area cannot be reserved.
<0	IPC errors

This function reserves an area within the virtual address space managed by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = L4Re::Rm::F::Reserved), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the L4Re::Rm::F::In_area flag and a start address within the area itself.

Note

When searching for a free place in the virtual address space (with *flags* = L4Re::Rm::F::Search_addr), the space between *start* and the end of the virtual address space is searched.

Definition at line 272 of file rm.

15.260.4.11 reserve_area() [2/2]

```
template<typename T >
long L4Re::Rm::reserve_area (
            T ** start,
            unsigned long size,
            Flags flags = Flags(0),
            unsigned char align = L4_PAGESHIFT ) const  [inline], [noexcept]
```

Reserve the given area in the region map.

Parameters

in,out	*start*	The virtual start address of the area to reserve. Returns the start address of the area.
	size	The size of the area to reserve (in bytes).
	flags	Flags for the reserved area (see F::Region_flags and F::Attach_flags).
	align	Alignment of area if searched as bits (log2 value).

Return values

0	Success
-L4_EADDRNOTAVAIL	The given area cannot be reserved.
<0	IPC errors

For more information, please refer to the analogous function

See also

L4Re::Rm::reserve_area.

Definition at line 298 of file rm.

The documentation for this class was generated from the following files:

- l4/re/rm
- l4/re/impl/rm_impl.h

15.261 L4Re::Rm::F Struct Reference

Rm flags definitions.

Collaboration diagram for L4Re::Rm::F:

Public Types

- enum Attach_flags : l4_uint32_t { Search_addr = 0x20000 , In_area = 0x40000 , Eager_map = 0x80000 , Attach_mask = 0xf0000 }

  Flags for attach operation.
- enum Region_flags : l4_uint16_t { Rights_mask = 0x0f , R = Dataspace::F::R , W = Dataspace::F::W , X = Dataspace::F::X , RW = Dataspace::F::RW , RX = Dataspace::F::RX , RWX = Dataspace::F::RWX , Detach_free = 0x200 , Pager = 0x400 , Reserved = 0x800 , Caching_mask = Dataspace::F::Caching_mask , Cache_normal = Dataspace::F::Normal , Cache_buffered = Dataspace::F::Bufferable , Cache_uncached = Dataspace::F::Uncacheable , Ds_map_mask = 0xff , Region_flags_mask = 0xffff }

15.261.1 Detailed Description

Rm flags definitions.

Definition at line 106 of file rm.

15.261.2 Member Enumeration Documentation

15.261.2.1 Attach_flags

`enum L4Re::Rm::F::Attach_flags : l4_uint32_t`

Flags for attach operation.

Enumerator

Search_addr	Search for a suitable address range.
In_area	Search only in area, or map into area.
Eager_map	Eagerly map the attached data space in.
Attach_mask	Mask of all attach flags.

Definition at line 109 of file rm.

15.261.2.2 Region_flags

```
enum L4Re::Rm::F::Region_flags : l4_uint16_t
```

Enumerator

Rights_mask	Region rights.
R	Readable region.
W	Writable region.
X	Executable region.
RW	Readable and writable region.
RX	Readable and executable region.
RWX	Readable, writable and executable region.
Detach_free	Free the portion of the data space after detach.
Pager	Region has a pager.
Reserved	Region is reserved (blocked)
Caching_mask	Mask of all Rm cache bits.
Cache_normal	Cache bits for normal cacheable memory.
Cache_buffered	Cache bits for buffered (write combining) memory.
Cache_uncached	Cache bits for uncached memory.
Ds_map_mask	Mask for all bits for cache options and rights.
Region_flags_mask	Mask of all region flags.

Definition at line 123 of file rm.

The documentation for this struct was generated from the following file:

- l4/re/rm

15.262 L4Re::Rm::Range Struct Reference

A range of virtual addresses.

Collaboration diagram for L4Re::Rm::Range:

Data Fields

- l4_addr_t start

  First address of the range.
- l4_addr_t end

  Last address of the range.

15.262.1 Detailed Description

A range of virtual addresses.

Definition at line 587 of file rm.

The documentation for this struct was generated from the following file:

- l4/re/rm

15.263 L4Re::Smart_cap_auto< Unmap_flags > Class Template Reference

Helper for Unique_cap and Unique_del_cap.

Collaboration diagram for L4Re::Smart_cap_auto< Unmap_flags >:

15.263.1 Detailed Description

template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Smart_cap_auto< Unmap_flags >

Helper for Unique_cap and Unique_del_cap.

Definition at line 147 of file cap_alloc.

The documentation for this class was generated from the following file:

- l4/re/cap_alloc

15.264 L4Re::Smart_count_cap< Unmap_flags > Class Template Reference

Helper for Ref_cap and Ref_del_cap.

Collaboration diagram for L4Re::Smart_count_cap< Unmap_flags >:

Public Member Functions

- void free (L4::Cap_base &c) noexcept

  Free operation for L4::Smart_cap (decrement ref count and delete if 0).
- L4::Cap_base copy (L4::Cap_base const &src)

  Copy operation for L4::Smart_cap (increment ref count).

Static Public Member Functions

- static void invalidate (L4::Cap_base &c) noexcept

  Invalidate operation for L4::Smart_cap.

15.264.1 Detailed Description

template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Smart_count_cap< Unmap_flags >

Helper for Ref_cap and Ref_del_cap.

Definition at line 182 of file cap_alloc.

The documentation for this class was generated from the following file:

- l4/re/cap_alloc

15.265 L4Re::Util::Br_manager Class Reference

Buffer-register (BR) manager for L4::Server.

Inheritance diagram for L4Re::Util::Br_manager:

Collaboration diagram for L4Re::Util::Br_manager:

Public Member Functions

- Br_manager ()

  Make a buffer-register (BR) manager.
- int alloc_buffer_demand (Demand const &d)

  Tells the server to allocate buffers for the given demand.
- L4::Cap< void > get_rcv_cap (int i) const

  Get capability slot allocated to the given receive buffer.
- int realloc_rcv_cap (int i)

  Allocate a new capability for the given receive buffer.
- void set_rcv_cap_flags (unsigned long flags)

  Set the receive flags for the buffers.

- int add_timeout (L4::Ipc_svr::Timeout *, l4_kernel_clock_t)

  No timeouts handled by us.
- int remove_timeout (L4::Ipc_svr::Timeout *)

  No timeouts handled by us.
- void setup_wait (l4_utcb_t *utcb, L4::Ipc_svr::Reply_mode)

  setup_wait() used the server loop (L4::Server)

Protected Member Functions

- unsigned first_free_br () const

  Used for assigning BRs for a timeout.

Additional Inherited Members

15.265.1 Detailed Description

Buffer-register (BR) manager for L4::Server.

Implementation of the L4::Ipc_svr::Server_iface API for managing the server-side receive buffers needed for a set of server objects running within a server.

Definition at line 36 of file br_manager.

15.265.2 Member Function Documentation

15.265.2.1 alloc_buffer_demand()

```
int L4Re::Util::Br_manager::alloc_buffer_demand (
            Demand const & demand )  [inline], [virtual]
```

Tells the server to allocate buffers for the given demand.

Parameters

demand	The total server-side demand of receive buffers needed for a given interface, see Demand.

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Implements L4::Ipc_svr::Server_iface.

Definition at line 64 of file br_manager.

References L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc(), L4::Cap_base::cap(), L4Re::Util::cap_alloc, L4::Type_info::Demand::caps, L4_EINVAL, L4_ENOMEM, L4_EOK, L4_ERANGE, L4_UTCB_GENERIC_BUFFERS_SIZE, L4::Type_info::Demand::mem, and L4::Type_info::Demand::ports.

Here is the call graph for this function:

15.265.2.2 get_rcv_cap()

```
L4::Cap<void> L4Re::Util::Br_manager::get_rcv_cap (
            int index ) const  [inline], [virtual]
```

Get capability slot allocated to the given receive buffer.

Parameters

index	The receive buffer index of the expected capability argument (0 <= `index` < `caps` registered with alloc_buffer_demand()).

Precondition

0 <= `index` < `caps` registered with alloc_buffer_demand()

Returns

Capability slot currently allocated to the given receive buffer.

Implements L4::Ipc_svr::Server_iface.

Definition at line 95 of file br_manager.

References L4_CAP_MASK.

15.265.2.3 realloc_rcv_cap()

```
int L4Re::Util::Br_manager::realloc_rcv_cap (
            int index )  [inline], [virtual]
```

Allocate a new capability for the given receive buffer.

Parameters

index	The receive buffer index of the expected capability argument (0 <= `index` < `caps` registered with alloc_buffer_demand()).

Precondition

0 <= `index` < `caps` registered with alloc_buffer_demand()

Returns

0 on success, < 0 on error.

Implements L4::Ipc_svr::Server_iface.

Definition at line 103 of file br_manager.

References L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc(), L4::Cap_base::cap(), L4Re::Util::cap_alloc, L4_EINVAL, L4_ENOMEM, and L4_EOK.

Here is the call graph for this function:

15.265.2.4 set_rcv_cap_flags()

```
void L4Re::Util::Br_manager::set_rcv_cap_flags (
            unsigned long flags )  [inline]
```

Set the receive flags for the buffers.

Precondition

Must be called before any handlers are registered.

Parameters

flags	New receive capability flags, see l4_msg_item_consts_t.

Definition at line 127 of file br_manager.

References l4_assert.

The documentation for this class was generated from the following file:

- l4/re/util/br_manager

15.266 L4Re::Util::Br_manager_hooks Struct Reference

Predefined server-loop hooks for a server loop using the Br_manager.

Inheritance diagram for L4Re::Util::Br_manager_hooks:

Collaboration diagram for L4Re::Util::Br_manager_hooks:

Additional Inherited Members

15.266.1 Detailed Description

Predefined server-loop hooks for a server loop using the Br_manager.

This class can be used whenever a server loop including full management of receive buffer resources is needed.

Definition at line 174 of file br_manager.

The documentation for this struct was generated from the following file:

- l4/re/util/br_manager

15.267 L4Re::Util::Br_manager_timeout_hooks Struct Reference

Predefined server-loop hooks for a server with using the Br_manager and a timeout queue.

Inheritance diagram for L4Re::Util::Br_manager_timeout_hooks:

Collaboration diagram for L4Re::Util::Br_manager_timeout_hooks:

Additional Inherited Members

15.267.1 Detailed Description

Predefined server-loop hooks for a server with using the Br_manager and a timeout queue.

This class can be used for server loops that need the full package of buffer-register management and a timeout queue.

Definition at line 188 of file br_manager.

The documentation for this struct was generated from the following file:

- l4/re/util/br_manager

15.268 L4Re::Util::Cap_alloc_base Class Reference

Capability allocator.

Inherited by L4Re::Util::Cap_alloc< Size >.

Collaboration diagram for L4Re::Util::Cap_alloc_base:

Public Member Functions

- template<typename T >
  L4::Cap< T > alloc () noexcept

  Allocate a capability slot.
- template<typename T >
  void free (L4::Cap< T > const &cap, l4_cap_idx_t task=L4_INVALID_CAP, l4_umword_t unmap_↩ flags=L4_FP_ALL_SPACES) noexcept

  Free a capability slot.

15.268.1 Detailed Description

Capability allocator.

Definition at line 38 of file bitmap_cap_alloc.

The documentation for this class was generated from the following file:

- l4/re/util/bitmap_cap_alloc

15.269 L4Re::Util::Counter< COUNTER > Struct Template Reference

Counter for Counting_cap_alloc with variable data width.

Inheritance diagram for L4Re::Util::Counter< COUNTER >:

Collaboration diagram for L4Re::Util::Counter< COUNTER >:

15.269.1 Detailed Description

template<typename COUNTER = unsigned char>
struct L4Re::Util::Counter< COUNTER >

Counter for Counting_cap_alloc with variable data width.

Definition at line 36 of file counting_cap_alloc.

The documentation for this struct was generated from the following file:

- l4/re/util/counting_cap_alloc

15.270 L4Re::Util::Counting_cap_alloc< COUNTERTYPE > Class Template Reference

Internal reference-counting cap allocator.

Collaboration diagram for L4Re::Util::Counting_cap_alloc< COUNTERTYPE >:

Public Member Functions

- L4::Cap< void > alloc () noexcept

  Allocate a new capability slot.
- template<typename T >
  L4::Cap< T > alloc () noexcept

  Allocate a new capability slot.
- void take (L4::Cap< void > cap) noexcept

  Increase the reference counter for the capability.
- bool free (L4::Cap< void > cap, l4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_flags=L4_FP_ALL_SPACES) noexcept

  Free the capability.
- bool release (L4::Cap< void > cap, l4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_flags=L4_FP_ALL_SPACES) noexcept

  Decrease the reference counter for a capability.
- long last () noexcept

  Return highest capability id managed by this allocator.

Protected Member Functions

- Counting_cap_alloc () noexcept

  Create a new, empty allocator.
- void setup (void *m, long capacity, long bias) noexcept

  Set up the backing memory for the allocator and the area of managed capability slots.

15.270.1 Detailed Description

template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
class L4Re::Util::Counting_cap_alloc< COUNTERTYPE >

Internal reference-counting cap allocator.

This is intended for internal use only. L4Re applications should use L4Re::Util::cap_alloc().

Allocator for capability slots that automatically frees the slot and optionally unmaps the capability when the reference count goes down to zero. Reference counting must be done manually via take() and release(). The backing store for the reference counters must be provided in the setup() method. The allocator can recognize capability slots that are not managed by itself and does nothing on such slots.

Note

The user must ensure that the backing store is zero-initialized.

The user must ensure that the capability slots managed by this allocator are not used by a different allocator, see setup().

The operations in this class are not thread-safe.

Definition at line 75 of file counting_cap_alloc.

15.270.2 Constructor & Destructor Documentation

15.270.2.1 Counting_cap_alloc()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::Counting_cap_alloc ( )  [inline], [protected],
[noexcept]
```

Create a new, empty allocator.

Needs to be initialized with setup() before it can be used.

Definition at line 104 of file counting_cap_alloc.

15.270.3 Member Function Documentation

15.270.3.1 alloc() [1/2]

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
L4::Cap<void> L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc ( ) [inline], [noexcept]
```

Allocate a new capability slot.

Returns

The newly allocated capability slot, invalid if the allocator was exhausted.

Examples

examples/libs/l4re/c++/mem_alloc/ma+rm.cc, examples/libs/l4re/c++/shared_ds/ds_clnt.cc, and examples/libs/l4re/c++/shared_

Definition at line 135 of file counting_cap_alloc.

References L4_CAP_SHIFT.

Referenced by L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc(), L4Re::Util::Br_manager::alloc_buffer_demand(), L4Re::Util::make_ref_cap(), L4Re::Util::make_ref_del_cap(), L4Re::Util::make_shared_cap(), L4Re::Util::make_shared_del_cap(), L4Re::Util::make_unique_cap(), L4Re::Util::make_unique_del_cap(), and L4Re::Util::Br_manager::realloc_rcv_cap().

Here is the caller graph for this function:

15.270.3.2 alloc() [2/2]

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
template<typename T >
L4::Cap<T> L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc ( )  [inline], [noexcept]
```

Allocate a new capability slot.

Returns

The newly allocated capability slot, invalid if the allocator was exhausted.

Definition at line 153 of file counting_cap_alloc.

References L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc().

Here is the call graph for this function:

15.270.3.3 free()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free (
            L4::Cap< void > cap,
            l4_cap_idx_t task = L4_INVALID_CAP,
            unsigned unmap_flags = L4_FP_ALL_SPACES )  [inline], [noexcept]
```

Free the capability.

Parameters

cap	Capability to free.
task	If set, task to unmap the capability from.
unmap_flags	Flags for unmap, see l4_unmap_flags_t.

Precondition

The capability has been allocated. Calling free twice on a capability managed by this allocator results in undefined behaviour.

Returns

True, if the capability was managed by this allocator.

Examples

examples/libs/l4re/c++/mem_alloc/ma+rm.cc, examples/libs/l4re/c++/shared_ds/ds_clnt.cc, examples/libs/l4re/c++/shared_ds/d and examples/libs/l4re/streammap/client.cc.

Definition at line 191 of file counting_cap_alloc.

References l4_assert, l4_is_valid_cap(), and l4_task_unmap().

Referenced by L4Re::Util::Smart_cap_auto< Unmap_flags >::free(), and L4Re::Util::Object_registry::unregister_obj().

Here is the call graph for this function:

Here is the caller graph for this function:

15.270.3.4 release()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::release (
            L4::Cap< void > cap,
            l4_cap_idx_t task = L4_INVALID_CAP,
            unsigned unmap_flags = L4_FP_ALL_SPACES )  [inline], [noexcept]
```

Decrease the reference counter for a capability.

Parameters

cap	Capability to release.
task	If set, task to unmap the capability from.
unmap_flags	Flags for unmap, see l4_unmap_flags_t.

Precondition

The capability has been allocated. Calling release on a free capability results in undefined behaviour.

Returns

True, if the capability was freed as a result of this operation. If false is returned the capability is either still in use or is not managed by this allocator.

Does nothing apart from returning false if the capability is not managed by this allocator.

Definition at line 229 of file counting_cap_alloc.

References L4::dec, l4_assert, L4_INVALID_CAP, and l4_task_unmap().

Referenced by L4Re::Util::Smart_count_cap< Unmap_flags >::free().

Here is the call graph for this function:

Here is the caller graph for this function:

15.270.3.5 setup()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::setup (
            void * m,
            long capacity,
            long bias )  [inline], [protected], [noexcept]
```

Set up the backing memory for the allocator and the area of managed capability slots.

Parameters

m	Pointer to backing memory.
capacity	Number of capabilities that can be stored.
bias	First capability id to use by this allocator.

The allocator will manage the capability slots between `bias` and `bias` + `capacity` - 1 (inclusive). It is the responsibility of the user to ensure that these slots are not used otherwise.

Definition at line 121 of file counting_cap_alloc.

15.270.3.6 take()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::take (
            L4::Cap< void > cap )  [inline], [noexcept]
```

Increase the reference counter for the capability.

Parameters

cap	Capability, whose reference counter should be increased.

If the capability was still free, it will be automatically allocated. Silently does nothing if the capability is not managed by this allocator.

Definition at line 168 of file counting_cap_alloc.

Referenced by L4Re::Util::Smart_count_cap< Unmap_flags >::copy().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/re/util/counting_cap_alloc

15.271 L4Re::Util::Dataspace_svr Class Reference

Dataspace server class.

Collaboration diagram for L4Re::Util::Dataspace_svr:

Public Member Functions

- int map (Dataspace::Offset offset, Dataspace::Map_addr local_addr, Dataspace::Flags flags, Dataspace::↩ Map_addr min_addr, Dataspace::Map_addr max_addr, L4::Ipc::Snd_fpage &memory)

  Map a region of the dataspace.
- virtual int map_hook (Dataspace::Offset offs, Dataspace::Flags flags, Dataspace::Map_addr min, Dataspace::Map_addr max)

  A hook that is called as the first operation in each map request.
- virtual void take () noexcept

  Take a reference to this dataspace.
- virtual unsigned long release () noexcept

  Release a reference to this dataspace.
- virtual long copy (l4_addr_t dst_offs, l4_umword_t src_id, l4_addr_t src_offs, unsigned long size) noexcept

  Copy from src dataspace to this destination dataspace.
- virtual long clear (unsigned long offs, unsigned long size) const noexcept

  Clear a region in the dataspace.
- virtual long allocate (l4_addr_t offset, l4_size_t size, unsigned access) noexcept

  Allocate a region within a dataspace.
- virtual unsigned long page_shift () const noexcept

  Define the size of the flexpage to map.
- virtual bool is_static () const noexcept

  Return whether the dataspace is static.

15.271.1 Detailed Description

Dataspace server class.

The default implementation of the interface provides a continuous dataspace with contiguous pages.

Definition at line 40 of file dataspace_svr.

15.271.2 Member Function Documentation

15.271.2.1 allocate()

```
virtual long L4Re::Util::Dataspace_svr::allocate (
            l4_addr_t offset,
            l4_size_t size,
            unsigned access )  [inline], [virtual], [noexcept]
```

Allocate a region within a dataspace.

Parameters

offset	Offset in the dataspace, in bytes.
size	Size of the range, in bytes.
access	Access mode with which the memory backing the dataspace region should be allocated.

Return values

0	Success
<0	Error

Definition at line 157 of file dataspace_svr.

References L4_ENODEV.

15.271.2.2 clear()

```
virtual long L4Re::Util::Dataspace_svr::clear (
            unsigned long offs,
            unsigned long size ) const  [virtual], [noexcept]
```

Clear a region in the dataspace.

Parameters

offs	Start of the region
size	Size of the region

Return values

0	Success
<0	Error

15.271.2.3 copy()

```
virtual long L4Re::Util::Dataspace_svr::copy (
            l4_addr_t dst_offs,
            l4_umword_t src_id,
            l4_addr_t src_offs,
            unsigned long size )  [inline], [virtual], [noexcept]
```

Copy from src dataspace to this destination dataspace.

Parameters

dst_offs	Offset into the destination dataspace
src_id	Local id of the source dataspace
src_offs	Offset into the source dataspace
size	Number of bytes to copy

Return values

>=0	Number of bytes copied
<0	An error occured. The error code may depend on the implementation.

Definition at line 128 of file dataspace_svr.

References L4_ENODEV.

15.271.2.4 is_static()

```
virtual bool L4Re::Util::Dataspace_svr::is_static ( ) const  [inline], [virtual], [noexcept]
```

Return whether the dataspace is static.

Returns

True if dataspace is static

Definition at line 173 of file dataspace_svr.

15.271.2.5 map()

```
int L4Re::Util::Dataspace_svr::map (
            Dataspace::Offset offset,
            Dataspace::Map_addr local_addr,
            Dataspace::Flags flags,
            Dataspace::Map_addr min_addr,
            Dataspace::Map_addr max_addr,
            L4::Ipc::Snd_fpage & memory )
```

Map a region of the dataspace.

Parameters

	offset	Offset to start within data space
	local_addr	Local address to map to.
	flags	Dataspace flags, see L4Re::Dataspace::F::Flags.
	min_addr	Defines start of receive window.
	max_addr	Defines end of receive window.
out	*memory*	Send fpage to map

Return values

0	Success
<0	Error

15.271.2.6 map_hook()

```
virtual int L4Re::Util::Dataspace_svr::map_hook (
            Dataspace::Offset offs,
            Dataspace::Flags flags,
            Dataspace::Map_addr min,
            Dataspace::Map_addr max )  [inline], [virtual]
```

A hook that is called as the first operation in each map request.

Parameters

offs	Offs param to map
flags	Flags param to map
min	Min param to map
max	Max param to map

Return values

<0	Error and the map request will be aborted with that error.
>=0	Success

See also

map

Definition at line 89 of file dataspace_svr.

References cxx::max(), and cxx::min().

Here is the call graph for this function:

15.271.2.7 page_shift()

```
virtual unsigned long L4Re::Util::Dataspace_svr::page_shift ( ) const  [inline], [virtual],
[noexcept]
```

Define the size of the flexpage to map.

Returns

flexpage size

Definition at line 165 of file dataspace_svr.

References L4_LOG2_PAGESIZE.

15.271.2.8 release()

```
virtual unsigned long L4Re::Util::Dataspace_svr::release ( )  [inline], [virtual], [noexcept]
```

Release a reference to this dataspace.

Returns

Number of references to the dataspace

Default does nothing and returns always zero.

Definition at line 113 of file dataspace_svr.

15.271.2.9 take()

```
virtual void L4Re::Util::Dataspace_svr::take ( )  [inline], [virtual], [noexcept]
```

Take a reference to this dataspace.

Default does nothing.

Definition at line 103 of file dataspace_svr.

The documentation for this class was generated from the following file:

- l4/re/util/dataspace_svr

15.272 L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference

An event buffer consumer.

Inheritance diagram for L4Re::Util::Event_buffer_consumer_t< PAYLOAD >:

Collaboration diagram for L4Re::Util::Event_buffer_consumer_t< PAYLOAD >:

Public Member Functions

- template<typename CB , typename D >
  void foreach_available_event (CB const &cb, D data=D())

  Call function on every available event.
- template<typename CB , typename D >
  void process (L4::Cap< L4::Irq > irq, L4::Cap< L4::Thread > thread, CB const &cb, D data=D())

  Continuously wait for events and process them.

15.272.1 Detailed Description

template<typename PAYLOAD>
class L4Re::Util::Event_buffer_consumer_t< PAYLOAD >

An event buffer consumer.

Definition at line 93 of file event_buffer.

15.272.2 Member Function Documentation

15.272.2.1 foreach_available_event()

```
template<typename PAYLOAD >
template<typename CB , typename D >
void L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::foreach_available_event (
            CB const & cb,
            D data = D() )  [inline]
```

Call function on every available event.

Parameters

cb	Function callback.
data	Data to pass as an argument to the callback.

Definition at line 104 of file event_buffer.

Referenced by L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process().

Here is the caller graph for this function:

15.272.2.2 process()

```
template<typename PAYLOAD >
template<typename CB , typename D >
void L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process (
            L4::Cap< L4::Irq > irq,
            L4::Cap< L4::Thread > thread,
            CB const & cb,
            D data = D() )  [inline]
```

Continuously wait for events and process them.

Parameters

irq	Event signal to wait for.
thread	Thread capability of the thread calling this function.
cb	Callback function that is called for each received event.
data	Data to pass as an argument to the processing callback.

Note

This function never returns.

Definition at line 125 of file event_buffer.

References L4::Rcv_endpoint::bind_thread(), L4::Cap_base::cap(), L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::foreach_ava
l4_error(), l4_ipc_error(), L4_IPC_NEVER, l4_irq_receive(), and l4_utcb().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/re/util/event_buffer

15.273 L4Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference

Event_buffer utility class.

Inheritance diagram for L4Re::Util::Event_buffer_t< PAYLOAD >:

Collaboration diagram for L4Re::Util::Event_buffer_t< PAYLOAD >:

Public Member Functions

- void * buf () const noexcept

  Return the buffer.
- long attach (L4::Cap< L4Re::Dataspace > ds, L4::Cap< L4Re::Rm > rm) noexcept

  Attach event buffer from address space.
- long detach (L4::Cap< L4Re::Rm > rm) noexcept

  Detach event buffer from address space.

15.273.1 Detailed Description

template<typename PAYLOAD>
class L4Re::Util::Event_buffer_t< PAYLOAD >

Event_buffer utility class.

Definition at line 36 of file event_buffer.

15.273.2 Member Function Documentation

15.273.2.1 attach()

```
template<typename PAYLOAD >
long L4Re::Util::Event_buffer_t< PAYLOAD >::attach (
            L4::Cap< L4Re::Dataspace > ds,
            L4::Cap< L4Re::Rm > rm )  [inline], [noexcept]
```

Attach event buffer from address space.

Parameters

ds	Dataspace of the event buffer.
rm	Region manager to attach buffer to.

Returns

0 on success, negative error code otherwise.

Definition at line 56 of file event_buffer.

References L4::Ipc::make_cap_rw(), L4Re::Rm::F::RW, and L4Re::Rm::F::Search_addr.

Here is the call graph for this function:

15.273.2.2 buf()

```
template<typename PAYLOAD >
void* L4Re::Util::Event_buffer_t< PAYLOAD >::buf ( ) const  [inline], [noexcept]
```

Return the buffer.

Returns

Pointer to the event buffer.

Definition at line 46 of file event_buffer.

15.273.2.3 detach()

```
template<typename PAYLOAD >
long L4Re::Util::Event_buffer_t< PAYLOAD >::detach (
            L4::Cap< L4Re::Rm > rm )  [inline], [noexcept]
```

Detach event buffer from address space.

Parameters

rm	Region manager to detach buffer from.

Returns

0 on success, negative error code otherwise.

Definition at line 78 of file event_buffer.

The documentation for this class was generated from the following file:

- l4/re/util/event_buffer

15.274 L4Re::Util::Event_svr< SVR > Class Template Reference

Convenience wrapper for implementing an event server.

Inherits L4Re::Util::Icu_cap_array_svr< ICU >.

Collaboration diagram for L4Re::Util::Event_svr< SVR >:

Public Member Functions

- long op_get_buffer (L4Re::Event::Rights, L4::Ipc::Cap< L4Re::Dataspace > &ds)

  Handle L4Re::Event protocol.

15.274.1 Detailed Description

template<typename SVR>
class L4Re::Util::Event_svr< SVR >

Convenience wrapper for implementing an event server.

See also

L4Re::Event, L4Re::Util::Event_t

Definition at line 39 of file event_svr.

The documentation for this class was generated from the following file:

- l4/re/util/event_svr

15.275 L4Re::Util::Event_t< PAYLOAD > Class Template Reference

Convenience wrapper for getting access to an event object.

Collaboration diagram for L4Re::Util::Event_t< PAYLOAD >:

Public Types

- enum Mode { Mode_irq , Mode_polling }

  Modes of operation.

Public Member Functions

- template<typename IRQ_TYPE >
  int init (L4::Cap< L4Re::Event > event, L4Re::Env const *env=L4Re::Env::env(), L4Re::Cap_alloc *ca=L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))

  Initialise an event object.
- int init_poll (L4::Cap< L4Re::Event > event, L4Re::Env const *env=L4Re::Env::env(), L4Re::Cap_alloc *ca=L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))

  Initialise an event object in polling mode.
- L4Re::Event_buffer_t< PAYLOAD > & buffer ()

  Get event buffer.
- L4::Cap< L4::Triggerable > irq () const

  Get event IRQ.

15.275.1 Detailed Description

template<typename PAYLOAD>
class L4Re::Util::Event_t< PAYLOAD >

Convenience wrapper for getting access to an event object.

After calling init() the class supplies the event-buffer and the associated IRQ object.

Definition at line 43 of file event.

15.275.2 Member Enumeration Documentation

15.275.2.1 Mode

```
template<typename PAYLOAD >
enum L4Re::Util::Event_t::Mode
```

Modes of operation.

Enumerator

Mode_irq	Create an IRQ and attach, to get notifications.
Mode_polling	Do not use an IRQ.

Definition at line 49 of file event.

15.275.3 Member Function Documentation

15.275.3.1 buffer()

```
template<typename PAYLOAD >
L4Re::Event_buffer_t<PAYLOAD>& L4Re::Util::Event_t< PAYLOAD >::buffer ( ) [inline]
```

Get event buffer.

Returns

Event buffer object.

Definition at line 159 of file event.

15.275.3.2 init()

```
template<typename PAYLOAD >
template<typename IRQ_TYPE >
int L4Re::Util::Event_t< PAYLOAD >::init (
            L4::Cap< L4Re::Event > event,
            L4Re::Env const * env = L4Re::Env::env(),
            L4Re::Cap_alloc * ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc) )
[inline]
```

Initialise an event object.

Template Parameters

IRQ_TYPE	Type used for handling notifications from the event provider. This must be derived from L4::Triggerable.

Parameters

event	Capability to event.
env	Pointer to L4Re-Environment
ca	Pointer to capability allocator.

Return values

0	Success
-L4_ENOMEM	No memory to allocate required capabilities.
<0	Other IPC errors.

Definition at line 70 of file event.

References L4_ENOMEM, l4_error(), L4::Ipc::make_cap_rw(), L4Re::Rm::F::RW, and L4Re::Rm::F::Search_addr.

Here is the call graph for this function:

15.275.3.3 init_poll()

```
template<typename PAYLOAD >
int L4Re::Util::Event_t< PAYLOAD >::init_poll (
            L4::Cap< L4Re::Event > event,
            L4Re::Env const * env = L4Re::Env::env(),
            L4Re::Cap_alloc * ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc) )
[inline]
```

Initialise an event object in polling mode.

Parameters

event	Capability to event.
env	Pointer to L4Re-Environment
ca	Pointer to capability allocator.

Return values

0	Success
-L4_ENOMEM	No memory to allocate required capabilities.
<0	Other IPC errors.

Definition at line 123 of file event.

References L4_ENOMEM, L4::Ipc::make_cap_rw(), L4Re::Rm::F::RW, and L4Re::Rm::F::Search_addr.

Here is the call graph for this function:

15.275.3.4 irq()

```
template<typename PAYLOAD >
L4::Cap<L4::Triggerable> L4Re::Util::Event_t< PAYLOAD >::irq ( ) const  [inline]
```

Get event IRQ.

Returns

Event IRQ.

Definition at line 166 of file event.

The documentation for this class was generated from the following file:

- l4/re/util/event

15.276 L4Re::Util::Item_alloc_base Class Reference

Item allocator.

Inherited by L4Re::Util::Item_alloc< Bits >.

Collaboration diagram for L4Re::Util::Item_alloc_base:

15.276.1 Detailed Description

Item allocator.

Definition at line 38 of file item_alloc.

The documentation for this class was generated from the following file:

- l4/re/util/item_alloc

15.277 L4Re::Util::Names::Name Class Reference

Name class.

Inheritance diagram for L4Re::Util::Names::Name:

Collaboration diagram for L4Re::Util::Names::Name:

Additional Inherited Members

15.277.1 Detailed Description

Name class.

Definition at line 42 of file name_space_svr.

The documentation for this class was generated from the following file:

- l4/re/util/name_space_svr

15.278 L4Re::Util::Names::Name_space Class Reference

Abstract server-side implementation of the L4::Namespace interface.

Collaboration diagram for L4Re::Util::Names::Name_space:

Protected Member Functions

- virtual Entry ∗ alloc_dynamic_entry (Name const &n, unsigned flags)=0

  Allocate a new entry for the given name.
- virtual void free_dynamic_entry (Entry ∗e)=0

  Free an entry previously allocated with alloc_dynamic_entry().
- virtual int get_epiface (l4_umword_t data, bool is_local, L4::Epiface ∗∗lo)=0

  Return a pointer to the epiface assigned to a given label.
- virtual int copy_receive_cap (L4::Cap< void > ∗cap)=0

  Return the receive capability for permanent use.
- virtual void free_capability (L4::Cap< void > cap)=0

  Free a capability previously acquired with copy_receive_cap().
- virtual void free_epiface (L4::Epiface ∗epiface)=0

  Free epiface previously acquired with get_epiface().

15.278.1 Detailed Description

Abstract server-side implementation of the L4::Namespace interface.

Definition at line 184 of file name_space_svr.

15.278.2 Member Function Documentation

15.278.2.1 alloc_dynamic_entry()

```
virtual Entry* L4Re::Util::Names::Name_space::alloc_dynamic_entry (
            Name const & n,
            unsigned flags )  [protected], [pure virtual]
```

Allocate a new entry for the given name.

Parameters

n	Name of the entry, must be copied.
flags	Entry flags, see Obj::Flags.

Returns

A pointer to the newly allocated entry or NULL on error.

This method is called when a new entry was received. It must allocate memory, copy `n` out of the receive buffer and wrap everything in an Entry.

15.278.2.2 copy_receive_cap()

```
virtual int L4Re::Util::Names::Name_space::copy_receive_cap (
            L4::Cap< void > * cap )  [protected], [pure virtual]
```

Return the receive capability for permanent use.

Parameters

out	*cap*	Capability slot with the received capability. Must be permanently available until free_capability() is called.

This method is called when a new entry is registered together with a capability mapping. It must decide whether and where to store the capability and return the final capability slot. Typical implementations return the capability slot in the receive window and allocate a new receive window.

15.278.2.3 free_capability()

```
virtual void L4Re::Util::Names::Name_space::free_capability (
            L4::Cap< void > cap )  [protected], [pure virtual]
```

Free a capability previously acquired with copy_receive_cap().

Parameters

in	*cap*	Capability to free.

Counterpart of copy_receive_cap. Free the capability slot when the entry is deleted or changed.

15.278.2.4 free_dynamic_entry()

```
virtual void L4Re::Util::Names::Name_space::free_dynamic_entry (
            Entry * e )  [protected], [pure virtual]
```

Free an entry previously allocated with alloc_dynamic_entry().

Parameters

e	Entry to free.

15.278.2.5 free_epiface()

```
virtual void L4Re::Util::Names::Name_space::free_epiface (
            L4::Epiface * epiface )  [protected], [pure virtual]
```

Free epiface previously acquired with get_epiface().

Parameters

in	*epiface*	Epiface to free.

Called when an entry that points to an epiface is deleted allowing implementations that hold resources to free them.

15.278.2.6 get_epiface()

```
virtual int L4Re::Util::Names::Name_space::get_epiface (
            l4_umword_t data,
            bool is_local,
            L4::Epiface ** lo )  [protected], [pure virtual]
```

Return a pointer to the epiface assigned to a given label.

Parameters

in	*data*	Label or in the local case the capability slot of the receiving capability.
in	*is_local*	If true, a local capability slot was supplied, if false the label of a locally bound IPC gate.
out	*lo*	Pointer to epiface responsible for the capability.

Returns

L4_EOK if a valid interface could be found or an error message otherwise.

This method is called when a new entry is registered and some local ID was received for the capability. In this case, the generic implementation needs to get the epiface in order to get the capability.

The callee must make sure that the epiface remains valid until free_epiface is called. In particular, the capability slot must not be reallocated as long as the namespace server holds a reference to the epiface.

The documentation for this class was generated from the following file:

- l4/re/util/name_space_svr

15.279 L4Re::Util::Object_registry Class Reference

A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

Inheritance diagram for L4Re::Util::Object_registry:

Collaboration diagram for L4Re::Util::Object_registry:

Public Member Functions

- Object_registry (L4::Ipc_svr::Server_iface ∗sif)

  Create a registry for the main thread of the task using the default factory.
- Object_registry (L4::Ipc_svr::Server_iface ∗sif, L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory)

  Create a registry for arbitrary threads.
- L4::Cap< void > register_obj (L4::Epiface ∗o, char const ∗service) override

  Register a new server object to a pre-allocated receive endpoint.
- L4::Cap< void > register_obj (L4::Epiface ∗o) override

  Register a new server object on a newly allocated capability.
- L4::Cap< L4::Irq > register_irq_obj (L4::Epiface ∗o) override

  Register a handler for an interrupt.
- L4::Cap< L4::Rcv_endpoint > register_obj (L4::Epiface ∗o, L4::Cap< L4::Rcv_endpoint > ep) override

  Register a handler for an already existing interrupt.
- void unregister_obj (L4::Epiface ∗o, bool unmap=true) override

  Remove a server object from the handler list.

Additional Inherited Members

15.279.1 Detailed Description

A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

This class manages most of the setup of a server object. If necessary, an IPC gate is created, the specified thread is bound to the IPC gate. Incoming IPC is dispatched to the server object based on the label of the IPC gate.

The object registry is also able to manage IRQ endpoints. They require a different method for the object creation. Otherwise they are handled in the same way as IPC gates: a server object is responsible to process the incoming interrupts.

Definition at line 52 of file object_registry.

15.279.2 Constructor & Destructor Documentation

15.279.2.1 Object_registry() [1/2]

```
L4Re::Util::Object_registry::Object_registry (
            L4::Ipc_svr::Server_iface * sif )  [inline], [explicit]
```

Create a registry for the main thread of the task using the default factory.

Parameters

sif	Server loop interface.

Definition at line 78 of file object_registry.

15.279.2.2 Object_registry() [2/2]

```
L4Re::Util::Object_registry::Object_registry (
            L4::Ipc_svr::Server_iface * sif,
            L4::Cap< L4::Thread > server,
            L4::Cap< L4::Factory > factory )  [inline]
```

Create a registry for arbitrary threads.

Parameters

sif	Server loop interface.
server	Capability to the thread that executes the server objects.
factory	Capability to a factory object capable of creating new IPC gates.

Definition at line 92 of file object_registry.

15.279.3 Member Function Documentation

15.279.3.1 register_irq_obj()

```
L4::Cap<L4::Irq> L4Re::Util::Object_registry::register_irq_obj (
            L4::Epiface * o )  [inline], [override], [virtual]
```

Register a handler for an interrupt.

Parameters

o	Server object that handles IRQs.

Return values

L4::Cap<L4::Irq>	Capability to a new IRQ object on success.
L4::Cap<L4::Irq>::Invalid	The allocation of the IRQ has failed.

The IRQ will be newly allocated using the registry's factory object. The caller must call unregister_obj() to free all resources.

Implements L4::Registry_iface.

Examples

examples/libs/l4re/c++/shared_ds/ds_srv.cc.

Definition at line 238 of file object_registry.

15.279.3.2 register_obj() [1/3]

```
L4::Cap<void> L4Re::Util::Object_registry::register_obj (
            L4::Epiface * o )  [inline], [override], [virtual]
```

Register a new server object on a newly allocated capability.

Parameters

o	Server object that handles IPC requests.

Return values

L4::Cap<void>	A valid capability to a new IPC gate.
L4::Cap<void>::Invalid	The allocation of the IPC gate has failed.

The IPC gate will be allocated using the registry's factory. The caller must call unregister_obj() to free all resources.

Implements L4::Registry_iface.

Definition at line 222 of file object_registry.

15.279.3.3 register_obj() [2/3]

```
L4::Cap<void> L4Re::Util::Object_registry::register_obj (
            L4::Epiface * o,
            char const * service )  [inline], [override], [virtual]
```

Register a new server object to a pre-allocated receive endpoint.

Parameters

o	Server object that handles IPC requests.
service	Name of a pre-allocated receive endpoint.

Return values

L4::Cap<void>	The capability known as `service` on success.
L4::Cap<void>::Invalid	No capability with the given name found.

The interface must be freed with unregister_obj() by the caller to unbind the thread from the capability.

Implements L4::Registry_iface.

Examples

examples/clntsrv/server.cc, examples/libs/l4re/c++/shared_ds/ds_srv.cc, and examples/libs/l4re/streammap/server.cc.

Definition at line 205 of file object_registry.

15.279.3.4 register_obj() [3/3]

```
L4::Cap<L4::Rcv_endpoint> L4Re::Util::Object_registry::register_obj (
            L4::Epiface * o,
            L4::Cap< L4::Rcv_endpoint > ep )  [inline], [override], [virtual]
```

Register a handler for an already existing interrupt.

Parameters

o	Server object that handles the IPC.
ep	Capability to a receive endpoint, may be a hardware or software interrupt or an IPC gate.

Return values

L4::Cap<L4::Rcv_endpoint>	Capability `ep` on success.
L4::Cap<L4::Rcv_endpoint>::Invalid	The IRQ attach operation has failed.

The interface must be freed with unregister_obj() by the caller to unbind the thread from the capability.

Implements L4::Registry_iface.

Definition at line 260 of file object_registry.

15.279.3.5 unregister_obj()

```
void L4Re::Util::Object_registry::unregister_obj (
            L4::Epiface * o,
            bool unmap = true )  [inline], [override], [virtual]
```

Remove a server object from the handler list.

Parameters

o	Server object to unbind.
unmap	Specifies if the object capability shall be unmapped (true) or not. The default (true) is to unmap the capability.

The capability used by the server object will be unmapped if `unmap` is true.

Implements L4::Registry_iface.

Definition at line 276 of file object_registry.

References L4::Thread::Modify_senders::add(), L4Re::Util::cap_alloc, L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free(), L4_FP_ALL_SPACES, L4::Thread::modify_senders(), L4::Epiface::obj_cap(), and L4::Epiface::set_server().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/re/util/object_registry

15.280 L4Re::Util::Ref_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap of the capability selector.

Collaboration diagram for L4Re::Util::Ref_cap< T >:

15.280.1 Detailed Description

template<typename T>
struct L4Re::Util::Ref_cap< T >

Automatic capability that implements automatic free and unmap of the capability selector.

Template Parameters

T	Type of the object that is referred by the capability.

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Usage:

```
L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap global_ds_cap;
{
  L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap
    ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
  // reference count for the allocated cap selector is now 1
  // use the dataspace cap
  L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));
  global_ds_cap = ds_cap;
  // reference count is now 2
  ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 165 of file cap_alloc.

The documentation for this struct was generated from the following file:

- l4/re/util/cap_alloc

15.281 L4Re::Util::Ref_del_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector.

Collaboration diagram for L4Re::Util::Ref_del_cap< T >:

15.281.1 Detailed Description

template<typename T>
struct L4Re::Util::Ref_del_cap< T >

Automatic capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

T	Type of the object that is referred by the capability.

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to Ref_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap global_ds_cap;
{
  L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap
    ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
  // reference count for the allocated cap selector is now 1
  // use the dataspace cap
  L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));
  global_ds_cap = ds_cap;
  // reference count is now 2
  ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).
```

Definition at line 206 of file cap_alloc.

The documentation for this struct was generated from the following file:

- l4/re/util/cap_alloc

15.282 L4Re::Util::Registry_server< LOOP_HOOKS > Class Template Reference

A server loop object which has a Object_registry included.

Inheritance diagram for L4Re::Util::Registry_server< LOOP_HOOKS >:

Collaboration diagram for L4Re::Util::Registry_server< LOOP_HOOKS >:

Public Member Functions

- Registry_server ()

  Create a new server loop object for the main thread of the task.
- Registry_server (l4_utcb_t *, L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory)

  Create a new server loop object for an arbitrary thread and factory.
- Registry_server (L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory)

Create a new server loop object for an arbitrary thread and factory.

- Object_registry const * registry () const

  Return registry of this server loop.
- Object_registry * registry ()

  Return registry of this server loop.
- void L4_NORETURN loop (l4_utcb_t *utcb=l4_utcb())

  Start the server loop.

Additional Inherited Members

15.282.1 Detailed Description

template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
class L4Re::Util::Registry_server< LOOP_HOOKS >

A server loop object which has a Object_registry included.

Examples

examples/clntsrv/server.cc, examples/libs/l4re/c++/shared_ds/ds_srv.cc, and examples/libs/l4re/streammap/server.cc.

Definition at line 306 of file object_registry.

15.282.2 Constructor & Destructor Documentation

15.282.2.1 Registry_server() [1/3]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server ( ) [inline]
```

Create a new server loop object for the main thread of the task.

Precondition

Must be called from the main thread or behaviour is undefined.

Definition at line 318 of file object_registry.

15.282.2.2 Registry_server() [2/3]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (
            l4_utcb_t * ,
            L4::Cap< L4::Thread > server,
            L4::Cap< L4::Factory > factory ) [inline]
```

Create a new server loop object for an arbitrary thread and factory.

Parameters

utcb	The UTCB of the thread running the server loop.
server	Capability to thread running the server loop.
factory	Capability to factory object used to create new IPC gates.

Deprecated {Note that this variant of the constructor is deprecated, please do not supply the UTCB pointer, it's not used.}

Definition at line 331 of file object_registry.

15.282.2.3 Registry_server() [3/3]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (
            L4::Cap< L4::Thread > server,
            L4::Cap< L4::Factory > factory )  [inline]
```

Create a new server loop object for an arbitrary thread and factory.

Parameters

server	Capability to thread running the server loop.
factory	Capability to factory object used to create new IPC gates.

Definition at line 342 of file object_registry.

15.282.3 Member Function Documentation

15.282.3.1 loop()

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
void L4_NORETURN L4Re::Util::Registry_server< LOOP_HOOKS >::loop (
            l4_utcb_t * utcb = l4_utcb() )  [inline]
```

Start the server loop.

Parameters

utcb	The UTCB of the thread running the server loop, defaults to l4_utcb().

Examples

examples/clntsrv/server.cc, examples/libs/l4re/c++/shared_ds/ds_srv.cc, and examples/libs/l4re/streammap/server.cc.

Definition at line 358 of file object_registry.

The documentation for this class was generated from the following file:

- l4/re/util/object_registry

15.283 L4Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference

Helper for Unique_cap and Unique_del_cap.

Collaboration diagram for L4Re::Util::Smart_cap_auto< Unmap_flags >:

Static Public Member Functions

- static void free (L4::Cap_base &c)

  Free operation for L4::Smart_cap.
- static void invalidate (L4::Cap_base &c)

  Invalidate operation for L4::Smart_cap.
- static L4::Cap_base copy (L4::Cap_base const &src)

  Copy operation for L4::Smart_cap.

15.283.1 Detailed Description

template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Util::Smart_cap_auto< Unmap_flags >

Helper for Unique_cap and Unique_del_cap.

Definition at line 59 of file cap_alloc.

The documentation for this class was generated from the following file:

- l4/re/util/cap_alloc

15.284 L4Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference

Helper for Ref_cap and Ref_del_cap.

Collaboration diagram for L4Re::Util::Smart_count_cap< Unmap_flags >:

Static Public Member Functions

- static void free (L4::Cap_base &c) noexcept

  Free operation for L4::Smart_cap (decrement ref count and delete if 0).
- static void invalidate (L4::Cap_base &c) noexcept

  Invalidate operation for L4::Smart_cap.
- static L4::Cap_base copy (L4::Cap_base const &src)

  Copy operation for L4::Smart_cap (increment ref count).

15.284.1 Detailed Description

template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Util::Smart_count_cap< Unmap_flags >

Helper for Ref_cap and Ref_del_cap.

Definition at line 99 of file cap_alloc.

The documentation for this class was generated from the following file:

- l4/re/util/cap_alloc

15.285 L4Re::Util::Vcon_svr< SVR > Class Template Reference

Console server template class.

Collaboration diagram for L4Re::Util::Vcon_svr< SVR >:

15.285.1 Detailed Description

template<typename SVR>
class L4Re::Util::Vcon_svr< SVR >

Console server template class.

This template uses vcon_write() and vcon_read() to get and deliver data from the implementor.

vcon_read() needs to update the status argument with the L4_vcon_read_stat flags, especially the L4_VCON_↩ READ_STAT_DONE flag to indicate that there's nothing more to read for the other end.

vcon_write() gets the live data from the UTCB. Make sure to copy out the data before using the UTCB again.

The size parameter of both functions is given in bytes.

Definition at line 46 of file vcon_svr.

The documentation for this class was generated from the following file:

- l4/re/util/vcon_svr

15.286 L4Re::Util::Video::Goos_svr Class Reference

Goos server class.

Collaboration diagram for L4Re::Util::Video::Goos_svr:

Public Member Functions

- L4::Cap< L4Re::Dataspace > get_fb () const

Return framebuffer memory dataspace.

- L4Re::Video::Goos::Info const * screen_info () const

  Goos information structure.
- L4Re::Video::View::Info const * view_info () const

  View information structure.
- virtual int refresh (int x, int y, int w, int h)

  Refresh area of the framebuffer.
- void init_infos ()

  Initialize the view information structure of this object.
- virtual ~Goos_svr ()

  Destructor.

Protected Attributes

- L4::Cap< L4Re::Dataspace > _fb_ds

  Goos memory dataspace.
- L4Re::Video::Goos::Info _screen_info

  Goos information.
- L4Re::Video::View::Info _view_info

  View information.

15.286.1 Detailed Description

Goos server class.

Definition at line 36 of file goos_svr.

15.286.2 Member Function Documentation

15.286.2.1 get_fb()

```
L4::Cap<L4Re::Dataspace> L4Re::Util::Video::Goos_svr::get_fb ( ) const  [inline]
```

Return framebuffer memory dataspace.

Returns

Goos memory dataspace

Definition at line 53 of file goos_svr.

References _fb_ds.

15.286.2.2 init_infos()

```
void L4Re::Util::Video::Goos_svr::init_infos ( ) [inline]
```

Initialize the view information structure of this object.

This function initializes the view info structure of this goos object based on the information in the goos information, i.e. the width, height and pixel_info of the goos information has to contain valid values before calling init_info().

Definition at line 89 of file goos_svr.

References _screen_info, _view_info, L4Re::Video::View::Info::buffer_index, L4Re::Video::View::Info::flags, L4Re::Video::View::Info::height, L4Re::Video::Goos::Info::height, L4Re::Video::View::Info::pixel_info, L4Re::Video::Goos::Info::pixel_in L4Re::Video::View::Info::view_index, L4Re::Video::View::Info::width, L4Re::Video::Goos::Info::width, L4Re::Video::View::Info::xpos, and L4Re::Video::View::Info::ypos.

15.286.2.3 refresh()

```
virtual int L4Re::Util::Video::Goos_svr::refresh (
            int x,
            int y,
            int w,
            int h ) [inline], [virtual]
```

Refresh area of the framebuffer.

Parameters

x	X coordinate (pixels)
y	Y coordinate (pixels)
w	Width of area in pixels
h	Height of area in pixels

Returns

0 on success, negative error code otherwise

Definition at line 77 of file goos_svr.

References L4_ENOSYS.

15.286.2.4 screen_info()

```
L4Re::Video::Goos::Info const* L4Re::Util::Video::Goos_svr::screen_info ( ) const [inline]
```

Goos information structure.

Returns

Return goos information structure.

Definition at line 59 of file goos_svr.

References _screen_info.

15.286.2.5 view_info()

```
L4Re::Video::View::Info const* L4Re::Util::Video::Goos_svr::view_info ( ) const  [inline]
```

View information structure.

Returns

Return view information structure.

Definition at line 65 of file goos_svr.

References _view_info.

The documentation for this class was generated from the following file:

- l4/re/util/video/goos_svr

15.287 L4Re::Vfs::Be_file Class Reference

Boiler plate class for implementing an open file for L4Re::Vfs.

Inheritance diagram for L4Re::Vfs::Be_file:

Collaboration diagram for L4Re::Vfs::Be_file:

Public Member Functions

- int unlock_all_locks () throw ()

  Unlock all locks on the file.
- L4::Cap< L4Re::Dataspace > data_space () const throw ()

  Get an L4Re::Dataspace object for the file.
- ssize_t readv (const struct iovec *, int) throw ()

  Default backend for POSIX read and readv functions.
- ssize_t writev (const struct iovec *, int) throw ()

  Default backend for POSIX write and writev functions.
- ssize_t pwritev (const struct iovec *, int, off64_t) throw ()

  Default backend for POSIX pwrite and pwritev functions.
- ssize_t preadv (const struct iovec *, int, off64_t) throw ()

  Default backend for POSIX pread and preadv functions.
- off64_t lseek64 (off64_t, int) throw ()

  Default backend for POSIX seek and lseek functions.
- int ftruncate64 (off64_t) throw ()

  Default backend for the POSIX truncate, ftruncate and similar functions.
- int fsync () const throw ()

  Default backend for POSIX fsync.

- int fdatasync () const throw ()

  Default backend for POSIX fdatasync.
- int ioctl (unsigned long, va_list) throw ()

  Default backend for POSIX ioctl.
- int fstat64 (struct stat64 ∗) const throw ()

  Get status information for the file.
- int fchmod (mode_t) throw ()

  Default backend for POSIX chmod and fchmod.
- int get_status_flags () const throw ()

  Default backend for POSIX fcntl subfunctions.
- int set_status_flags (long) throw ()

  Default backend for POSIX fcntl subfunctions.
- int get_lock (struct flock64 ∗) throw ()

  Default backend for POSIX fcntl subfunctions.
- int set_lock (struct flock64 ∗, bool) throw ()

  Default backend for POSIX fcntl subfunctions.
- int faccessat (const char ∗, int, int) throw ()

  Default backend for POSIX access and faccessat functions.
- int fchmodat (const char ∗, mode_t, int) throw ()

  Default backend for POSIX fchmodat function.
- int utime (const struct utimbuf ∗) throw ()

  Default backend for POSIX utime.
- int utimes (const struct timeval[2]) throw ()

  Default backend for POSIX utimes.
- int utimensat (const char ∗, const struct timespec[2], int) throw ()

  Default backend for POSIX utimensat.
- int mkdir (const char ∗, mode_t) throw ()

  Default backend for POSIX mkdir and mkdirat.
- int unlink (const char ∗) throw ()

  Default backend for POSIX unlink, unlinkat.
- int rename (const char ∗, const char ∗) throw ()

  Default backend for POSIX rename, renameat.
- int link (const char ∗, const char ∗) throw ()

  Default backend for POSIX link, linkat.
- int symlink (const char ∗, const char ∗) throw ()

  Default backend for POSIX symlink, symlinkat.
- int rmdir (const char ∗) throw ()

  Default backend for POSIX rmdir, rmdirat.
- ssize_t readlink (char ∗, size_t)

  Default backend for POSIX readlink, readlinkat.

15.287.1 Detailed Description

Boiler plate class for implementing an open file for L4Re::Vfs.

This class may be used as a base class for everything that a POSIX file descriptor may point to. This are things such as regular files, directories, special device files, streams, pipes, and so on.

Definition at line 40 of file backend.

15.287.2 Member Function Documentation

15.287.2.1 data_space()

```
L4::Cap<L4Re::Dataspace> L4Re::Vfs::Be_file::data_space ( ) const throw ( )   [inline], [virtual]
```

Get an L4Re::Dataspace object for the file.

This is used as a backend for POSIX mmap and mmap2 functions.

Note

mmap is not possible if the function returns an invalid capability.

Returns

A capability to an L4Re::Dataspace, that represents the file contents in an L4Re way.

Implements L4Re::Vfs::Regular_file.

Definition at line 57 of file backend.

15.287.2.2 fstat64()

```
int L4Re::Vfs::Be_file::fstat64 (
            struct stat64 * buf ) const throw ( )   [inline], [virtual]
```

Get status information for the file.

This is the backend for POSIX fstat, stat, fstat64 and friends.

Parameters

out	*buf*	This buffer is filled with the status information.

Returns

0 on success, or <0 on error.

Implements L4Re::Vfs::Generic_file.

Definition at line 96 of file backend.

15.287.2.3 unlock_all_locks()

```
int L4Re::Vfs::Be_file::unlock_all_locks ( ) throw ( )   [inline], [virtual]
```

Unlock all locks on the file.

Note

All locks means all locks independent by which file the locks were taken.

This method is called by the POSIX close implementation to get the POSIX semantics of releasing all locks taken by this application on a close for any fd referencing the real file.

Returns

0 on success, or <0 on error.

Implements L4Re::Vfs::Generic_file.

Definition at line 53 of file backend.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/backend

15.288 L4Re::Vfs::Be_file_system Class Reference

Boilerplate class for implementing a L4Re::Vfs::File_system.

Inheritance diagram for L4Re::Vfs::Be_file_system:

Collaboration diagram for L4Re::Vfs::Be_file_system:

Public Member Functions

- Be_file_system (char const ∗fstype) throw ()

  Create a file-system object for the given fstype.
- ∼Be_file_system () throw ()

  Destroy a file-system object.
- char const ∗ type () const throw ()

  Return the file-system type.

15.288.1 Detailed Description

Boilerplate class for implementing a L4Re::Vfs::File_system.

This class already takes care of registering and unregistering the file system in the global registry and implements the type() method.

Definition at line 308 of file backend.

15.288.2 Constructor & Destructor Documentation

15.288.2.1 Be_file_system()

```
L4Re::Vfs::Be_file_system::Be_file_system (
            char const * fstype ) throw ( )   [inline], [explicit]
```

Create a file-system object for the given *fstype*.

Parameters

fstype	The type that type() shall return.

This constructor takes care of registering the file system in the registry of L4Re::Vfs::vfs_ops.

Definition at line 322 of file backend.

15.288.2.2 ~Be_file_system()

```
L4Re::Vfs::Be_file_system::~Be_file_system ( ) throw ( )   [inline]
```

Destroy a file-system object.

This destructor takes care of removing this file system from the registry of L4Re::Vfs::vfs_ops.

Definition at line 334 of file backend.

15.288.3 Member Function Documentation

15.288.3.1 type()

```
char const* L4Re::Vfs::Be_file_system::type ( ) const throw ( )   [inline], [virtual]
```

Return the file-system type.

Returns the file-system type given as *fstype* in the constructor.

Implements L4Re::Vfs::File_system.

Definition at line 344 of file backend.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/backend

15.289 L4Re::Vfs::Directory Class Reference

Interface for a POSIX file that is a directory.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Directory:

Collaboration diagram for L4Re::Vfs::Directory:

Public Member Functions

- virtual int faccessat (const char ∗path, int mode, int flags)=0 throw ()

  Check access permissions on the given file.
- virtual int mkdir (const char ∗path, mode_t mode)=0 throw ()

  Create a new subdirectory.
- virtual int unlink (const char ∗path)=0 throw ()

  Unlink the given file from that directory.
- virtual int rename (const char ∗src_path, const char ∗dst_path)=0 throw ()

  Rename the given file.
- virtual int link (const char ∗src_path, const char ∗dst_path)=0 throw ()

  Create a hard link (second name) for the given file.
- virtual int symlink (const char ∗src_path, const char ∗dst_path)=0 throw ()

  Create a symbolic link for the given file.
- virtual int rmdir (const char ∗path)=0 throw ()

  Delete an empty directory.

15.289.1 Detailed Description

Interface for a POSIX file that is a directory.

This interface provides functionality for directory files in the L4Re::Vfs. However, real objects use always the combined L4Re::Vfs::File interface.

Definition at line 140 of file vfs.h.

15.289.2 Member Function Documentation

15.289.2.1 faccessat()

```
virtual int L4Re::Vfs::Directory::faccessat (
            const char * path,
            int mode,
            int flags ) throw ( )   [pure virtual]
```

Check access permissions on the given file.

Backend function for POSIX access and faccessat functions.

Parameters

path	The path relative to this directory. Note: *path* is relative to this directory and may contain subdirectories.
mode	The access mode to check.
flags	The flags as in POSIX faccessat (AT_EACCESS, AT_SYMLINK_NOFOLLOW).

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

15.289.2.2 link()

```
virtual int L4Re::Vfs::Directory::link (
            const char * src_path,
            const char * dst_path ) throw ( )   [pure virtual]
```

Create a hard link (second name) for the given file.

Backend for the POSIX link and linkat functions.

Parameters

src_path	The old name to the file. Note: *src_path* is relative to this directory and may contain subdirectories.
dst_path	The new (second) name for the file. Note: *dst_path* is relative to this directory and may contain subdirectories.

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

15.289.2.3 mkdir()

```
virtual int L4Re::Vfs::Directory::mkdir (
            const char * path,
            mode_t mode ) throw ( )   [pure virtual]
```

Create a new subdirectory.

Backend for POSIX mkdir and mkdirat function calls.

Parameters

path	The name of the subdirectory to create. Note: *path* is relative to this directory and may contain subdirectories.
mode	The file mode to use for the new directory.

Returns

0 on success, or <0 on error. -ENOTDIR if this or some component in path is is not a directory.

Implemented in L4Re::Vfs::Be_file.

15.289.2.4 rename()

```
virtual int L4Re::Vfs::Directory::rename (
            const char * src_path,
            const char * dst_path ) throw ( )   [pure virtual]
```

Rename the given file.

Backend for the POSIX rename, renameat functions.

Parameters

src_path	The old name to the file to rename. Note: *src_path* is relative to this directory and may contain subdirectories.
dst_path	The new name for the file. Note: *dst_path* is relative to this directory and may contain subdirectories.

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

15.289.2.5 rmdir()

```
virtual int L4Re::Vfs::Directory::rmdir (
            const char * path ) throw ( )   [pure virtual]
```

Delete an empty directory.

Backend for POSIX rmdir, rmdirat functions.

Parameters

path	The name of the directory to remove. Note: *path* is relative to this directory and may contain subdirectories.

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

15.289.2.6 symlink()

```
virtual int L4Re::Vfs::Directory::symlink (
            const char * src_path,
            const char * dst_path ) throw ( )   [pure virtual]
```

Create a symbolic link for the given file.

Backend for the POSIX symlink and symlinkat functions.

Parameters

src_path	The old name to the file. Note: *src_path* shall be an absolute path.
dst_path	The name for symlink. Note: *dst_path* is relative to this directory and may contain subdirectories.

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

15.289.2.7 unlink()

```
virtual int L4Re::Vfs::Directory::unlink (
            const char * path ) throw ( )   [pure virtual]
```

Unlink the given file from that directory.

Backend for the POSIX unlink and unlinkat functions.

Parameters

path	The name to the file to unlink. Note: *path* is relative to this directory and may contain subdirectories.

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

15.290 L4Re::Vfs::File Class Reference

The basic interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File:

Collaboration diagram for L4Re::Vfs::File:

Additional Inherited Members

15.290.1 Detailed Description

The basic interface for an open POSIX file.

An open POSIX file can be anything that hides behind a POSIX file descriptor. This means that even a directories are files. An open file can be anything from a directory to a special device file so see Generic_file, Regular_file, Directory, and Special_file for more information.

Note

For implementing a backend for the L4Re::Vfs you may use L4Re::Vfs::Be_file as a base class.

Definition at line 435 of file vfs.h.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

15.291 L4Re::Vfs::File_system Class Reference

Basic interface for an L4Re::Vfs file system.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File_system:

Collaboration diagram for L4Re::Vfs::File_system:

Public Member Functions

- virtual char const ∗ type () const =0 throw ()

  Returns the type of the file system, used in mount as fstype argument.
- virtual int mount (char const ∗source, unsigned long mountflags, void const ∗data, cxx::Ref_ptr< File > ∗dir)=0 throw ()

  Create a directory object dir representing source mounted with this file system.

15.291.1 Detailed Description

Basic interface for an L4Re::Vfs file system.

Note

For implementing a special file system you may use L4Re::Vfs::Be_file_system as a base class.

The may purpose of this interface is that there is a single object for each supported file-system type (e.g., ext2, vfat) exists in your application and is registered at the L4Re::Vfs::Fs singleton available in via L4Re::Vfs::vfs_ops. At the end the POSIX mount function call the File_system::mount method for the given file-system type given in mount.

Definition at line 829 of file vfs.h.

15.291.2 Member Function Documentation

15.291.2.1 mount()

```
virtual int L4Re::Vfs::File_system::mount (
            char const * source,
            unsigned long mountflags,
            void const * data,
            cxx::Ref_ptr< File > * dir ) throw ( )   [pure virtual]
```

Create a directory object *dir* representing *source* mounted with this file system.

Parameters

source	The path to the source device to mount. This may also be some URL or anything file-system specific.
mountflags	The mount flags as specified in the POSIX mount call.
data	The data as specified in the POSIX mount call. The contents are file-system specific.

Return values

dir	A new directory object representing the file-system root directory.

Returns

0 on success, and <0 on error (e.g. -EINVAL).

Referenced by L4Re::Vfs::Fs::mount().

Here is the caller graph for this function:

15.291.2.2 type()

```
virtual char const* L4Re::Vfs::File_system::type ( ) const throw ( )   [pure virtual]
```

Returns the type of the file system, used in mount as fstype argument.

Note

This method is already provided by Be_file_system.

Implemented in L4Re::Vfs::Be_file_system.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

15.292 L4Re::Vfs::Fs Class Reference

POSIX File-system related functionality.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Fs:

Collaboration diagram for L4Re::Vfs::Fs:

Public Member Functions

- virtual cxx::Ref_ptr< File > get_file (int fd)=0 throw ()

  Get the L4Re::Vfs::File for the file descriptor fd.
- virtual cxx::Ref_ptr< File > get_root ()=0 throw ()

  Get the directory object for the applications root directory.
- virtual cxx::Ref_ptr< File > get_cwd () throw ()

  Get the directory object for the applications current working directory.
- virtual void set_cwd (cxx::Ref_ptr< File > const &) throw ()

  Set the current working directory for the application.
- virtual int alloc_fd (cxx::Ref_ptr< File > const &f=cxx::Ref_ptr<>::Nil)=0 throw ()

  Allocate the next free file descriptor.
- virtual cxx::Pair< cxx::Ref_ptr< File >, int > set_fd (int fd, cxx::Ref_ptr< File > const &f=cxx::Ref_ptr<>↩ ::Nil)=0 throw ()

  Set the file object referenced by the file descriptor fd.
- virtual cxx::Ref_ptr< File > free_fd (int fd)=0 throw ()

  Free the file descriptor fd.
- virtual int mount (char const *path, cxx::Ref_ptr< File > const &dir)=0 throw ()

  Mount a given file object at the given global path in the VFS.
- int mount (char const *source, char const *target, char const *fstype, unsigned long mountflags, void const *data) throw ()

  Backend for the POSIX mount call.

15.292.1 Detailed Description

POSIX File-system related functionality.

Note

This class usually exists as a singleton as a superclass of L4Re::Vfs::Ops (

See also

L4Re::Vfs::vfs_ops).

Definition at line 881 of file vfs.h.

15.292.2 Member Function Documentation

15.292.2.1 alloc_fd()

```
virtual int L4Re::Vfs::Fs::alloc_fd (
            cxx::Ref_ptr< File > const & f = cxx::Ref_ptr<>::Nil ) throw ( )   [pure virtual]
```

Allocate the next free file descriptor.

Parameters

f	The file to assign to that file descriptor.

Returns

the allocated file descriptor, or -EMFILE on error.

15.292.2.2 free_fd()

```
virtual cxx::Ref_ptr<File> L4Re::Vfs::Fs::free_fd (
            int fd ) throw ( )   [pure virtual]
```

Free the file descriptor *fd*.

Parameters

fd	The file descriptor to free.

Returns

A pointer to the file object that was assigned to the fd.

15.292.2.3 get_file()

```
virtual cxx::Ref_ptr<File> L4Re::Vfs::Fs::get_file (
            int fd ) throw ( )   [pure virtual]
```

Get the L4Re::Vfs::File for the file descriptor *fd*.

Parameters

fd	The POSIX file descriptor number.

Returns

A pointer to the File object, or 0 if *fd* is not open.

15.292.2.4 mount()

```
virtual int L4Re::Vfs::Fs::mount (
            char const * path,
            cxx::Ref_ptr< File > const & dir ) throw ( )   [pure virtual]
```

Mount a given file object at the given global path in the VFS.

Parameters

path	The global path to mount *dir* at.
dir	A pointer to the file/directory object that shall be mounted at *path*.

Returns

0 on success, or <0 on error.

15.292.2.5 set_fd()

```
virtual cxx::Pair<cxx::Ref_ptr<File>, int> L4Re::Vfs::Fs::set_fd (
            int fd,
            cxx::Ref_ptr< File > const & f = cxx::Ref_ptr<>::Nil ) throw ( )   [pure virtual]
```

Set the file object referenced by the file descriptor *fd*.

Parameters

fd	The file descriptor to set to *f*.
f	The file object to assign.

Returns

A pair of a pointer to the file object that was previously assigned to fd (`first`) and a return value (`second`). `second` contains `-#EBADF` if the passed file descriptor is outside the valid range. `first` contains a Nil pointer in that case. On success, `second` contains 0.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

15.293 L4Re::Vfs::Generic_file Class Reference

The common interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Generic_file:

Collaboration diagram for L4Re::Vfs::Generic_file:

Public Member Functions

- virtual int unlock_all_locks ()=0 throw ()

  Unlock all locks on the file.
- virtual int fstat64 (struct stat64 ∗buf) const =0 throw ()

  Get status information for the file.
- virtual int fchmod (mode_t)=0 throw ()

  Change POSIX access rights on that file.
- virtual int get_status_flags () const =0 throw ()

  Get file status flags (fcntl F_GETFL).
- virtual int set_status_flags (long flags)=0 throw ()

  Set file status flags (fcntl F_SETFL).

15.293.1 Detailed Description

The common interface for an open POSIX file.

This interface is common to all kinds of open files, independent of the file type (e.g., directory, regular file etc.). However, in the L4Re::Vfs the interface File is used for every real object.

See also

L4Re::Vfs::File for mor information.

Definition at line 62 of file vfs.h.

15.293.2 Member Function Documentation

15.293.2.1 fchmod()

```
virtual int L4Re::Vfs::Generic_file::fchmod (
            mode_t  ) throw ( )   [pure virtual]
```

Change POSIX access rights on that file.

Backend for POSIX chmod and fchmod.

Implemented in L4Re::Vfs::Be_file.

15.293.2.2 fstat64()

```
virtual int L4Re::Vfs::Generic_file::fstat64 (
            struct stat64 * buf ) const throw ( )   [pure virtual]
```

Get status information for the file.

This is the backend for POSIX fstat, stat, fstat64 and friends.

Parameters

out	*buf*	This buffer is filled with the status information.

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

15.293.2.3 get_status_flags()

```
virtual int L4Re::Vfs::Generic_file::get_status_flags ( ) const throw ( )   [pure virtual]
```

Get file status flags (fcntl F_GETFL).

This function is used by the fcntl implementation for the F_GETFL command.

Returns

flags such as O_RDONLY, O_WRONLY, O_RDWR, O_DIRECT, O_ASYNC, O_NOATIME, O_NONBLOCK, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

15.293.2.4 set_status_flags()

```
virtual int L4Re::Vfs::Generic_file::set_status_flags (
            long flags ) throw ( )   [pure virtual]
```

Set file status flags (fcntl F_SETFL).

This function is used by the fcntl implementation for the F_SETFL command.

Parameters

flags	The file status flags to set. This must be a combination of `O_RDONLY`, `O_WRONLY`, `O_RDWR`, `O_APPEND`, `O_ASYNC`, `O_DIRECT`, `O_NOATIME`, `O_NONBLOCK`.

Note

Creation flags such as `O_CREAT`, `O_EXCL`, `O_NOCTTY`, `O_TRUNC` are ignored.

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

15.293.2.5 unlock_all_locks()

```
virtual int L4Re::Vfs::Generic_file::unlock_all_locks ( ) throw ( )   [pure virtual]
```

Unlock all locks on the file.

Note

All locks means all locks independent by which file the locks were taken.

This method is called by the POSIX close implementation to get the POSIX semantics of releasing all locks taken by this application on a close for any fd referencing the real file.

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

15.294 L4Re::Vfs::Mman Class Reference

Interface for the POSIX memory management.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Mman:

Collaboration diagram for L4Re::Vfs::Mman:

Public Member Functions

- virtual int mmap2 (void ∗start, size_t len, int prot, int flags, int fd, off_t offset, void ∗∗ptr)=0 throw ()

  Backend for the mmap2 system call.
- virtual int munmap (void ∗start, size_t len)=0 throw ()

  Backend for the munmap system call.
- virtual int mremap (void ∗old, size_t old_sz, size_t new_sz, int flags, void ∗∗new_addr)=0 throw ()

  Backend for the mremap system call.
- virtual int mprotect (const void ∗a, size_t sz, int prot)=0 throw ()

  Backend for the mprotect system call.
- virtual int msync (void ∗addr, size_t len, int flags)=0 throw ()

  Backend for the msync system call.
- virtual int madvise (void ∗addr, size_t len, int advice)=0 throw ()

  Backend for the madvice system call.

15.294.1 Detailed Description

Interface for the POSIX memory management.

Note

This interface exists usually as a singleton as superclass of L4Re::Vfs::Ops.

An implementation for this interface is in l4/l4re_vfs/impl/vfs_impl.h and used by the l4re_vfs library or by the VFS implementation in ldso.

Definition at line 745 of file vfs.h.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

15.295 L4Re::Vfs::Ops Class Reference

Interface for the POSIX backends for an application.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Ops:

Collaboration diagram for L4Re::Vfs::Ops:

Additional Inherited Members

15.295.1 Detailed Description

Interface for the POSIX backends for an application.

Note

There usually exists a singe instance of this interface available via L4Re::Vfs::vfs_ops that is used for all kinds of C-Library functions.

Definition at line 1007 of file vfs.h.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

15.296 L4Re::Vfs::Regular_file Class Reference

Interface for a POSIX file that provides regular file semantics.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Regular_file:

Collaboration diagram for L4Re::Vfs::Regular_file:

Public Member Functions

- virtual L4::Cap< L4Re::Dataspace > data_space () const =0 throw ()

  Get an L4Re::Dataspace object for the file.
- virtual ssize_t readv (const struct iovec *, int iovcnt)=0 throw ()

  Read one or more blocks of data from the file.
- virtual ssize_t writev (const struct iovec *, int iovcnt)=0 throw ()

  Write one or more blocks of data to the file.
- virtual off64_t lseek64 (off64_t, int)=0 throw ()

  Change the file pointer.
- virtual int ftruncate64 (off64_t pos)=0 throw ()

  Truncate the file at the given position.
- virtual int fsync () const =0 throw ()

  Sync the data and meta data to persistent storage.
- virtual int fdatasync () const =0 throw ()

  Sync the data to persistent storage.
- virtual int get_lock (struct flock64 *lock)=0 throw ()

  Test if the given lock can be placed in the file.
- virtual int set_lock (struct flock64 *lock, bool wait)=0 throw ()

  Acquire or release the given lock on the file.

15.296.1 Detailed Description

Interface for a POSIX file that provides regular file semantics.

Real objects use always the combined L4Re::Vfs::File interface.

Definition at line 267 of file vfs.h.

15.296.2 Member Function Documentation

15.296.2.1 data_space()

```
virtual L4::Cap<L4Re::Dataspace> L4Re::Vfs::Regular_file::data_space ( ) const throw ( )
[pure virtual]
```

Get an L4Re::Dataspace object for the file.

This is used as a backend for POSIX mmap and mmap2 functions.

Note

mmap is not possible if the function returns an invalid capability.

Returns

A capability to an L4Re::Dataspace, that represents the file contents in an L4Re way.

Implemented in L4Re::Vfs::Be_file.

15.296.2.2 fdatasync()

```
virtual int L4Re::Vfs::Regular_file::fdatasync ( ) const throw ( )   [pure virtual]
```

Sync the data to persistent storage.

This is the backend for POSIX fdatasync.

Implemented in L4Re::Vfs::Be_file.

15.296.2.3 fsync()

```
virtual int L4Re::Vfs::Regular_file::fsync ( ) const throw ( )   [pure virtual]
```

Sync the data and meta data to persistent storage.

This is the backend for POSIX fsync.

Implemented in L4Re::Vfs::Be_file.

15.296.2.4 ftruncate64()

```
virtual int L4Re::Vfs::Regular_file::ftruncate64 (
            off64_t pos ) throw ( )   [pure virtual]
```

Truncate the file at the given position.

This function is the backend for truncate and friends.

Parameters

pos	The offset at which the file shall be truncated.

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

15.296.2.5 get_lock()

```
virtual int L4Re::Vfs::Regular_file::get_lock (
            struct flock64 * lock ) throw ( )   [pure virtual]
```

Test if the given lock can be placed in the file.

This function is used as backend for fcntl F_GETLK commands.

Parameters

lock	The lock that shall be placed on the file. The *l_type* member will contain `F_UNLCK` if the lock could be placed.

Returns

0 on success, <0 on error.

Implemented in L4Re::Vfs::Be_file.

15.296.2.6 lseek64()

```
virtual off64_t L4Re::Vfs::Regular_file::lseek64 (
            off64_t ,
            int  ) throw ( )   [pure virtual]
```

Change the file pointer.

This is the backend for POSIX seek, lseek and friends.

Returns

The new file position, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

15.296.2.7 readv()

```
virtual ssize_t L4Re::Vfs::Regular_file::readv (
            const struct iovec * ,
            int iovcnt ) throw ( )   [pure virtual]
```

Read one or more blocks of data from the file.

This function acts as backend for POSIX read and readv calls and reads data starting for the f_pos pointer of that open file. The file pointer is advanced according to the number of red bytes.

Returns

The number of bytes red from the file. or <0 on error-

Implemented in L4Re::Vfs::Be_file.

15.296.2.8 set_lock()

```
virtual int L4Re::Vfs::Regular_file::set_lock (
            struct flock64 * lock,
            bool wait ) throw ( )   [pure virtual]
```

Acquire or release the given lock on the file.

This function is used as backend for fcntl F_SETLK and F_SETLKW commands.

Parameters

lock	The lock that shall be placed on the file.
wait	If true, then block if there is a conflicting lock on the file.

Returns

0 on success, <0 on error.

Implemented in L4Re::Vfs::Be_file.

15.296.2.9 writev()

```
virtual ssize_t L4Re::Vfs::Regular_file::writev (
            const struct iovec * ,
            int iovcnt ) throw ( )   [pure virtual]
```

Write one or more blocks of data to the file.

This function acts as backend for POSIX write and writev calls. The data is written starting at the current file pointer and the file pointer must be advanced according to the number of written bytes.

Returns

The number of bytes written to the file, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

15.297 L4Re::Vfs::Special_file Class Reference

Interface for a POSIX file that provides special file semantics.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Special_file:

Collaboration diagram for L4Re::Vfs::Special_file:

Public Member Functions

- virtual int ioctl (unsigned long cmd, va_list args)=0 throw ()

  The famous IO control.

15.297.1 Detailed Description

Interface for a POSIX file that provides special file semantics.

Real objects use always the combined L4Re::Vfs::File interface.

Definition at line 400 of file vfs.h.

15.297.2 Member Function Documentation

15.297.2.1 ioctl()

```
virtual int L4Re::Vfs::Special_file::ioctl (
            unsigned long cmd,
            va_list args ) throw ( )   [pure virtual]
```

The famous IO control.

Backend for POSIX generic object invocation ioctl.

Parameters

cmd	The ioctl command.
args	The arguments for the ioctl, usually some kind of pointer.

Returns

>=0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

15.298 L4Re::Video::Color_component Class Reference

A color component.

Collaboration diagram for L4Re::Video::Color_component:

Public Member Functions

- Color_component ()

  Constructor.
- Color_component (unsigned char bits, unsigned char shift)

  Constructor.
- unsigned char size () const

  Return the number of bits used by the component.
- unsigned char shift () const

  Return the position of the component in the pixel.
- bool operator== (Color_component const &o) const

  Compare for equality.
- int get (unsigned long v) const

  Get component from value (normalized to 16bits).
- long unsigned set (int v) const

  Transform 16bit normalized value to the component in the color space.
- template<typename OUT >
  void dump (OUT &s) const

  Dump information on the view information to a stream.

15.298.1 Detailed Description

A color component.

Definition at line 32 of file colors.

15.298.2 Constructor & Destructor Documentation

15.298.2.1 Color_component()

```
L4Re::Video::Color_component::Color_component (
            unsigned char bits,
            unsigned char shift )  [inline]
```

Constructor.

Parameters

bits	Number of bits used by the component
shift	Position in bits of the component in the pixel

Definition at line 47 of file colors.

15.298.3 Member Function Documentation

15.298.3.1 dump()

```
template<typename OUT >
void L4Re::Video::Color_component::dump (
            OUT & s ) const  [inline]
```

Dump information on the view information to a stream.

Parameters

s	Stream

Returns

The stream

Definition at line 94 of file colors.

15.298.3.2 get()

```
int L4Re::Video::Color_component::get (
            unsigned long v ) const  [inline]
```

Get component from value (normalized to 16bits).

Parameters

v	Value

Returns

Converted value

Definition at line 74 of file colors.

15.298.3.3 operator==()

```
bool L4Re::Video::Color_component::operator== (
            Color_component const & o ) const  [inline]
```

Compare for equality.

Returns

True if the same components are described, false if not.

Definition at line 66 of file colors.

15.298.3.4 set()

```
long unsigned L4Re::Video::Color_component::set (
            int v ) const  [inline]
```

Transform 16bit normalized value to the component in the color space.

Parameters

v	Value return Converted value.

Definition at line 85 of file colors.

15.298.3.5 shift()

```
unsigned char L4Re::Video::Color_component::shift ( ) const  [inline]
```

Return the position of the component in the pixel.

Returns

Position in bits of the component in the pixel

Definition at line 60 of file colors.

Referenced by L4Re::Video::Pixel_info::padding().

Here is the caller graph for this function:

15.298.3.6 size()

```
unsigned char L4Re::Video::Color_component::size ( ) const  [inline]
```

Return the number of bits used by the component.

Returns

Number of bits used by the component

Definition at line 54 of file colors.

Referenced by L4Re::Video::Pixel_info::bits_per_pixel(), L4Re::Video::Pixel_info::has_alpha(), and L4Re::Video::Pixel_info::padding()

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/re/video/colors

15.299 L4Re::Video::Goos Class Reference

Class that abstracts framebuffers.

Inheritance diagram for L4Re::Video::Goos:

Collaboration diagram for L4Re::Video::Goos:

Data Structures

- struct Info

  Information structure of a Goos.

Public Types

- enum Flags { F_auto_refresh = 0x01 , F_pointer = 0x02 , F_dynamic_views = 0x04 , F_dynamic_buffers = 0x08 }

  Flags for a Goos.

Public Member Functions

- long info (Info *info)

  Return the Goos information of the Goos.
- long get_static_buffer (unsigned idx, L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf)

  Return a static buffer of a Goos.
- long create_buffer (unsigned long size, L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf)

  Create a buffer.
- long delete_buffer (unsigned idx)

  Delete a buffer.
- int create_view (View *view, l4_utcb_t *utcb=l4_utcb()) const noexcept

  Create a view.
- int delete_view (View const &v, l4_utcb_t *utcb=l4_utcb()) const noexcept

  Delete a view.
- View view (unsigned index) const noexcept

  Return a view.
- long refresh (int x, int y, int w, int h)

  Trigger refreshing of the given area on the virtual screen.

Additional Inherited Members

15.299.1 Detailed Description

Class that abstracts framebuffers.

A framebuffer is the pixel data that is displayed on a screen and a Goos object lets the user manipulate that data. A Goos makes use of two kinds of objects:

- Buffers in the form of L4Re::Dataspace objects. These hold the bytes for the pixel data.
- L4Re::Video::View objects.

Both can either be static, that is their number and configuration is fixed and determined by the framebuffer, or they can be dynamic, with the user allocating them.

Definition at line 226 of file goos.

15.299.2 Member Enumeration Documentation

15.299.2.1 Flags

`enum L4Re::Video::Goos::Flags`

Flags for a Goos.

Enumerator

F_auto_refresh	The graphics display is automatically refreshed.
F_pointer	We have a mouse pointer.
F_dynamic_views	Supports dynamically allocated views.
F_dynamic_buffers	Supports dynamically allocated buffers.

Definition at line 231 of file goos.

15.299.3 Member Function Documentation

15.299.3.1 create_buffer()

```
long L4Re::Video::Goos::create_buffer (
            unsigned long size,
            L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf )
```

Create a buffer.

Parameters

size	Size of buffer in bytes.
rbuf	Capability slot to point the buffer dataspace to.

Return values

>=0	Success, the value returned is the buffer index.
<0	Error

15.299.3.2 create_view()

```
int L4Re::Video::Goos::create_view (
            View * view,
            l4_utcb_t * utcb = l4_utcb() ) const  [inline], [noexcept]
```

Create a view.

Parameters

out	*view*	A view object.
	utcb	UTCB of the caller. This is a default parameter.

Return values

>=0	Success, the value returned is the view index.
<0	Error

Definition at line 315 of file goos.

15.299.3.3 delete_buffer()

```
long L4Re::Video::Goos::delete_buffer (
            unsigned idx )
```

Delete a buffer.

Parameters

idx	Buffer to delete.

Return values

0	Success
<0	Error

15.299.3.4 delete_view()

```
int L4Re::Video::Goos::delete_view (
            View const & v,
            l4_utcb_t * utcb = l4_utcb() ) const  [inline], [noexcept]
```

Delete a view.

Parameters

v	The view object to delete.
utcb	UTCB of the caller. This is a default parameter.

Return values

0	Success
<0	Error

Definition at line 335 of file goos.

15.299.3.5 get_static_buffer()

```
long L4Re::Video::Goos::get_static_buffer (
            unsigned idx,
            L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf )
```

Return a static buffer of a Goos.

Parameters

idx	Index of the static buffer.
rbuf	Capability slot to point the buffer dataspace to.

Return values

0	Success
<0	Error

15.299.3.6 info()

```
long L4Re::Video::Goos::info (
            Info * info )
```

Return the Goos information of the Goos.

Parameters

out	*info*	Goos information structure pointer.

Return values

0	Success
<0	Error

15.299.3.7 view()

```
View L4Re::Video::Goos::view (
            unsigned index ) const  [inline], [noexcept]
```

Return a view.

Parameters

index	Index of the view to return.

Returns

The view.

Definition at line 366 of file goos.

The documentation for this class was generated from the following file:

- l4/re/video/goos

15.300 L4Re::Video::Goos::Info Struct Reference

Information structure of a Goos.

Collaboration diagram for L4Re::Video::Goos::Info:

Public Member Functions

- bool auto_refresh () const

  Return whether this Goos does auto refreshing or the view refresh functions must be used to make changes visible.
- bool has_pointer () const

  Return whether a pointer is used by the provider of the Goos.
- bool has_dynamic_views () const

  Return whether dynamic view are supported.
- bool has_dynamic_buffers () const

  Return whether dynamic buffers are supported.

Data Fields

- unsigned long width

  Width.
- unsigned long height

  Height.
- unsigned flags

  Flags, see Flags.
- unsigned num_static_views

  Number of static view.
- unsigned num_static_buffers

  Number of static buffers.
- Pixel_info pixel_info

  Pixel information.

15.300.1 Detailed Description

Information structure of a Goos.

Definition at line 240 of file goos.

The documentation for this struct was generated from the following file:

- l4/re/video/goos

15.301 L4Re::Video::Pixel_info Class Reference

Pixel information.

Collaboration diagram for L4Re::Video::Pixel_info:

Public Member Functions

- Color_component const & r () const

  Return the red color compoment of the pixel.
- Color_component const & g () const

  Return the green color compoment of the pixel.
- Color_component const & b () const

  Return the blue color compoment of the pixel.
- Color_component const & a () const

  Return the alpha color compoment of the pixel.
- Color_component const padding () const

  Compute the padding pseudo component.
- unsigned char bytes_per_pixel () const

  Query size of pixel in bytes.
- unsigned char bits_per_pixel () const

  Number of bits of the pixel.
- bool has_alpha () const

  Return whether the pixel has an alpha channel.
- void r (Color_component const &c)

  Set the red color component of the pixel.
- void g (Color_component const &c)

  Set the green color component of the pixel.
- void b (Color_component const &c)

  Set the blue color component of the pixel.
- void a (Color_component const &c)

  Set the alpha color component of the pixel.
- void bytes_per_pixel (unsigned char bpp)

  Set the size of the pixel in bytes.

- Pixel_info ()=default

  Constructor.
- Pixel_info (unsigned char bpp, char r, char rs, char g, char gs, char b, char bs, char a=0, char as=0)

  Constructor.
- template<typename VBI >
  Pixel_info (VBI const *vbi)

  Convenience constructor.
- bool operator== (Pixel_info const &o) const

  Compare for complete equality of the color space.
- template<typename OUT >
  void dump (OUT &s) const

  Dump information on the pixel to a stream.

15.301.1 Detailed Description

Pixel information.

This class wraps the information on a pixel, such as the size and position of each color component in the pixel.

Definition at line 107 of file colors.

15.301.2 Constructor & Destructor Documentation

15.301.2.1 Pixel_info() [1/2]

```
L4Re::Video::Pixel_info::Pixel_info (
            unsigned char bpp,
            char r,
            char rs,
            char g,
            char gs,
            char b,
            char bs,
            char a = 0,
            char as = 0 )  [inline]
```

Constructor.

Parameters

bpp	Size of pixel in bytes.
r	Red component size.
rs	Red component shift.
g	Green component size.
gs	Green component shift.
b	Blue component size.
bs	Blue component shift.
a	Alpha component size, defaults to 0.
as	Alpha component shift, defaults to 0.

Definition at line 225 of file colors.

15.301.2.2 Pixel_info() [2/2]

```
template<typename VBI >
L4Re::Video::Pixel_info::Pixel_info (
            VBI const * vbi )  [inline], [explicit]
```

Convenience constructor.

Parameters

vbi	Suitable information structure. Convenience constructor to create the pixel info from a VESA Framebuffer Info.

Definition at line 237 of file colors.

15.301.3 Member Function Documentation

15.301.3.1 a() [1/2]

```
Color_component const& L4Re::Video::Pixel_info::a ( ) const  [inline]
```

Return the alpha color compoment of the pixel.

Returns

Alpha color component.

Definition at line 136 of file colors.

15.301.3.2 a() [2/2]

```
void L4Re::Video::Pixel_info::a (
            Color_component const & c )  [inline]
```

Set the alpha color component of the pixel.

Parameters

c	Alpha color component.

Definition at line 200 of file colors.

15.301.3.3 b() [1/2]

```
Color_component const& L4Re::Video::Pixel_info::b ( ) const  [inline]
```

Return the blue color compoment of the pixel.

Returns

Blue color component.

Definition at line 130 of file colors.

15.301.3.4 b() [2/2]

```
void L4Re::Video::Pixel_info::b (
            Color_component const & c )  [inline]
```

Set the blue color component of the pixel.

Parameters

c	Blue color component.

Definition at line 194 of file colors.

15.301.3.5 bits_per_pixel()

```
unsigned char L4Re::Video::Pixel_info::bits_per_pixel ( ) const  [inline]
```

Number of bits of the pixel.

Returns

Number of bits used by the pixel.

Definition at line 169 of file colors.

References L4Re::Video::Color_component::size().

Here is the call graph for this function:

15.301.3.6 bytes_per_pixel() [1/2]

```
unsigned char L4Re::Video::Pixel_info::bytes_per_pixel ( ) const  [inline]
```

Query size of pixel in bytes.

Returns

Size of pixel in bytes.

Definition at line 163 of file colors.

15.301.3.7 bytes_per_pixel() [2/2]

```
void L4Re::Video::Pixel_info::bytes_per_pixel (
            unsigned char bpp )  [inline]
```

Set the size of the pixel in bytes.

Parameters

bpp	Size of pixel in bytes.

Definition at line 206 of file colors.

15.301.3.8 dump()

```
template<typename OUT >
void L4Re::Video::Pixel_info::dump (
            OUT & s ) const  [inline]
```

Dump information on the pixel to a stream.

Parameters

s	Stream

Returns

The stream

Definition at line 260 of file colors.

Referenced by L4Re::Video::View::Info::dump().

Here is the caller graph for this function:

15.301.3.9 g() [1/2]

```
Color_component const& L4Re::Video::Pixel_info::g ( ) const  [inline]
```

Return the green color compoment of the pixel.

Returns

Green color component.

Definition at line 124 of file colors.

15.301.3.10 g() [2/2]

```
void L4Re::Video::Pixel_info::g (
            Color_component const & c )  [inline]
```

Set the green color component of the pixel.

Parameters

c	Green color component.

Definition at line 188 of file colors.

15.301.3.11 has_alpha()

```
bool L4Re::Video::Pixel_info::has_alpha ( ) const  [inline]
```

Return whether the pixel has an alpha channel.

Returns

True if the pixel has an alpha channel, false if not.

Definition at line 176 of file colors.

References L4Re::Video::Color_component::size().

Here is the call graph for this function:

15.301.3.12 operator==()

```
bool L4Re::Video::Pixel_info::operator== (
            Pixel_info const & o ) const  [inline]
```

Compare for complete equality of the color space.

Parameters

o	A Pixel_info to compare to.

Returns

true if the both Pixel_info's are equal, false if not.

Definition at line 249 of file colors.

15.301.3.13 padding()

```
Color_component const L4Re::Video::Pixel_info::padding ( ) const  [inline]
```

Compute the padding pseudo component.

The padding pseudo component represents the tailing bits that are reserved in RGB32 and similar pixel formats.

Returns

Padding pseudo component.

Definition at line 144 of file colors.

References L4Re::Video::Color_component::shift(), and L4Re::Video::Color_component::size().

Here is the call graph for this function:

15.301.3.14 r() [1/2]

```
Color_component const& L4Re::Video::Pixel_info::r ( ) const  [inline]
```

Return the red color compoment of the pixel.

Returns

Red color component.

Definition at line 118 of file colors.

15.301.3.15 r() [2/2]

```
void L4Re::Video::Pixel_info::r (
            Color_component const & c )  [inline]
```

Set the red color component of the pixel.

Parameters

c	Red color component.

Definition at line 182 of file colors.

The documentation for this class was generated from the following file:

- l4/re/video/colors

15.302 L4Re::Video::View Class Reference

View of a framebuffer.

Collaboration diagram for L4Re::Video::View:

Data Structures

- struct Info

  Information structure of a view.

Public Types

- enum Flags {
  F_none = 0x00 , F_set_buffer = 0x01 , F_set_buffer_offset = 0x02 , F_set_bytes_per_line = 0x04 , F_set_pixel = 0x08 , F_set_position = 0x10 , F_dyn_allocated = 0x20 , F_set_background = 0x40 , F_set_flags = 0x80 , F_fully_dynamic }

  Flags on a view.
- enum V_flags { F_above = 0x1000 , F_flags_mask = 0xff000 }

  Property flags of a view.

Public Member Functions

- int info (Info *info) const noexcept

  Return the view information of the view.
- int set_info (Info const &info) const noexcept

  Set the information structure for this view.
- int set_viewport (int scr_x, int scr_y, int w, int h, unsigned long buf_offset) const noexcept

  Set the position of the view in the Goos.
- int stack (View const &pivot, bool behind=true) const noexcept

  Move this view in the view stack.
- int push_top () const noexcept

  Make this view the top-most view.
- int push_bottom () const noexcept

  Push this view the back.
- int refresh (int x, int y, int w, int h) const noexcept

  Refresh/Redraw the view.
- bool valid () const

  Return whether this view is valid.

15.302.1 Detailed Description

View of a framebuffer.

A view is a rectangular subset of a framebuffer managed by a Goos object. The Goos orders multiple views in a stack which determines which view is on top in case they overlap. The view's pixel data is provided by a backing buffer, which must belong to the Goos. It can be static or dynamically allocated, depending on the framebuffer.

See also

L4Re::Video::Goos

Definition at line 42 of file goos.

15.302.2 Member Enumeration Documentation

15.302.2.1 Flags

`enum L4Re::Video::View::Flags`

Flags on a view.

Enumerator

F_none	everything for this view is static (the VESA-FB case)
F_set_buffer	buffer object for this view can be changed
F_set_buffer_offset	buffer offset can be set
F_set_bytes_per_line	bytes per line can be set

Enumerator

F_set_pixel	pixel type can be set
F_set_position	position on screen can be set
F_dyn_allocated	View is dynamically allocated.
F_set_background	Set view as background for session.
F_set_flags	Set view flags (. **See also** V_flags)
F_fully_dynamic	Flags for a fully dynamic view.

Definition at line 62 of file goos.

15.302.2.2 V_flags

```
enum L4Re::Video::View::V_flags
```

Property flags of a view.

Such flags can be set or deleted with the F_set_flags operation using the `set_info()` method.

Enumerator

F_above	Flag the view as stay on top.
F_flags_mask	Mask containing all possible property flags.

Definition at line 85 of file goos.

15.302.3 Member Function Documentation

15.302.3.1 info()

```
int L4Re::Video::View::info (
            Info * info ) const  [inline], [noexcept]
```

Return the view information of the view.

Parameters

out	*info*	Information structure pointer.

Return values

0	Success
<0	Error

Definition at line 370 of file goos.

15.302.3.2 refresh()

```
int L4Re::Video::View::refresh (
            int x,
            int y,
            int w,
            int h ) const  [inline], [noexcept]
```

Refresh/Redraw the view.

Parameters

x	X position.
y	Y position.
w	Width.
h	Height.

Return values

0	Success
<0	Error

Definition at line 382 of file goos.

15.302.3.3 set_info()

```
int L4Re::Video::View::set_info (
            Info const & info ) const  [inline], [noexcept]
```

Set the information structure for this view.

Parameters

info	Information structure.

Return values

0	Success

Return values

<0	Error

The function will also set the view port according to the values given in the information structure.

Definition at line 374 of file goos.

15.302.3.4 set_viewport()

```
int L4Re::Video::View::set_viewport (
            int scr_x,
            int scr_y,
            int w,
            int h,
            unsigned long buf_offset ) const  [inline], [noexcept]
```

Set the position of the view in the Goos.

Parameters

scr_x	X position
scr_y	Y position
w	Width
h	Height
buf_offset	Offset in the buffer in bytes

Return values

0	Success
<0	Error

Definition at line 386 of file goos.

References L4Re::Video::View::Info::buffer_offset, L4Re::Video::View::Info::flags, L4Re::Video::View::Info::height, L4Re::Video::View::Info::width, L4Re::Video::View::Info::xpos, and L4Re::Video::View::Info::ypos.

15.302.3.5 stack()

```
int L4Re::Video::View::stack (
            View const & pivot,
            bool behind = true ) const  [inline], [noexcept]
```

Move this view in the view stack.

Parameters

pivot	View to move relative to
behind	When true move the view behind the pivot view, if false move the view before the pivot view.

Return values

0	Success
<0	Error

Definition at line 378 of file goos.

The documentation for this class was generated from the following file:

- l4/re/video/goos

15.303 L4Re::Video::View::Info Struct Reference

Information structure of a view.

Collaboration diagram for L4Re::Video::View::Info:

Public Member Functions

- bool has_static_buffer () const

  Return whether the view has a static buffer.
- bool has_static_buffer_offset () const

  Return whether the static buffer offset is available.
- bool has_set_buffer () const

  Return whether a buffer is set.
- bool has_set_buffer_offset () const

Return whether the given buffer offset is valid.

- bool has_set_bytes_per_line () const

  Return whether the given bytes-per-line value is valid.
- bool has_set_pixel () const

  Return whether the given pixel information is valid.
- bool has_set_position () const

  Return whether the position information given is valid.
- template<typename OUT >
  void dump (OUT &s) const

  Dump information on the view information to a stream.

Data Fields

- unsigned flags

  Flags, see Flags and V_flags.
- unsigned view_index

  Index of the view.
- unsigned long xpos

  X position in pixels of the view in the Goos.
- unsigned long ypos

  Y position in pixels of the view in the Goos.
- unsigned long width

  Width of the view in pixels.
- unsigned long height

  Height of the view in pixels.
- unsigned long buffer_offset

  Offset in the memory buffer in bytes.
- unsigned long bytes_per_line

  Bytes per line.
- Pixel_info pixel_info

  Pixel information.
- unsigned buffer_index

  Number of the buffer used for this view.

15.303.1 Detailed Description

Information structure of a view.

Definition at line 94 of file goos.

The documentation for this struct was generated from the following file:

- l4/re/video/goos

15.304 l4re_aux_t Struct Reference

Auxiliary descriptor.

```
#include <l4aux.h>
```

Collaboration diagram for l4re_aux_t:

Data Fields

- char const * binary

  Binary name.
- l4_cap_idx_t kip_ds

  Data space of the KIP.
- l4_umword_t dbg_lvl

  Debug levels for l4re.
- l4_umword_t ldr_flags

  Flags for l4re, see l4re_aux_ldr_flags_t.

15.304.1 Detailed Description

Auxiliary descriptor.

Definition at line 51 of file l4aux.h.

The documentation for this struct was generated from the following file:

- l4/re/l4aux.h

15.305 l4re_ds_stats_t Struct Reference

Information about the data space.

```
#include <dataspace.h>
```

Collaboration diagram for l4re_ds_stats_t:

Data Fields

- l4re_ds_size_t size

  size
- l4re_ds_flags_t flags

  flags

15.305.1 Detailed Description

Information about the data space.

Definition at line 49 of file dataspace.h.

The documentation for this struct was generated from the following file:

- l4/re/c/dataspace.h

15.306 l4re_elf_aux_mword_t Struct Reference

Auxiliary vector element for a single unsigned data word.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re_elf_aux_mword_t:

15.306.1 Detailed Description

Auxiliary vector element for a single unsigned data word.

Definition at line 124 of file elf_aux.h.

The documentation for this struct was generated from the following file:

- l4/re/elf_aux.h

15.307 l4re_elf_aux_t Struct Reference

Generic header for each auxiliary vector element.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re_elf_aux_t:

15.307.1 Detailed Description

Generic header for each auxiliary vector element.

Definition at line 104 of file elf_aux.h.

The documentation for this struct was generated from the following file:

- l4/re/elf_aux.h

15.308 l4re_elf_aux_vma_t Struct Reference

Auxiliary vector element for a reserved virtual memory area.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re_elf_aux_vma_t:

15.308.1 Detailed Description

Auxiliary vector element for a reserved virtual memory area.

Definition at line 113 of file elf_aux.h.

The documentation for this struct was generated from the following file:

- l4/re/elf_aux.h

15.309 l4re_env_cap_entry_t Struct Reference

Entry in the L4Re environment array for the named inital objects.

```
#include <env.h>
```

Collaboration diagram for l4re_env_cap_entry_t:

Public Member Functions

- l4re_env_cap_entry_t () L4_NOTHROW

  Create an invalid entry.
- l4re_env_cap_entry_t (char const ∗n, l4_cap_idx_t c, l4_umword_t f=0) L4_NOTHROW

  Create an entry with the name n, capability c, and flags f.

Data Fields

- l4_cap_idx_t cap

  The capability selector for the object.
- l4_umword_t flags

  Some flags for the object.
- char name [16]

  The name of the object.

15.309.1 Detailed Description

Entry in the L4Re environment array for the named inital objects.

Definition at line 50 of file env.h.

15.309.2 Constructor & Destructor Documentation

15.309.2.1 l4re_env_cap_entry_t()

```
l4re_env_cap_entry_t::l4re_env_cap_entry_t (
            char const * n,
            l4_cap_idx_t c,
            l4_umword_t f = 0 )  [inline]
```

Create an entry with the name *n*, capability *c*, and flags *f*.

Parameters

n	is the name of the initial object.
c	is the capability selector that refers the initial object.
f	are the additional flags for the object.

Definition at line 81 of file env.h.

References name.

15.309.3 Field Documentation

15.309.3.1 flags

```
l4_umword_t l4re_env_cap_entry_t::flags
```

Some flags for the object.

Note

Currently unused.

Definition at line 61 of file env.h.

Referenced by l4re_env_get_cap_l().

The documentation for this struct was generated from the following file:

- l4/re/env.h

15.310 l4re_env_t Struct Reference

Initial environment data structure.

```
#include <env.h>
```

Collaboration diagram for l4re_env_t:

Data Fields

- l4_cap_idx_t parent

  Parent object-capability.
- l4_cap_idx_t rm

  Region map object-capability.
- l4_cap_idx_t mem_alloc

  Memory allocator object-capability.
- l4_cap_idx_t log

  Logging object-capability.
- l4_cap_idx_t main_thread

  Object-capability of the first user thread.
- l4_cap_idx_t factory

Object-capability of the factory available to the task.

- l4_cap_idx_t scheduler

  Object capability for the scheduler set to use.

- l4_cap_idx_t first_free_cap

  First capability index available to the application.

- l4_fpage_t utcb_area

  UTCB area of the task.

- l4_addr_t first_free_utcb

  First UTCB within the UTCB area available to the application.

15.310.1 Detailed Description

Initial environment data structure.

See also

Initial environment

Definition at line 109 of file env.h.

The documentation for this struct was generated from the following file:

- l4/re/env.h

15.311 l4re_event_t Struct Reference

Event structure used in buffer.

```
#include <event.h>
```

Collaboration diagram for l4re_event_t:

Data Fields

- long long time

  Time stamp of the event.
- unsigned short type

  Type of the event.
- unsigned short code

  Code of the event.
- int value

  Value of the event.
- l4_umword_t stream_id

  Stream ID.

15.311.1 Detailed Description

Event structure used in buffer.

Definition at line 40 of file event.h.

The documentation for this struct was generated from the following file:

- l4/re/c/event.h

15.312 l4re_video_color_component_t Struct Reference

Color component structure.

```
#include <colors.h>
```

Collaboration diagram for l4re_video_color_component_t:

Data Fields

- unsigned char size

  Size in bits.
- unsigned char shift

  offset in pixel

15.312.1 Detailed Description

Color component structure.

Definition at line 31 of file colors.h.

The documentation for this struct was generated from the following file:

- l4/re/c/video/colors.h

15.313 l4re_video_goos_info_t Struct Reference

Goos information structure.

```
#include <goos.h>
```

Collaboration diagram for l4re_video_goos_info_t:

Data Fields

- unsigned long width

  Width of the goos.
- unsigned long height

  Height of the goos.
- unsigned flags

  Flags of the framebuffer, see l4re_video_goos_info_flags_t.
- unsigned num_static_views

  Number of static views.
- unsigned num_static_buffers

  Number of static buffers.
- l4re_video_pixel_info_t pixel_info

  Pixel layout of the goos.

15.313.1 Detailed Description

Goos information structure.

Definition at line 51 of file goos.h.

The documentation for this struct was generated from the following file:

- l4/re/c/video/goos.h

15.314 l4re_video_pixel_info_t Struct Reference

Pixel_info structure.

```
#include <colors.h>
```

Collaboration diagram for l4re_video_pixel_info_t:

Data Fields

- l4re_video_color_component_t a

  Colors.
- unsigned char bytes_per_pixel

  Bytes per pixel.

15.314.1 Detailed Description

Pixel_info structure.

Definition at line 41 of file colors.h.

The documentation for this struct was generated from the following file:

- l4/re/c/video/colors.h

15.315 l4re_video_view_info_t Struct Reference

View information structure.

```
#include <view.h>
```

Collaboration diagram for l4re_video_view_info_t:

Data Fields

- unsigned flags

  Flags.
- unsigned view_index

  Number of view in the goos.
- unsigned long height

  Position in goos and size of view.
- unsigned long buffer_offset

  Memory offset in goos buffer.
- unsigned long bytes_per_line

  Size of line in view.

- l4re_video_pixel_info_t pixel_info

  Pixel info.
- unsigned buffer_index

  Number of buffer of goos.

15.315.1 Detailed Description

View information structure.

Definition at line 59 of file view.h.

The documentation for this struct was generated from the following file:

- l4/re/c/video/view.h

15.316 l4re_video_view_t Struct Reference

C representation of a goos view.

```
#include <view.h>
```

Collaboration diagram for l4re_video_view_t:

15.316.1 Detailed Description

C representation of a goos view.

A view is a visible rectangle that provides a view to the contents of a buffer (frame buffer) memory object and is placed on a real screen.

Definition at line 78 of file view.h.

The documentation for this struct was generated from the following file:

- l4/re/c/video/view.h

15.317 l4util_idt_desc_t Struct Reference

IDT entry.

```
#include <idt.h>
```

Collaboration diagram for l4util_idt_desc_t:

Data Fields

- l4_uint64_t b

  see Intel doc
- l4_uint32_t b

  see Intel doc

15.317.1 Detailed Description

IDT entry.

Definition at line 33 of file idt.h.

The documentation for this struct was generated from the following file:

- amd64/l4/util/idt.h

15.318 l4util_idt_header_t Struct Reference

Header of an IDT table.

```
#include <idt.h>
```

Collaboration diagram for l4util_idt_header_t:

Data Fields

- l4_uint16_t limit

  limit field (see Intel doc)
- void * base

  idt base (see Intel doc)

15.318.1 Detailed Description

Header of an IDT table.

Definition at line 40 of file idt.h.

The documentation for this struct was generated from the following file:

- amd64/l4/util/idt.h

15.319 l4util_l4mod_info Struct Reference

Base module structure.

```
#include <l4mod.h>
```

Collaboration diagram for l4util_l4mod_info:

Data Fields

- l4_uint64_t flags

  Flags.
- l4_uint64_t cmdline

  Pointer to kernel command line.
- l4_uint64_t mods_addr

  Module list.
- l4_uint32_t mods_count

  Number of modules.
- l4_uint64_t vbe_ctrl_info

  VESA video info, valid if one of vbe_ctrl_info or vbe_mode_info is not zero.
- l4_uint64_t vbe_mode_info

  VESA video mode info.

15.319.1 Detailed Description

Base module structure.

Definition at line 30 of file l4mod.h.

15.319.2 Field Documentation

15.319.2.1 vbe_ctrl_info

```
l4_uint64_t l4util_l4mod_info::vbe_ctrl_info
```

VESA video info, valid if one of vbe_ctrl_info or vbe_mode_info is not zero.

VESA video controller info

Definition at line 42 of file l4mod.h.

The documentation for this struct was generated from the following file:

- l4/util/l4mod.h

15.320 l4util_l4mod_mod Struct Reference

A single module.

```
#include <l4mod.h>
```

Collaboration diagram for l4util_l4mod_mod:

Data Fields

- l4_uint64_t flags

  Module flags (`l4util_l4mod_mod_info_flag`)
- l4_uint64_t mod_start

  Starting address of module in memory.
- l4_uint64_t mod_end

  End address of module in memory.
- l4_uint64_t cmdline

  Module command line.

15.320.1 Detailed Description

A single module.

Definition at line 21 of file l4mod.h.

The documentation for this struct was generated from the following file:

- l4/util/l4mod.h

15.321 l4util_mb_addr_range_t Struct Reference

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_addr_range_t:

Data Fields

- l4_uint32_t struct_size

  Size of structure.
- l4_uint64_t addr

  Start address.
- l4_uint64_t size

  Size of memory range.
- l4_uint32_t type

  type of memory range

15.321.1 Detailed Description

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

Definition at line 50 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

15.322 l4util_mb_apm_t Struct Reference

APM BIOS info.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_apm_t:

15.322.1 Detailed Description

APM BIOS info.

Definition at line 92 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

15.323 l4util_mb_drive_t Struct Reference

Drive Info structure.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_drive_t:

Data Fields

- l4_uint8_t drive_number

  <The size of this structure.
- l4_uint8_t drive_mode

  <The BIOS drive number.
- l4_uint16_t drive_cylinders

  <The access mode (see below).
- l4_uint8_t drive_heads

  <number of cylinders
- l4_uint8_t drive_sectors

  <number of heads
- l4_uint16_t drive_ports [0]

  <number of sectors per track

15.323.1 Detailed Description

Drive Info structure.

Definition at line 75 of file mb_info.h.

15.323.2 Field Documentation

15.323.2.1 drive_cylinders

```
l4_uint16_t l4util_mb_drive_t::drive_cylinders
```

<The access mode (see below).

Definition at line 80 of file mb_info.h.

15.323.2.2 drive_mode

```
l4_uint8_t l4util_mb_drive_t::drive_mode
```

<The BIOS drive number.

Definition at line 79 of file mb_info.h.

15.323.2.3 drive_number

```
l4_uint8_t l4util_mb_drive_t::drive_number
```

<The size of this structure.

Definition at line 78 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

15.324 l4util_mb_info_t Struct Reference

MultiBoot Info description.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_info_t:

Data Fields

- l4_uint32_t flags

  MultiBoot info version number.
- l4_uint32_t mem_lower

  available memory below 1MB
- l4_uint32_t mem_upper

  available memory starting from 1MB [kB]
- l4_uint32_t boot_device

  "root" partition
- l4_uint32_t cmdline

  Kernel command line.
- l4_uint32_t mods_count

  number of modules
- l4_uint32_t mods_addr

  module list
- l4_uint32_t mmap_length

  size of memory mapping buffer
- l4_uint32_t mmap_addr

  address of memory mapping buffer
- l4_uint32_t drives_length

  size of drive info buffer

- l4_uint32_t drives_addr

  address of driver info buffer
- l4_uint32_t config_table

  ROM configuration table.
- l4_uint32_t boot_loader_name

  Boot Loader Name.
- l4_uint32_t apm_table

  APM table.
- l4_uint32_t vbe_ctrl_info

  VESA video contoller info.
- l4_uint32_t vbe_mode_info

  VESA video mode info.
- l4_uint16_t vbe_mode

  VESA video mode number.
- l4_uint16_t vbe_interface_seg

  VESA segment of prot BIOS interface.
- l4_uint16_t vbe_interface_off

  VESA offset of prot BIOS interface.
- l4_uint16_t vbe_interface_len

  VESA lenght of prot BIOS interface.
- l4_uint32_t tabsize

  (a.out) Kernel symbol table info
- l4_uint32_t num

  (ELF) Kernel section header table

15.324.1 Detailed Description

MultiBoot Info description.

This is the struct passed to the boot image. This is done by placing its address in the EAX register.

Definition at line 249 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

15.325 l4util_mb_mod_t Struct Reference

The structure type "mod_list" is used by the multiboot_info structure.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_mod_t:

Data Fields

- l4_uint32_t mod_start

  Starting address of module in memory.
- l4_uint32_t mod_end

  End address of module in memory.
- l4_uint32_t cmdline

  Module command line.
- l4_uint32_t pad

  padding to take it to 16 bytes

15.325.1 Detailed Description

The structure type "mod_list" is used by the multiboot_info structure.

Definition at line 35 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

15.326 l4util_mb_vbe_ctrl_t Struct Reference

VBE controller information.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_vbe_ctrl_t:

15.326.1 Detailed Description

VBE controller information.

Definition at line 108 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

15.327 l4util_mb_vbe_mode_t Struct Reference

VBE mode information.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_vbe_mode_t:

Data Fields

all VESA versions

- l4_uint16_t mode_attributes

  Mode attributes.
- l4_uint8_t win_a_attributes

Window A attributes.

- l4_uint8_t win_b_attributes

  Window B attributes.
- l4_uint16_t win_granularity

  Window granularity.
- l4_uint16_t win_size

  Window size.
- l4_uint16_t win_a_segment

  Window A start segment.
- l4_uint16_t win_b_segment

  Window B start segment.
- l4_uint32_t win_func

  Real mode pointer to window function.
- l4_uint16_t bytes_per_scanline

  Bytes per scan line.

>= VESA version 1.2

- l4_uint16_t x_resolution

  Horizontal resolution in pixels or characters.
- l4_uint16_t y_resolution

  Vertical resolution in pixels or characters.
- l4_uint8_t x_char_size

  Character cell width in pixels.
- l4_uint8_t y_char_size

  Character cell height in pixels.
- l4_uint8_t number_of_planes

  Number of memory planes.
- l4_uint8_t bits_per_pixel

  Bits per pixel.
- l4_uint8_t number_of_banks

  Number of banks.
- l4_uint8_t memory_model

  Memory model type.
- l4_uint8_t bank_size

  Bank size in KiB.
- l4_uint8_t number_of_image_pages

  Number of images.
- l4_uint8_t reserved0

  Reserved for page function.

direct color

- l4_uint8_t red_mask_size

  Size of direct color red mask in bits.
- l4_uint8_t red_field_position

  Bit position of LSB of red mask.
- l4_uint8_t green_mask_size

  Size of direct color green mask in bits.
- l4_uint8_t green_field_position

  Bit position of LSB of green mask.
- l4_uint8_t blue_mask_size

  Size of direct color blue mask in bits.
- l4_uint8_t blue_field_position

  Bit position of LSB of blue mask.
- l4_uint8_t reserved_mask_size

  Size of direct color reserved mask in bits.
- l4_uint8_t reserved_field_position

Bit position of LSB of reserved mask.

- l4_uint8_t direct_color_mode_info

  Direct color mode attributes.

>= VESA version 2.0

- l4_uint32_t phys_base

  Physical address for flat memory memory frame buffer.
- l4_uint32_t reserved1

  Reserved – always set to 0.
- l4_uint16_t reversed2

  Reserved – always set to 0.

>= VESA version 3.0

- l4_uint16_t linear_bytes_per_scanline

  Bytes per scan line for linear modes.
- l4_uint8_t banked_number_of_image_pages

  Number of images for banked modes.
- l4_uint8_t linear_number_of_image_pages

  Number of images for linear modes.
- l4_uint8_t linear_red_mask_size

  Size of direct color red mask (linear modes).
- l4_uint8_t linear_red_field_position

  Bit position of LSB of red mask (linear modes).
- l4_uint8_t linear_green_mask_size

  Size of direct color green mask (linear modes).
- l4_uint8_t linear_green_field_position

  Bit position of LSB of green mask (linear modes).
- l4_uint8_t linear_blue_mask_size

  Size of direct color blue mask (linear modes).
- l4_uint8_t linear_blue_field_position

  Bit position of LSB of blue mask (linear modes).
- l4_uint8_t linear_reserved_mask_size

  Size of direct color reserved mask (linear modes).
- l4_uint8_t linear_reserved_field_position

  Bit position of LSB of reserved mask (linear modes).
- l4_uint32_t max_pixel_clock

  Maximum pixel clock (in Hz) for graphics mode.
- l4_uint8_t **reserved3** [190]

15.327.1 Detailed Description

VBE mode information.

Definition at line 127 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

15.328 L4vbus::Device Class Reference

Device on a L4vbus::Vbus.

Inheritance diagram for L4vbus::Device:

Collaboration diagram for L4vbus::Device:

Public Member Functions

- L4::Cap< Vbus > bus_cap () const

  Access the Vbus capability of the underlying virtual bus.
- l4vbus_device_handle_t dev_handle () const

  Access the device handle of this device.

- int device_by_hid (Device *child, char const *hid, int depth=L4VBUS_MAX_DEPTH, l4vbus_device_t *devinfo=0) const

  Find a device by the hardware interface identifier (HID).
- int next_device (Device *child, int depth=L4VBUS_MAX_DEPTH, l4vbus_device_t *devinfo=0) const

  Find next child following `child`.
- int device (l4vbus_device_t *devinfo) const

  Obtain detailed information about a Vbus device.
- int get_resource (int res_idx, l4vbus_resource_t *res) const

  Obtain the resource description of an individual device resource.
- int is_compatible (char const *cid) const

  Check if the given device has a compatibility ID (CID) or HID that matches cid.
- bool operator== (Device const &o) const

  Test if two devices are the same Vbus device.
- bool operator!= (Device const &o) const

  Test if two devices are not the same.

Protected Attributes

- L4::Cap< Vbus > _bus
- l4vbus_device_handle_t _dev

  The device handle for this device.

15.328.1 Detailed Description

Device on a L4vbus::Vbus.

Definition at line 76 of file vbus.

15.328.2 Member Function Documentation

15.328.2.1 bus_cap()

```
L4::Cap<Vbus> L4vbus::Device::bus_cap ( ) const  [inline]
```

Access the Vbus capability of the underlying virtual bus.

Returns

the capability to the underlying Vbus.

Definition at line 88 of file vbus.

References _bus.

Referenced by L4vbus::Pci_host_bridge::cfg_read(), L4vbus::Pci_dev::cfg_read(), L4vbus::Pci_host_bridge::cfg_write(), L4vbus::Pci_dev::cfg_write(), L4vbus::Pci_host_bridge::irq_enable(), and L4vbus::Pci_dev::irq_enable().

Here is the caller graph for this function:

15.328.2.2 dev_handle()

```
l4vbus_device_handle_t L4vbus::Device::dev_handle ( ) const  [inline]
```

Access the device handle of this device.

Returns

the device handle for this device.

The device handle is used to directly address the device on its virtual bus.

Definition at line 97 of file vbus.

References _dev.

15.328.2.3 device()

```
int L4vbus::Device::device (
            l4vbus_device_t * devinfo ) const  [inline]
```

Obtain detailed information about a Vbus device.

Parameters

out	*devinfo*	Information structure which contains details about the device. The pointer might be NULL.

Return values

0	Success.
-L4_ENODEV	No device with the given device handle `dev` could be found.

Definition at line 169 of file vbus.

References _bus, _dev, and l4vbus_get_device().

Here is the call graph for this function:

15.328.2.4 device_by_hid()

```
int L4vbus::Device::device_by_hid (
            Device * child,
            char const * hid,
            int depth = L4VBUS_MAX_DEPTH,
            l4vbus_device_t * devinfo = 0 ) const  [inline]
```

Find a device by the hardware interface identifier (HID).

This function searches the vbus for a device with the given HID and returns a handle to the first matching device. The HID usually conforms to an ACPI HID or a Linux device tree compatible identifier.

It is possible to have multiple devices with the same HID on a vbus. In order to find all matching devices this function has to be called repeatedly with `child` pointing to the device found in the previous iteration. The iteration starts at `child` that might be any device node in the tree.

Parameters

in,out	*child*	Handle of the device from where in the device tree the search should start. To start searching from the beginning `child` must be initialized using the default (L4VBUS_NULL). If a matching device is found its handle is returned through this parameter.
	hid	HID of the device
	depth	Maximum depth for the recursive lookup
out	*devinfo*	Device information structure (might be NULL)

Return values

>=0	A device with the given HID was found.
-L4_ENOENT	No device with the given HID could be found on the vbus.
-L4_EINVAL	Invalid or no HID provided.
-L4_ENODEV	Function called on a non-existing device.

Definition at line 129 of file vbus.

15.328.2.5 get_resource()

```
int L4vbus::Device::get_resource (
            int res_idx,
            l4vbus_resource_t * res ) const  [inline]
```

Obtain the resource description of an individual device resource.

Parameters

	res_idx	Index of the resource for which the resource description should be returned. The total number of resources for a device is available in the l4vbus_device_t structure that is returned by L4vbus::Device::device_by_hid() and L4vbus::Device::next_device().
out	*res*	Descriptor of the resource.

This function returns the resource descriptor of an individual device resource selected by the `res_idx` parameter.

Return values

0	Success.
-L4_ENOENT	Invalid resource index `res_idx`.

Definition at line 189 of file vbus.

References _bus, _dev, and l4vbus_get_resource().

Here is the call graph for this function:

15.328.2.6 is_compatible()

```
int L4vbus::Device::is_compatible (
            char const * cid ) const  [inline]
```

Check if the given device has a compatibility ID (CID) or HID that matches *cid*.

Parameters

cid	the compatibility ID to test

Returns

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

Definition at line 203 of file vbus.

References _bus, _dev, and l4vbus_is_compatible().

Here is the call graph for this function:

15.328.2.7 next_device()

```
int L4vbus::Device::next_device (
            Device * child,
            int depth = L4VBUS_MAX_DEPTH,
            l4vbus_device_t * devinfo = 0 ) const  [inline]
```

Find next child following `child`.

Parameters

in,out	*child*	Handle of the device that precedes the device that shall be returned. To start from the beginning, `child` must be initialized with L4VBUS_NULL. If a device is found, its handle is returned through this parameter.
	depth	Depth to look for
out	*devinfo*	Device information (might be NULL)

Returns

0 on success, else failure

Definition at line 151 of file vbus.

15.328.2.8 operator"!=()

```
bool L4vbus::Device::operator!= (
            Device const & o ) const  [inline]
```

Test if two devices are not the same.

Returns

true if the two devices are different, false else.

Definition at line 219 of file vbus.

References _bus, and _dev.

15.328.2.9 operator==()

```
bool L4vbus::Device::operator== (
            Device const & o ) const  [inline]
```

Test if two devices are the same Vbus device.

Returns

true if the two devices are the same, false else.

Definition at line 210 of file vbus.

References _bus, and _dev.

15.328.3 Field Documentation

15.328.3.1 _bus

```
L4::Cap<Vbus> L4vbus::Device::_bus [protected]
```

The Vbus capability (where this device is located on).

Definition at line 225 of file vbus.

Referenced by bus_cap(), L4vbus::Gpio_pin::config_get(), L4vbus::Gpio_module::config_pad(), L4vbus::Gpio_pin::config_pad(), L4vbus::Gpio_pin::config_pull(), device(), L4vbus::Gpio_pin::get(), L4vbus::Gpio_module::get(), get_resource(), is_compatible(), operator!=(), operator==(), L4vbus::Gpio_pin::set(), L4vbus::Gpio_module::set(), L4vbus::Gpio_module::setup(), L4vbus::Gpio_pin::setup(), L4vbus::Gpio_pin::to_irq(), and L4vbus::Icu::vicu().

The documentation for this class was generated from the following file:

- l4/vbus/vbus

15.329 L4vbus::Gpio_module Class Reference

A Gpio_module groups multiple GPIO pins together.

Inheritance diagram for L4vbus::Gpio_module:

Collaboration diagram for L4vbus::Gpio_module:

Data Structures

- struct Pin_slice

  A slice of the pins provided by this module.

Public Member Functions

- int setup (Pin_slice const &mask, unsigned mode, unsigned value) const

  Configure function of multiple GPIO pins at once.
- int config_pad (Pin_slice const &mask, unsigned func, unsigned value) const

  Hardware specific configuration function for multiple GPIO pins.
- int get (unsigned offset, unsigned *data) const

  Read values of multiple GPIO pins at once.
- int set (Pin_slice const &mask, unsigned data)

  Set multiple GPIO output pins at once.
- Gpio_pin pin (unsigned pin) const

  Get Gpio_pin for a specific pin of this Gpio_module.

Additional Inherited Members

15.329.1 Detailed Description

A Gpio_module groups multiple GPIO pins together.

Definition at line 135 of file vbus_gpio.

15.329.2 Member Function Documentation

15.329.2.1 config_pad()

```
int L4vbus::Gpio_module::config_pad (
            Pin_slice const & mask,
            unsigned func,
            unsigned value ) const  [inline]
```

Hardware specific configuration function for multiple GPIO pins.

Parameters

mask	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
func	Hardware specific configuration register, usually offset to the GPIO chip's base address.
value	Value which is written into the hardware specific configuration register for the specified pins

Returns

0 if OK, error code otherwise

Definition at line 187 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_multi_config_pad().

Here is the call graph for this function:

15.329.2.2 get()

```
int L4vbus::Gpio_module::get (
            unsigned offset,
            unsigned * data ) const  [inline]
```

Read values of multiple GPIO pins at once.

Parameters

	offset	Pin corresponding to the LSB in *data*. Note: allowed may be hardware specific.
out	*data*	Each bit returns the value (0 or 1) for the corresponding GPIO pin. The value of pins that are not accessible is undefined.

Returns

0 if OK, error code otherwise

Definition at line 203 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_multi_get().

Here is the call graph for this function:

15.329.2.3 pin()

```
Gpio_pin L4vbus::Gpio_module::pin (
            unsigned pin ) const  [inline]
```

Get Gpio_pin for a specific pin of this Gpio_module.

Parameters

pin	GPIO pin number to get Gpio_pin for.

Returns

Gpio_pin

Definition at line 231 of file vbus_gpio.

15.329.2.4 set()

```
int L4vbus::Gpio_module::set (
            Pin_slice const & mask,
            unsigned data )  [inline]
```

Set multiple GPIO output pins at once.

Parameters

mask	Mask of GPIO pins to set. A bit set to 1 selects this pin. A maximum of 32 pins can be set at once. The real number depends on the hardware and the driver implementation.
data	Each bit corresponds to the GPIO pin in *mask*. The value of each bit is written to the GPIO pin if its bit in *mask* is set.

Returns

0 if OK, error code otherwise

Definition at line 219 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_multi_set().

Here is the call graph for this function:

15.329.2.5 setup()

```
int L4vbus::Gpio_module::setup (
            Pin_slice const & mask,
            unsigned mode,
            unsigned value ) const  [inline]
```

Configure function of multiple GPIO pins at once.

Parameters

mask	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
mode	GPIO function, see L4vbus_gpio_generic_func for generic functions. Hardware specific functions must be provided in the lower 8 bits.
value	Optional value to set the GPIO pins to if they are configured as output pins

Returns

0 if OK, error code otherwise

Definition at line 168 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_multi_setup().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/vbus/vbus_gpio

15.330 L4vbus::Gpio_module::Pin_slice Struct Reference

A slice of the pins provided by this module.

Collaboration diagram for L4vbus::Gpio_module::Pin_slice:

15.330.1 Detailed Description

A slice of the pins provided by this module.

Data type to specify a selection of pins for the 'multi' methods.

Definition at line 148 of file vbus_gpio.

The documentation for this struct was generated from the following file:

- l4/vbus/vbus_gpio

15.331 L4vbus::Gpio_pin Class Reference

A GPIO pin.

Inheritance diagram for L4vbus::Gpio_pin:

Collaboration diagram for L4vbus::Gpio_pin:

Public Member Functions

- int get () const

  Read value of GPIO input pin.
- int set (int value) const

  Set GPIO output pin.
- int setup (unsigned mode, unsigned value) const

Configure the function of a GPIO pin.

- int config_pull (unsigned mode) const

  Generic function to set pull up/down mode.
- int config_pad (unsigned func, unsigned value) const

  Hardware specific configuration function.
- int config_get (unsigned func, unsigned *value) const

  Read hardware specific configuration.
- int to_irq () const

  Create IRQ for GPIO pin.
- unsigned pin () const

  Get pin number.

Additional Inherited Members

15.331.1 Detailed Description

A GPIO pin.

Definition at line 28 of file vbus_gpio.

15.331.2 Member Function Documentation

15.331.2.1 config_get()

```
int L4vbus::Gpio_pin::config_get (
            unsigned func,
            unsigned * value ) const  [inline]
```

Read hardware specific configuration.

Parameters

	func	Hardware specific configuration register to read from. Usually this is an offset to the GPIO chip's base address.
out	*value*	The configuration value.

Returns

0 if OK, error code otherwise

Definition at line 104 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_config_get().

Here is the call graph for this function:

15.331.2.2 config_pad()

```
int L4vbus::Gpio_pin::config_pad (
            unsigned func,
            unsigned value ) const  [inline]
```

Hardware specific configuration function.

Parameters

func	Hardware specific configuration register, usually offset to the GPIO chip's base address
value	Value which is written into the hardware specific configuration register for the specified pin

Returns

0 if OK, error code otherwise

Definition at line 91 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_config_pad().

Here is the call graph for this function:

15.331.2.3 config_pull()

```
int L4vbus::Gpio_pin::config_pull (
            unsigned mode ) const  [inline]
```

Generic function to set pull up/down mode.

Parameters

mode	mode for pull up/down resistors, see L4vbus_gpio_pull_modes

Returns

0 if OK, error code otherwise

Definition at line 77 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_config_pull().

Here is the call graph for this function:

15.331.2.4 get()

```
int L4vbus::Gpio_pin::get ( ) const [inline]
```

Read value of GPIO input pin.

Returns

Value of GPIO pin (usually 0 or 1), negative error code otherwise.

Definition at line 40 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_get().

Here is the call graph for this function:

15.331.2.5 pin()

```
unsigned L4vbus::Gpio_pin::pin ( ) const  [inline]
```

Get pin number.

Returns

GPIO pin number

Definition at line 124 of file vbus_gpio.

15.331.2.6 set()

```
int L4vbus::Gpio_pin::set (
            int value ) const  [inline]
```

Set GPIO output pin.

Parameters

value	Value to write to the GPIO pin (usually 0 or 1)

Returns

0 if OK, error code otherwise

Definition at line 51 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_set().

Here is the call graph for this function:

15.331.2.7 setup()

```
int L4vbus::Gpio_pin::setup (
            unsigned mode,
            unsigned value ) const  [inline]
```

Configure the function of a GPIO pin.

Parameters

mode	GPIO function, see L4vbus_gpio_generic_func for generic functions. Hardware specific functions must be provided in the lower 8 bits.
value	Optional value to set the GPIO pin to if it is configured as an output pin

Returns

0 if OK, error code otherwise

Definition at line 66 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_setup().

Here is the call graph for this function:

15.331.2.8 to_irq()

```
int L4vbus::Gpio_pin::to_irq ( ) const [inline]
```

Create IRQ for GPIO pin.

Returns

IRQ number if OK, negative error code otherwise

Definition at line 114 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_to_irq().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/vbus/vbus_gpio

15.332 L4vbus::Icu Class Reference

Vbus Interrupt controller API.

Inheritance diagram for L4vbus::Icu:

Collaboration diagram for L4vbus::Icu:

Public Types

- enum Src_types { Src_dev_handle = L4VBUS_ICU_SRC_DEV_HANDLE }

  Flags that can be used with the ICU on a vbus device.

Public Member Functions

- int vicu (L4::Cap< L4::Icu > icu) const

  Request an L4::Icu capability for this Vbus's virtual ICU.

Additional Inherited Members

15.332.1 Detailed Description

Vbus Interrupt controller API.

Every Vbus contains a virtual interrupt control unit that manages the IRQs of the devices on the bus. This class provides access to a capability to an L4::Icu object which can then be used to interface with the IRQs.

See also

L4::Icu

Definition at line 240 of file vbus.

15.332.2 Member Enumeration Documentation

15.332.2.1 Src_types

```
enum L4vbus::Icu::Src_types
```

Flags that can be used with the ICU on a vbus device.

Enumerator

Src_dev_handle	Flag to denote that the value should be interpreted as a device handle. This flag may be used in the `source` parameter in L4::Icu::msi_info() to denote that the ICU should interpret the source ID as a device handle.

Definition at line 244 of file vbus.

15.332.3 Member Function Documentation

15.332.3.1 vicu()

```
int L4vbus::Icu::vicu (
            L4::Cap< L4::Icu > icu ) const  [inline]
```

Request an L4::Icu capability for this Vbus's virtual ICU.

Parameters

out	*icu*	Capability slot where the L4::Icu capability shall be stored.

Return values

0	Success.
otherwise	IPC error.

Definition at line 265 of file vbus.

References L4vbus::Device::_bus, L4vbus::Device::_dev, L4::Cap_base::cap(), and l4vbus_vicu_get_cap().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/vbus/vbus

15.333 L4vbus::Pci_dev Class Reference

A PCI device.

Inheritance diagram for L4vbus::Pci_dev:

Collaboration diagram for L4vbus::Pci_dev:

Public Member Functions

- int cfg_read (l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width) const

  Read from the device's vPCI configuration space.
- int cfg_write (l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width) const

  Write to the device's vPCI configuration space.
- int irq_enable (unsigned char *trigger, unsigned char *polarity) const

  Enable the device's PCI interrupt.

Additional Inherited Members

15.333.1 Detailed Description

A PCI device.

Definition at line 95 of file vbus_pci.

15.333.2 Member Function Documentation

15.333.2.1 cfg_read()

```
int L4vbus::Pci_dev::cfg_read (
            l4_uint32_t reg,
            l4_uint32_t * value,
            l4_uint32_t width ) const [inline]
```

Read from the device's vPCI configuration space.

Parameters

	reg	Register in configuration space to read
out	*value*	Value that has been read
	width	Width to read in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Definition at line 107 of file vbus_pci.

References L4vbus::Device::_dev, L4vbus::Device::bus_cap(), and l4vbus_pcidev_cfg_read().

Here is the call graph for this function:

15.333.2.2 cfg_write()

```
int L4vbus::Pci_dev::cfg_write (
            l4_uint32_t reg,
            l4_uint32_t value,
            l4_uint32_t width ) const [inline]
```

Write to the device's vPCI configuration space.

Parameters

reg	Register in configuration space to write
value	Value to write
width	Width to write in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Definition at line 123 of file vbus_pci.

References L4vbus::Device::_dev, L4vbus::Device::bus_cap(), and l4vbus_pcidev_cfg_write().

Here is the call graph for this function:

15.333.2.3 irq_enable()

```
int L4vbus::Pci_dev::irq_enable (
            unsigned char * trigger,
            unsigned char * polarity ) const [inline]
```

Enable the device's PCI interrupt.

Parameters

out	*trigger*	False if interrupt is level-triggered
out	*polarity*	True if interrupt is of low polarity

Returns

On success: Interrupt line to be used, else failure

Definition at line 139 of file vbus_pci.

References L4vbus::Device::_dev, L4vbus::Device::bus_cap(), and l4vbus_pcidev_irq_enable().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/vbus/vbus_pci

15.334 L4vbus::Pci_host_bridge Class Reference

A Pci host bridge.

Inheritance diagram for L4vbus::Pci_host_bridge:

Collaboration diagram for L4vbus::Pci_host_bridge:

Public Member Functions

- int cfg_read (l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width) const

  Read from the vPCI configuration space using the PCI root bridge.
- int cfg_write (l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width) const

  Write to the vPCI configuration space using the PCI root bridge.

- int irq_enable (l4_uint32_t bus, l4_uint32_t devfn, int pin, unsigned char *trigger, unsigned char *polarity) const

  Enable PCI interrupt for a specific device using the PCI root bridge.

Additional Inherited Members

15.334.1 Detailed Description

A Pci host bridge.

Definition at line 27 of file vbus_pci.

15.334.2 Member Function Documentation

15.334.2.1 cfg_read()

```
int L4vbus::Pci_host_bridge::cfg_read (
            l4_uint32_t bus,
            l4_uint32_t devfn,
            l4_uint32_t reg,
            l4_uint32_t * value,
            l4_uint32_t width ) const  [inline]
```

Read from the vPCI configuration space using the PCI root bridge.

Parameters

	bus	Bus number
	devfn	Device id (upper 16bit) and function (lower 16bit)
	reg	Register in configuration space to read
out	*value*	Value that has been read
	width	Width to read in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Definition at line 41 of file vbus_pci.

References L4vbus::Device::_dev, L4vbus::Device::bus_cap(), and l4vbus_pci_cfg_read().

Here is the call graph for this function:

15.334.2.2 cfg_write()

```
int L4vbus::Pci_host_bridge::cfg_write (
            l4_uint32_t bus,
            l4_uint32_t devfn,
            l4_uint32_t reg,
            l4_uint32_t value,
            l4_uint32_t width ) const  [inline]
```

Write to the vPCI configuration space using the PCI root bridge.

Parameters

bus	Bus number
devfn	Device id (upper 16bit) and function (lower 16bit)
reg	Register in configuration space to write
value	Value to write
width	Width to write in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Definition at line 60 of file vbus_pci.

References L4vbus::Device::_dev, L4vbus::Device::bus_cap(), and l4vbus_pci_cfg_write().

Here is the call graph for this function:

15.334.2.3 irq_enable()

```
int L4vbus::Pci_host_bridge::irq_enable (
            l4_uint32_t bus,
            l4_uint32_t devfn,
            int pin,
            unsigned char * trigger,
            unsigned char * polarity ) const  [inline]
```

Enable PCI interrupt for a specific device using the PCI root bridge.

Parameters

	bus	Bus number
	devfn	Device id (upper 16bit) and function (lower 16bit)
	pin	Interrupt pin (normally as reported in configuration register INTR)
out	*trigger*	False if interrupt is level-triggered
out	*polarity*	True if interrupt is of low polarity

Returns

On success: Interrupt line to be used, else failure

Definition at line 81 of file vbus_pci.

References L4vbus::Device::_dev, L4vbus::Device::bus_cap(), and l4vbus_pci_irq_enable().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/vbus/vbus_pci

15.335 L4vbus::Pm< DEC > Class Template Reference

Power-management API mixin.

Inheritance diagram for L4vbus::Pm< DEC >:

Collaboration diagram for L4vbus::Pm< DEC >:

Public Member Functions

- int pm_suspend () const

  Suspend the module.
- int pm_resume () const

  Resume the module.

15.335.1 Detailed Description

template<typename DEC>
class L4vbus::Pm< DEC >

Power-management API mixin.

Devices that inherit from this mixin provide an API to be suspended and resumed.

Definition at line 52 of file vbus.

The documentation for this class was generated from the following file:

- l4/vbus/vbus

15.336 L4vbus::Vbus Class Reference

The virtual bus (Vbus) interface.

Inheritance diagram for L4vbus::Vbus:

Collaboration diagram for L4vbus::Vbus:

Public Member Functions

- int request_ioport (l4vbus_resource_t *res) const

  Request the given IO port resource from the bus.
- int request_resource (l4vbus_resource_t *res, int=0) const

  Request the given resource from the bus.
- int release_ioport (l4vbus_resource_t *res) const

Release the given IO port resource from the bus.

- int release_resource (l4vbus_resource_t *res) const

  Release the given resource from the bus.

- Device root () const

  Get the root device of the device tree of this bus.

- int assign_dma_domain (unsigned domain_id, unsigned flags, L4::Cap< L4Re::Dma_space > dma_space) const

  Assign an L4Re::Dma_space to a DMA domain.

- int assign_dma_domain (unsigned domain_id, unsigned flags, L4::Cap< L4::Task > dma_space) const

  Assign a kernel DMA space to a DMA domain.

Additional Inherited Members

15.336.1 Detailed Description

The virtual bus (Vbus) interface.

See also

L4Re Vbus API

Definition at line 278 of file vbus.

15.336.2 Member Function Documentation

15.336.2.1 assign_dma_domain() [1/2]

```
int L4vbus::Vbus::assign_dma_domain (
            unsigned domain_id,
            unsigned flags,
            L4::Cap< L4::Task > dma_space ) const  [inline]
```

Assign a kernel DMA space to a DMA domain.

Parameters

domain_id	DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.
flags	A combination of L4vbus_dma_domain_assign_flags.
dma_space	The DMA space capability to bind or unbind, this must be a kernel DMA space (L4::Task created with L4_PROTO_DMA_SPACE)

Return values

0	Operation completed successfully.

Return values

-L4_ENOENT	The vbus does not support a global DMA domain or no DMA domain could be found.
-L4_EINVAL	Invalid argument used.
-L4_EBUSY	DMA domain is already active, this means another DMA space is already assigned.

Definition at line 393 of file vbus.

References L4::Cap_base::cap(), L4Re::Dataspace::flags(), l4vbus_assign_dma_domain(), L4VBUS_DMAD_KERNEL_DMA_SPACE and L4VBUS_DMAD_L4RE_DMA_SPACE.

Here is the call graph for this function:

15.336.2.2 assign_dma_domain() [2/2]

```
int L4vbus::Vbus::assign_dma_domain (
            unsigned domain_id,
            unsigned flags,
            L4::Cap< L4Re::Dma_space > dma_space ) const  [inline]
```

Assign an L4Re::Dma_space to a DMA domain.

Parameters

domain_id	DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.
flags	A combination of L4vbus_dma_domain_assign_flags.
dma_space	The DMA space capability to bind or unbind, this must be an L4Re::Dma_space

Return values

0	Operation completed successfully.
-L4_ENOENT	The vbus does not support a global DMA domain or no DMA domain could be found.
-L4_EINVAL	Invalid argument used.

Return values

-L4_EBUSY	DMA domain is already active, this means another DMA space is already assigned.

Definition at line 368 of file vbus.

References L4::Cap_base::cap(), L4Re::Dataspace::flags(), l4vbus_assign_dma_domain(), L4VBUS_DMAD_KERNEL_DMA_SPACE and L4VBUS_DMAD_L4RE_DMA_SPACE.

Here is the call graph for this function:

15.336.2.3 release_ioport()

```
int L4vbus::Vbus::release_ioport (
            l4vbus_resource_t * res ) const  [inline]
```

Release the given IO port resource from the bus.

Parameters

in	*res*	The IO port resource to be released from the bus.

Returns

>=0 on success, <0 on error.

Definition at line 321 of file vbus.

References l4vbus_release_ioport().

Here is the call graph for this function:

15.336.2.4 release_resource()

```
int L4vbus::Vbus::release_resource (
            l4vbus_resource_t * res ) const  [inline]
```

Release the given resource from the bus.

Parameters

in	*res*	The resource to be released from the bus.

Returns

>=0 on success, <0 on error.

Deprecated This function is deprecated since Q3 2019. Use release_ioport() instead.

Definition at line 338 of file vbus.

References l4vbus_release_ioport().

Here is the call graph for this function:

15.336.2.5 request_ioport()

```
int L4vbus::Vbus::request_ioport (
            l4vbus_resource_t * res ) const  [inline]
```

Request the given IO port resource from the bus.

Parameters

in	*res*	The IO port resource to be requested from the bus.

Return values

0	Success.
-L4_EINVAL	Resource is not an IO port resource.
-L4_ENOENT	No matching IO port resource found.

Definition at line 291 of file vbus.

References l4vbus_request_ioport().

Here is the call graph for this function:

15.336.2.6 request_resource()

```
int L4vbus::Vbus::request_resource (
            l4vbus_resource_t * res,
            int  = 0 ) const  [inline]
```

Request the given resource from the bus.

Parameters

in	*res*	The resource to be requested from the bus.
	flags	Ignored.

Returns

0 on success, else failure.

Deprecated This function is deprecated since Q3 2019. Use request_ioport() instead.

Definition at line 309 of file vbus.

References l4vbus_request_ioport().

Here is the call graph for this function:

15.336.2.7 root()

```
Device L4vbus::Vbus::root ( ) const [inline]
```

Get the root device of the device tree of this bus.

Returns

A Vbus device representing the root of the device tree.

Definition at line 347 of file vbus.

References L4VBUS_ROOT_BUS.

The documentation for this class was generated from the following file:

- l4/vbus/vbus

15.337 l4vbus_device_t Struct Reference

Detailed information about a vbus device.

```
#include <vbus_types.h>
```

Collaboration diagram for l4vbus_device_t:

Data Fields

- l4_uint32_t type

  Bitfield of supported sub-interfaces, see l4vbus_iface_type_t.
- char name [L4VBUS_DEV_NAME_LEN]

  Name.
- unsigned num_resources

  Number of resources for this device.
- unsigned flags

  Flags, see l4vbus_device_flags_t.

15.337.1 Detailed Description

Detailed information about a vbus device.

Definition at line 66 of file vbus_types.h.

The documentation for this struct was generated from the following file:

- l4/vbus/vbus_types.h

15.338 l4vbus_resource_t Struct Reference

Description of a single vbus resource.

```
#include <vbus_types.h>
```

Collaboration diagram for l4vbus_resource_t:

Data Fields

- l4_uint16_t type

  Resource type, see l4vbus_resource_type_t.
- l4_uint16_t flags

  Flags.
- l4vbus_paddr_t start

  Start of resource range.
- l4vbus_paddr_t end

  End of resource range (inclusive)
- l4vbus_device_handle_t provider

  Device handle of the provider of the resource.
- l4_uint32_t id

  Resource ID (4 bytes), usually a 4 letter ASCII name is used.

15.338.1 Detailed Description

Description of a single vbus resource.

Definition at line 23 of file vbus_types.h.

The documentation for this struct was generated from the following file:

- l4/vbus/vbus_types.h

15.339 L4vcpu::State Class Reference

C++ implementation of state word in the vCPU area.

Collaboration diagram for L4vcpu::State:

Public Member Functions

- State (unsigned v)

  Initialize state.
- void add (unsigned bits) throw ()

  Add flags.
- void clear (unsigned bits) throw ()

  Clear flags.
- void set (unsigned v) throw ()

  Set flags.

15.339.1 Detailed Description

C++ implementation of state word in the vCPU area.

Definition at line 31 of file vcpu.

15.339.2 Constructor & Destructor Documentation

15.339.2.1 State()

```
L4vcpu::State::State (
            unsigned v )  [inline], [explicit]
```

Initialize state.

Parameters

v	Initial state.

Definition at line 41 of file vcpu.

15.339.3 Member Function Documentation

15.339.3.1 add()

```
void L4vcpu::State::add (
            unsigned bits ) throw ( )   [inline]
```

Add flags.

Parameters

bits	Bits to add to the word.

Definition at line 48 of file vcpu.

15.339.3.2 clear()

```
void L4vcpu::State::clear (
            unsigned bits ) throw ( )   [inline]
```

Clear flags.

Parameters

bits	Bits to clear in the word.

Definition at line 55 of file vcpu.

15.339.3.3 set()

```
void L4vcpu::State::set (
            unsigned v ) throw ( )   [inline]
```

Set flags.

Parameters

v	Set the word to the value of v.

Definition at line 62 of file vcpu.

The documentation for this class was generated from the following file:

- l4/vcpu/vcpu

15.340 L4vcpu::Vcpu Class Reference

C++ implementation of the vCPU save state area.

Inheritance diagram for L4vcpu::Vcpu:

Collaboration diagram for L4vcpu::Vcpu:

Public Member Functions

- void irq_disable () throw ()

  Disable the vCPU for event delivery.
- unsigned irq_disable_save () throw ()

  Disable the vCPU for event delivery and return previous state.
- State ∗ state () throw ()

Get state word.

- State state () const throw ()

  Get state word.
- State ∗ saved_state () throw ()

  Get saved_state word.
- State saved_state () const throw ()

  Get saved_state word.
- l4_uint16_t sticky_flags () const throw ()

  Get sticky flags.
- void irq_enable (l4_utcb_t ∗utcb, l4vcpu_event_hndl_t do_event_work_cb, l4vcpu_setup_ipc_t setup_ipc) throw ()

  Enable the vCPU for event delivery.
- void irq_restore (unsigned s, l4_utcb_t ∗utcb, l4vcpu_event_hndl_t do_event_work_cb, l4vcpu_setup_ipc_t setup_ipc) throw ()

  Restore a previously saved IRQ/event state.
- void wait_for_event (l4_utcb_t ∗utcb, l4vcpu_event_hndl_t do_event_work_cb, l4vcpu_setup_ipc_t setup_ipc) throw ()

  Wait for event.
- void task (L4::Cap< L4::Task > const task=L4::Cap< L4::Task >::Invalid) throw ()

  Set the task of the vCPU.
- int is_page_fault_entry () const

  Return whether the entry reason was a page fault.
- int is_irq_entry () const

  Return whether the entry reason was an IRQ/IPC message.
- l4_vcpu_regs_t ∗ r () throw ()

  Return pointer to register state.
- l4_vcpu_regs_t const ∗ r () const throw ()

  Return pointer to register state.
- l4_vcpu_ipc_regs_t ∗ i () throw ()

  Return pointer to IPC state.
- l4_vcpu_ipc_regs_t const ∗ i () const throw ()

  Return pointer to IPC state.
- void entry_sp (l4_umword_t sp)

  Set vCPU entry stack pointer.
- void entry_ip (l4_umword_t ip)

  Set vCPU entry instruction pointer.
- void print_state (const char ∗prefix="") const throw ()

  Print the state of the vCPU.

Static Public Member Functions

- static int ext_alloc (Vcpu ∗∗vcpu, l4_addr_t ∗ext_state, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) throw ()

  Allocate state area for an extended vCPU.
- static Vcpu ∗ cast (void ∗x) throw ()

  Cast a void pointer to a class pointer.
- static Vcpu ∗ cast (l4_addr_t x) throw ()

  Cast an address to a class pointer.

15.340.1 Detailed Description

C++ implementation of the vCPU save state area.

Definition at line 72 of file vcpu.

15.340.2 Member Function Documentation

15.340.2.1 cast() [1/2]

```
static Vcpu* L4vcpu::Vcpu::cast (
            l4_addr_t x ) throw ( )   [inline], [static]
```

Cast an address to a class pointer.

Parameters

x	Pointer.

Returns

Pointer to Vcpu class.

Definition at line 275 of file vcpu.

15.340.2.2 cast() [2/2]

```
static Vcpu* L4vcpu::Vcpu::cast (
            void * x ) throw ( )   [inline], [static]
```

Cast a void pointer to a class pointer.

Parameters

x	Pointer.

Returns

Pointer to Vcpu class.

Definition at line 265 of file vcpu.

15.340.2.3 entry_ip()

```
void L4vcpu::Vcpu::entry_ip (
            l4_umword_t ip )  [inline]
```

Set vCPU entry instruction pointer.

Parameters

ip	Instruction pointer address to set.

Definition at line 239 of file vcpu.

References l4_vcpu_state_t::entry_ip.

15.340.2.4 entry_sp()

```
void L4vcpu::Vcpu::entry_sp (
            l4_umword_t sp )  [inline]
```

Set vCPU entry stack pointer.

Parameters

sp	Stack pointer address to set.

Note

The value is only used when entering from a user-task.

Definition at line 232 of file vcpu.

References l4_vcpu_state_t::entry_sp.

15.340.2.5 ext_alloc()

```
static int L4vcpu::Vcpu::ext_alloc (
            Vcpu ** vcpu,
            l4_addr_t * ext_state,
            L4::Cap< L4::Task > task = L4Re::Env::env() ->task(),
            L4::Cap< L4Re::Rm > rm = L4Re::Env::env() ->rm() ) throw ( )   [static]
```

Allocate state area for an extended vCPU.

Return values

vcpu	Allocated vcpu-state area.
ext_state	Allocated extended vcpu-state area.

Parameters

task	Task to use for allocation, defaults to own task.
rm	Region manager to use for allocation defaults to standard region manager.

Returns

0 for success, error code otherwise

15.340.2.6 i() [1/2]

```
l4_vcpu_ipc_regs_t* L4vcpu::Vcpu::i ( ) throw ( )   [inline]
```

Return pointer to IPC state.

Returns

Pointer to IPC state.

Definition at line 216 of file vcpu.

References l4_vcpu_state_t::i.

15.340.2.7 i() [2/2]

```
l4_vcpu_ipc_regs_t const* L4vcpu::Vcpu::i ( ) const throw ( )   [inline]
```

Return pointer to IPC state.

Returns

Pointer to IPC state.

Definition at line 223 of file vcpu.

References l4_vcpu_state_t::i.

15.340.2.8 irq_disable_save()

```
unsigned L4vcpu::Vcpu::irq_disable_save ( ) throw ( )   [inline]
```

Disable the vCPU for event delivery and return previous state.

Returns

IRQ state before disabling IRQs.

Definition at line 85 of file vcpu.

References l4vcpu_irq_disable_save().

Here is the call graph for this function:

15.340.2.9 irq_enable()

```
void L4vcpu::Vcpu::irq_enable (
            l4_utcb_t * utcb,
            l4vcpu_event_hndl_t do_event_work_cb,
            l4vcpu_setup_ipc_t setup_ipc ) throw ( )   [inline]
```

Enable the vCPU for event delivery.

Parameters

utcb	The UTCB to use.
do_event_work_cb	Call-back function that is called in case an event (such as an interrupt) is pending.
setup_ipc	Call-back function that is called before an IPC operation is called, and before event delivery is enabled.

Definition at line 142 of file vcpu.

References l4vcpu_irq_enable().

Here is the call graph for this function:

15.340.2.10 irq_restore()

```
void L4vcpu::Vcpu::irq_restore (
            unsigned s,
            l4_utcb_t * utcb,
            l4vcpu_event_hndl_t do_event_work_cb,
            l4vcpu_setup_ipc_t setup_ipc ) throw ( )   [inline]
```

Restore a previously saved IRQ/event state.

Parameters

s	IRQ state to be restored.
utcb	The UTCB to use.
do_event_work_cb	Call-back function that is called in case an event (such as an interrupt) is pending.
setup_ipc	Call-back function that is called before an IPC operation is called, and before event delivery is enabled.

Definition at line 157 of file vcpu.

References l4vcpu_irq_restore().

Here is the call graph for this function:

15.340.2.11 is_irq_entry()

```
int L4vcpu::Vcpu::is_irq_entry ( ) const  [inline]
```

Return whether the entry reason was an IRQ/IPC message.

return 0 if not, !=0 otherwise.

Definition at line 195 of file vcpu.

References l4vcpu_is_irq_entry().

Here is the call graph for this function:

15.340.2.12 is_page_fault_entry()

```
int L4vcpu::Vcpu::is_page_fault_entry ( ) const  [inline]
```

Return whether the entry reason was a page fault.

return 0 if not, !=0 otherwise.

Definition at line 188 of file vcpu.

References l4vcpu_is_page_fault_entry().

Here is the call graph for this function:

15.340.2.13 r() [1/2]

```
l4_vcpu_regs_t* L4vcpu::Vcpu::r ( ) throw ( )  [inline]
```

Return pointer to register state.

Returns

Pointer to register state.

Definition at line 202 of file vcpu.

References l4_vcpu_state_t::r.

15.340.2.14 r() [2/2]

```
l4_vcpu_regs_t const* L4vcpu::Vcpu::r ( ) const throw ( )   [inline]
```

Return pointer to register state.

Returns

Pointer to register state.

Definition at line 209 of file vcpu.

References l4_vcpu_state_t::r.

15.340.2.15 saved_state() [1/2]

```
State* L4vcpu::Vcpu::saved_state ( ) throw ( )   [inline]
```

Get saved_state word.

Returns

Pointer to saved_state word in the vCPU

Definition at line 113 of file vcpu.

References l4_vcpu_state_t::saved_state.

15.340.2.16 saved_state() [2/2]

```
State L4vcpu::Vcpu::saved_state ( ) const throw ( )   [inline]
```

Get saved_state word.

Returns

Pointer to saved_state word in the vCPU

Definition at line 123 of file vcpu.

References l4_vcpu_state_t::saved_state.

15.340.2.17 state() [1/2]

```
State* L4vcpu::Vcpu::state ( ) throw ( )   [inline]
```

Get state word.

Returns

Pointer to state word in the vCPU

Definition at line 95 of file vcpu.

References l4_vcpu_state_t::state.

15.340.2.18 state() [2/2]

```
State L4vcpu::Vcpu::state ( ) const throw ( )   [inline]
```

Get state word.

Returns

Pointer to state word in the vCPU

Definition at line 106 of file vcpu.

References l4_vcpu_state_t::state.

15.340.2.19 task()

```
void L4vcpu::Vcpu::task (
            L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid ) throw ( )   [inline]
```

Set the task of the vCPU.

Parameters

task	Task to set, defaults to invalid task.

Definition at line 181 of file vcpu.

References l4_vcpu_state_t::user_task.

15.340.2.20 wait_for_event()

```
void L4vcpu::Vcpu::wait_for_event (
            l4_utcb_t * utcb,
            l4vcpu_event_hndl_t do_event_work_cb,
            l4vcpu_setup_ipc_t setup_ipc ) throw ( )   [inline]
```

Wait for event.

Parameters

utcb	The UTCB to use.
do_event_work_cb	Call-back function that is called in case an event (such as an interrupt) is pending.
setup_ipc	Call-back function that is called before an IPC operation is called.

Note that event delivery remains disabled after this function returns.

Definition at line 173 of file vcpu.

References l4vcpu_wait_for_event().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/vcpu/vcpu

15.341 L4virtio::Device Class Reference

IPC interface for virtio over L4 IPC.

Inheritance diagram for L4virtio::Device:

Collaboration diagram for L4virtio::Device:

Public Member Functions

- long set_status (unsigned status)

  Write the VIRTIO status register.
- long config_queue (unsigned queue)

  Trigger queue configuration of the given queue.

- long register_ds (L4::Ipc::Cap< L4Re::Dataspace > ds_cap, l4_uint64_t base, l4_umword_t offset, l4_umword_t size)

  Register a shared data space with VIRTIO host.
- long register_iface (L4::Ipc::Cap< L4::Triggerable > guest_irq, L4::Ipc::Out< L4::Cap< L4::Triggerable > > host_irq, L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > config_ds)

  Register client to the L4-VIRTIO device.
- long device_config (L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > config_ds, l4_addr_t *ds_offset)

  Get the dataspace with the L4virtio configuration page.
- long device_notification_irq (unsigned index, L4::Ipc::Out< L4::Cap< L4::Triggerable > > irq)

  Get the notification interrupt corresponding to the given index.

Additional Inherited Members

15.341.1 Detailed Description

IPC interface for virtio over L4 IPC.

The L4virtio protocol is an adaption of the mmio virtio transport 1.0(4). This interface allows to exchange the necessary resources: device configuration page, notification interrupts and dataspaces for payload.

Notification interrupts can be configured independently for changes to the configuration space and each queue through special L4virtio-specific notify_index fields in the config page and queue configuration. The interface distinguishes between device-to-driver and driver-to-device notification interrupts.

Device-to-driver interrupts are configured via the ICU interface. The device announces the maximum number of supported interrupts via Icu::info(). The driver can then bind interrupts using Icu::bind().

Driver-to-device interrupts must be requested from the device through device_notification_irq().

Definition at line 39 of file l4virtio.

15.341.2 Member Function Documentation

15.341.2.1 config_queue()

```
long L4virtio::Device::config_queue (
            unsigned queue )
```

Trigger queue configuration of the given queue.

Usually all queues are configured when the status is written to running. However, in some cases queues shall be disabled or enabled dynamically, in this case this function triggers a reconfiguration from the shared memory register of the queue config.

Parameters

queue	Queue index for the queue to be configured.

Return values

0	on success.
-L4_EIO	The queue's status is invalid.
-L4_ERANGE	The queue index exceeds the number of queues.
-L4_EINVAL	Otherwise.

Referenced by L4virtio::Driver::Device::config_queue().

Here is the caller graph for this function:

15.341.2.2 device_config()

```
long L4virtio::Device::device_config (
            L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > config_ds,
            l4_addr_t * ds_offset )
```

Get the dataspace with the L4virtio configuration page.

Parameters

config_ds	Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space.
ds_offset	Offset into the dataspace where the device configuration structure starts.

Referenced by L4virtio::Driver::Device::driver_connect().

Here is the caller graph for this function:

15.341.2.3 device_notification_irq()

```
long L4virtio::Device::device_notification_irq (
            unsigned index,
            L4::Ipc::Out< L4::Cap< L4::Triggerable > > irq )
```

Get the notification interrupt corresponding to the given index.

Parameters

	index	Index of the interrupt.
out	*irq*	Triggerable for the given index.

Return values

L4_EOK	Success.
L4_ENOSYS	IRQ notification not supported by device.
<0	Other error.

An index is only guaranteed to return an IRQ object when the index is set in one of the device notify index fields. The device must return the same interrupt for a given index as long as the index is in use. If an index disappears as a result of a configuration change and then is reused later, the interrupt is not guaranteed to be the same.

Interrupts must always be rerequested after a device reset.

Referenced by L4virtio::Driver::Device::driver_connect().

Here is the caller graph for this function:

15.341.2.4 register_ds()

```
long L4virtio::Device::register_ds (
            L4::Ipc::Cap< L4Re::Dataspace > ds_cap,
            l4_uint64_t base,
            l4_umword_t offset,
            l4_umword_t size )
```

Register a shared data space with VIRTIO host.

Parameters

ds_cap	Dataspace capability to register. The lower 8 bits determine the rights mask with which the guest's rights are masked during the registration of the dataspace at the VIRTIO host.
base	VIRTIO guest physical start address of shared memory region
offset	Offset within the data space that is attached to the given `base` in the guest physical memory.
size	Size of the memory region in the guest

Returns

0 on success, < 0 on error

Referenced by L4virtio::Driver::Device::register_ds().

Here is the caller graph for this function:

15.341.2.5 register_iface()

```
long L4virtio::Device::register_iface (
            L4::Ipc::Cap< L4::Triggerable > guest_irq,
            L4::Ipc::Out< L4::Cap< L4::Triggerable > > host_irq,
            L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > config_ds )
```

Register client to the L4-VIRTIO device.

Parameters

guest_irq	IRQ capability for valid IRQ object for device-to-driver notifications.
host_irq	Capability selector for receiving the driver-to-device notifications IRQ capability.
config_ds	Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space.

Return values

L4_EOK	Success.
-L4_ENOSYS	This interface is no longer supported by the server. Use get_device_config() etc. instead.

Deprecated Use device_config(), device_notification_irq() and Icu::bind() instead.

15.341.2.6 set_status()

```
long L4virtio::Device::set_status (
            unsigned status )
```

Write the VIRTIO status register.

Parameters

status	Status word to write to the VIRTIO status.

Return values

0	on success.

Note

All other registers are accessed via shared memory.

Referenced by L4virtio::Driver::Device::driver_acknowledge(), and L4virtio::Driver::Device::driver_connect().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/l4virtio

15.342 L4virtio::Driver::Block_device Class Reference

Simple class for accessing a virtio block device synchronously.

Inheritance diagram for L4virtio::Driver::Block_device:

Collaboration diagram for L4virtio::Driver::Block_device:

Data Structures

- class Handle

  Handle to an ongoing request.

Public Member Functions

- void setup_device (L4::Cap< L4virtio::Device > srvcap, l4_size_t usermem, void **userdata, Ptr< void > &user_devaddr, l4_uint32_t fmask0=-1U, l4_uint32_t fmask1=-1U)

  Setup a connection to a device and set up shared memory.
- l4virtio_block_config_t const & device_config () const

  Return a reference to the device configuration.
- Handle start_request (l4_uint64_t sector, l4_uint32_t type, Callback callback)

  Start the setup of a new request.
- int add_block (Handle handle, Ptr< void > addr, l4_uint32_t size)

  Add a data block to a request that has already been set up.
- int send_request (Handle handle)

  Process request asynchronously.
- int process_request (Handle handle)

  Process request synchronously.
- void process_used_queue ()

  Process and free all items in the used queue.

15.342.1 Detailed Description

Simple class for accessing a virtio block device synchronously.

Definition at line 346 of file virtio-block.

15.342.2 Member Function Documentation

15.342.2.1 add_block()

```
int L4virtio::Driver::Block_device::add_block (
            Handle handle,
            Ptr< void > addr,
            l4_uint32_t size )  [inline]
```

Add a data block to a request that has already been set up.

Parameters

handle	Handle to request previously set up with start_request().
addr	Address of data block in device address space.
size	Size of data block.

Return values

L4_OK	Block was successfully added.
-L4_EAGAIN	No descriptors available. Try again later.

Definition at line 524 of file virtio-block.

References L4virtio::Virtqueue::Desc::addr, L4virtio::Driver::Virtqueue::alloc_descriptor(), L4virtio::Driver::Virtqueue::desc(), L4virtio::Virtqueue::Desc::flags, L4_EAGAIN, L4_EOK, L4virtio::Virtqueue::Desc::len, L4virtio::Virtqueue::Desc::Flags::next(), L4virtio::Virtqueue::Desc::next, L4virtio::Virtqueue::Desc::Flags::raw, l4virtio_block_header_t::type, and L4virtio::Virtqueue::Desc::Fla

Here is the call graph for this function:

15.342.2.2 process_request()

```
int L4virtio::Driver::Block_device::process_request (
            Handle handle )  [inline]
```

Process request synchronously.

Parameters

handle	Handle to request to process.

Sends a request to the driver that was previously set up with start_request() and add_block() and wait for it to be executed.

Definition at line 595 of file virtio-block.

References L4virtio::Virtqueue::Desc::addr, L4virtio::Driver::Virtqueue::alloc_descriptor(), L4virtio::Driver::Virtqueue::desc(), L4virtio::Virtqueue::Desc::flags, L4_EINVAL, L4_EIO, L4_ENOSYS, L4_EOK, L4VIRTIO_BLOCK_S_IOERR, L4VIRTIO_BLOCK_S_OK, L4VIRTIO_BLOCK_S_UNSUPP, L4virtio::Virtqueue::Desc::len, L4virtio::Virtqueue::Desc::Flags::next(), L4virtio::Virtqueue::Desc::next, L4virtio::Virtqueue::Desc::Flags::raw, L4virtio::Driver::Device::send_and_wait(), and L4virtio::Virtqueue::Desc::Flags::write().

Here is the call graph for this function:

15.342.2.3 process_used_queue()

```
void L4virtio::Driver::Block_device::process_used_queue ( ) [inline]
```

Process and free all items in the used queue.

If the request has a callback registered it is called after the item has been removed from the queue.

Definition at line 646 of file virtio-block.

References L4Re::chksys(), L4virtio::Driver::Virtqueue::find_next_used(), L4_ENOSYS, and L4virtio::Virtqueue::num().

Here is the call graph for this function:

15.342.2.4 send_request()

```
int L4virtio::Driver::Block_device::send_request (
            Handle handle ) [inline]
```

Process request asynchronously.

Parameters

handle	Handle to request to send to the device

Return values

L4_OK	Request was successfully scheduled.
-L4_EAGAIN	No descriptors available. Try again later.

Sends a request to the driver that was previously set up with start_request() and add_block() and wait for it to be executed.

Definition at line 560 of file virtio-block.

References L4virtio::Virtqueue::Desc::addr, L4virtio::Driver::Virtqueue::alloc_descriptor(), L4virtio::Driver::Virtqueue::desc(), L4virtio::Virtqueue::Desc::flags, L4_EAGAIN, L4_EOK, L4virtio::Virtqueue::Desc::len, L4virtio::Virtqueue::Desc::Flags::next(), L4virtio::Virtqueue::Desc::next, L4virtio::Virtqueue::Desc::Flags::raw, L4virtio::Driver::Device::send(), and L4virtio::Virtqueue::Desc::Flags::write().

Here is the call graph for this function:

15.342.2.5 setup_device()

```
void L4virtio::Driver::Block_device::setup_device (
            L4::Cap< L4virtio::Device > srvcap,
            l4_size_t usermem,
            void ** userdata,
            Ptr< void > & user_devaddr,
            l4_uint32_t fmask0 = -1U,
            l4_uint32_t fmask1 = -1U )  [inline]
```

Setup a connection to a device and set up shared memory.

Parameters

srvcap	IPC capability of the channel to the server.
usermem	Size of additional memory to share with device.
userdata	Pointer to the region of user-usable memory.
user_devaddr	Address of user-usable memory in device address space.
fmask0	Feature bits 0..31 that the driver supports.
fmask1	Feature bits 32..63 that the driver supports.

This function starts a hand shake with the device and sets up the virtqueues for communication and the additional data structures for the block device. It will also allocate and share additional memory that the caller then can use freely, i.e. normally this memory would be used as a reception buffer. The caller may also decide to not make use of this convenience function and request 0 bytes in usermem. Then it has to allocate the block buffers for sending/receiving payload manually and share them using register_ds().

Definition at line 397 of file virtio-block.

References L4virtio::Virtqueue::avail_offset(), L4Re::chkcap(), L4Re::chksys(), L4virtio::Driver::Device::config_queue(), L4Re::Mem_alloc::Continuous, L4virtio::Driver::Device::driver_acknowledge(), L4virtio::Driver::Device::driver_connect(), L4Re::Env::env(), L4virtio::Driver::Virtqueue::init_queue(), L4_EINVAL, L4_ENODEV, L4_PAGESHIFT, l4_round_page(), l4_round_size(), l4util_bsr(), L4VIRTIO_ID_BLOCK, L4::Ipc::make_cap_rw(), L4virtio::Driver::Device::max_queue_size(), L4Re::Mem_alloc::Pinned, L4virtio::Driver::Device::register_ds(), L4Re::Rm::F::RW, L4Re::Rm::F::Search_addr, L4virtio::Virtqueue::total_size(), and L4virtio::Virtqueue::used_offset().

Here is the call graph for this function:

15.342.2.6 start_request()

```
Handle L4virtio::Driver::Block_device::start_request (
            l4_uint64_t sector,
            l4_uint32_t type,
            Callback callback )  [inline]
```

Start the setup of a new request.

Parameters

sector	First sector to write to/read from.
type	Request type.
callback	Function to call, when the request is finished. May be 0 for synchronous requests.

Definition at line 486 of file virtio-block.

References L4virtio::Virtqueue::Desc::addr, L4virtio::Driver::Virtqueue::alloc_descriptor(), L4virtio::Driver::Virtqueue::desc(), L4virtio::Virtqueue::Desc::flags, l4virtio_block_header_t::ioprio, L4virtio::Virtqueue::Desc::len, L4virtio::Virtqueue::Desc::Flags::raw, l4virtio_block_header_t::sector, and l4virtio_block_header_t::type.

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/client/virtio-block

15.343 L4virtio::Driver::Block_device::Handle Class Reference

Handle to an ongoing request.

Collaboration diagram for L4virtio::Driver::Block_device::Handle:

15.343.1 Detailed Description

Handle to an ongoing request.

Definition at line 366 of file virtio-block.

The documentation for this class was generated from the following file:

- l4/l4virtio/client/virtio-block

15.344 L4virtio::Driver::Device Class Reference

Client-side implementation for a general virtio device.

Inheritance diagram for L4virtio::Driver::Device:

Collaboration diagram for L4virtio::Driver::Device:

Public Member Functions

- void driver_connect (L4::Cap< L4virtio::Device > srvcap, bool manage_notify=true)

  Contacts the device and starts the initial handshake.
- int bind_notification_irq (unsigned index, L4::Cap< L4::Triggerable > irq) const

  Register a triggerable to receive notifications from the device.
- bool fail_state () const

Return true if the device is in a fail state.

- int driver_acknowledge ()

  Finalize handshake with the device.
- int register_ds (L4::Cap< L4Re::Dataspace > ds, l4_umword_t offset, l4_umword_t size, l4_uint64_t *devaddr)

  Share a dataspace with the device.
- int config_queue (int num, unsigned size, l4_uint64_t desc_addr, l4_uint64_t avail_addr, l4_uint64_t used_addr)

  Send the virtqueue configuration to the device.
- int max_queue_size (int num) const

  Maximum queue size allowed by the device.
- int send_and_wait (Virtqueue &queue, l4_uint16_t descno)

  Send a request to the device and wait for it to be processed.
- int wait (int index) const

  Wait for a notification from the device.
- int wait_for_next_used (Virtqueue &queue) const

  Wait for the next item to arrive in the used queue and return it.
- void send (Virtqueue &queue, l4_uint16_t descno)

  Send a request to the device.

15.344.1 Detailed Description

Client-side implementation for a general virtio device.

Definition at line 35 of file virtio-block.

15.344.2 Member Function Documentation

15.344.2.1 bind_notification_irq()

```
int L4virtio::Driver::Device::bind_notification_irq (
            unsigned index,
            L4::Cap< L4::Triggerable > irq ) const  [inline]
```

Register a triggerable to receive notifications from the device.

Parameters

	index	Index of the interrupt.
out	*irq*	Triggerable to register for notifications.

Definition at line 133 of file virtio-block.

References L4::Icu::bind(), and l4_error().

Here is the call graph for this function:

15.344.2.2 config_queue()

```
int L4virtio::Driver::Device::config_queue (
            int num,
            unsigned size,
            l4_uint64_t desc_addr,
            l4_uint64_t avail_addr,
            l4_uint64_t used_addr )  [inline]
```

Send the virtqueue configuration to the device.

Parameters

num	Number of queue to configure.
size	Size of rings in the queue, must be a power of 2)
desc_addr	Address of descriptor table (device address)
avail_addr	Address of available ring (device address)
used_addr	Address of used ring (device address)

Definition at line 203 of file virtio-block.

References L4virtio::Device::config_queue().

Referenced by L4virtio::Driver::Block_device::setup_device().

Here is the call graph for this function:

Here is the caller graph for this function:

15.344.2.3 driver_acknowledge()

```
int L4virtio::Driver::Device::driver_acknowledge ( ) [inline]
```

Finalize handshake with the device.

Must be called after all queues have been set up and before the first request is sent. It is still possible to add more shared dataspaces after the handshake has been finished.

Definition at line 147 of file virtio-block.

References L4Re::chksys(), L4_EINVAL, L4_EIO, L4_ENODEV, L4_EOK, L4VIRTIO_FEATURE_VERSION_1, l4virtio_get_feature(), L4VIRTIO_STATUS_DRIVER_OK, L4VIRTIO_STATUS_FEATURES_OK, and L4virtio::Device::set_status().

Referenced by L4virtio::Driver::Block_device::setup_device().

Here is the call graph for this function:

Here is the caller graph for this function:

15.344.2.4 driver_connect()

```
void L4virtio::Driver::Device::driver_connect (
            L4::Cap< L4virtio::Device > srvcap,
            bool manage_notify = true )  [inline]
```

Contacts the device and starts the initial handshake.

Parameters

srvcap	Capability for device communication.
manage_notify	Set up a semaphore for notifications from the device. See below.

Exceptions

L4::Runtime_error	if the initialisation fails

This function contacts the server, sets up the notification channels and the configuration dataspace. After this is done, the caller can set up any dataspaces it needs. The initialisation then needs to be finished by calling driver_acknowledge().

Per default this function creates and registers a semaphore for receiving notification from the device. This semaphore is used in the blocking functions send_and_wait(), wait() and next_used().

When `manage_notify` is false, then the caller may manually register and handle notification interrupts from the device. This is for example useful, when the client runs in an application with a server loop.

Definition at line 60 of file virtio-block.

References L4::Icu::bind(), L4Re::chkcap(), L4Re::chksys(), L4virtio::Device::device_config(), L4virtio::Device::device_notification_irq(
L4Re::Env::env(), L4_EINVAL, L4_EIO, L4_ENODEV, l4_error(), L4_PAGEMASK, L4_PAGESHIFT, L4_PAGESIZE, L4_SUPERPAGESIZE, L4VIRTIO_STATUS_ACKNOWLEDGE, L4VIRTIO_STATUS_DRIVER, L4::Ipc::make_cap_rw(), L4Re::Rm::F::RW, L4Re::Rm::F::Search_addr, and L4virtio::Device::set_status().

Referenced by L4virtio::Driver::Block_device::setup_device().

Here is the call graph for this function:

Here is the caller graph for this function:

15.344.2.5 max_queue_size()

```
int L4virtio::Driver::Device::max_queue_size (
            int num ) const  [inline]
```

Maximum queue size allowed by the device.

Parameters

num	Number of queue for which to determine the maximum size.

Definition at line 221 of file virtio-block.

Referenced by L4virtio::Driver::Block_device::setup_device().

Here is the caller graph for this function:

15.344.2.6 register_ds()

```
int L4virtio::Driver::Device::register_ds (
            L4::Cap< L4Re::Dataspace > ds,
            l4_umword_t offset,
            l4_umword_t size,
            l4_uint64_t * devaddr )  [inline]
```

Share a dataspace with the device.

Parameters

ds	Dataspace to share with the device.
offset	Offset in dataspace where the shared part starts.
size	Total size in bytes of the shared space.
devaddr	Start of shared space in the device address space.

Although this function allows to share only a part of the given dataspace for convenience, the granularity of sharing is always the dataspace level. Thus, the remainder of the dataspace is not protected from the device.

When communicating with the device, addresses must be given with respect to the device address space. This is not the same as the virtual address space of the client in order to not leak information about the address space layout.

Definition at line 187 of file virtio-block.

References L4::Ipc::make_cap_rw(), and L4virtio::Device::register_ds().

Referenced by L4virtio::Driver::Block_device::setup_device().

Here is the call graph for this function:

Here is the caller graph for this function:

15.344.2.7 send()

```
void L4virtio::Driver::Device::send (
            Virtqueue & queue,
            l4_uint16_t descno )  [inline]
```

Send a request to the device.

Parameters

queue	Queue that contains the request in its descriptor table
descno	Index of first entry in descriptor table where

Definition at line 298 of file virtio-block.

References L4virtio::Driver::Virtqueue::enqueue_descriptor(), and L4virtio::Virtqueue::no_notify_host().

Referenced by send_and_wait(), and L4virtio::Driver::Block_device::send_request().

Here is the call graph for this function:

Here is the caller graph for this function:

15.344.2.8 send_and_wait()

```
int L4virtio::Driver::Device::send_and_wait (
            Virtqueue & queue,
            l4_uint16_t descno )  [inline]
```

Send a request to the device and wait for it to be processed.

Parameters

queue	Queue that contains the request in its descriptor table
descno	Index of first entry in descriptor table where

This function provides a simple mechanism to send requests synchronously. It must not be used with other requests at the same time as it directly waits for a notification on the device irq cap.

Precondition

driver_connect() was called with manage_notify.

Definition at line 238 of file virtio-block.

References L4_EINVAL, L4_EOK, send(), and wait_for_next_used().

Referenced by L4virtio::Driver::Block_device::process_request().

Here is the call graph for this function:

Here is the caller graph for this function:

15.344.2.9 wait()

```
int L4virtio::Driver::Device::wait (
            int index ) const  [inline]
```

Wait for a notification from the device.

Parameters

index	Notification slot to wait for.

Precondition

driver_connect() was called with manage_notify.

Definition at line 259 of file virtio-block.

References L4_EEXIST, l4_ipc_error(), and l4_utcb().

Referenced by wait_for_next_used().

Here is the call graph for this function:

Here is the caller graph for this function:

15.344.2.10 wait_for_next_used()

```
int L4virtio::Driver::Device::wait_for_next_used (
            Virtqueue & queue ) const  [inline]
```

Wait for the next item to arrive in the used queue and return it.

Return values

>=0	Descriptor number of item removed from used queue.
<0	IPC error while waiting for notification.

The call blocks until the next item is available in the used queue.

Precondition

driver_connect() was called with manage_notify.

Definition at line 277 of file virtio-block.

References L4virtio::Driver::Virtqueue::find_next_used(), and wait().

Referenced by send_and_wait().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/client/virtio-block

15.345 L4virtio::Driver::Virtqueue Class Reference

Driver-side implementation of a Virtqueue.

Inheritance diagram for L4virtio::Driver::Virtqueue:

Collaboration diagram for L4virtio::Driver::Virtqueue:

Public Member Functions

- void initialize_rings (unsigned num)

  Initialize the descriptor table and the index structures of this queue.
- void init_queue (unsigned num, void ∗desc, void ∗avail, void ∗used)

  Initialize this virtqueue.
- void init_queue (unsigned num, void ∗base)

Initialize this virtqueue.

- l4_uint16_t alloc_descriptor ()

  Allocate and return an unused descriptor from the descriptor table.
- void enqueue_descriptor (l4_uint16_t descno)

  Enqueue a descriptor in the available ring.
- Desc & desc (l4_uint16_t descno)

  Return a reference to a descriptor in the descriptor table.
- l4_uint16_t find_next_used (l4_uint32_t *len=nullptr)

  Return the next finished block.
- void free_descriptor (l4_uint16_t head, l4_uint16_t tail)

  Free a chained list of descriptors in the descriptor queue.

Additional Inherited Members

15.345.1 Detailed Description

Driver-side implementation of a Virtqueue.

Adds function for managing the descriptor list, enqueueing new and dequeueing finished requests.

Note

The Virtqueue implementation is not thread-safe.

Definition at line 475 of file virtqueue.

15.345.2 Member Function Documentation

15.345.2.1 alloc_descriptor()

```
l4_uint16_t L4virtio::Driver::Virtqueue::alloc_descriptor ( ) [inline]
```

Allocate and return an unused descriptor from the descriptor table.

The descriptor will be removed from the free list, the content should be considered undefined. After use, it needs to be freed using free_descriptor().

Returns

The index of the reserved descriptor or Virtqueue::Eoq if no free descriptor is available.

Note: the implementation uses (2^16 - 1) as the end of queue marker. That means that the final entry in the queue can not be allocated iff the queue size is 2^16.

Definition at line 563 of file virtqueue.

References L4virtio::Virtqueue::_desc, and L4virtio::Virtqueue::Desc::next.

Referenced by L4virtio::Driver::Block_device::add_block(), L4virtio::Driver::Block_device::process_request(), L4virtio::Driver::Block_device::send_request(), and L4virtio::Driver::Block_device::start_request().

Here is the caller graph for this function:

15.345.2.2 desc()

```
Desc& L4virtio::Driver::Virtqueue::desc (
            l4_uint16_t descno )  [inline]
```

Return a reference to a descriptor in the descriptor table.

Parameters

descno	Index of the descriptor, expected to be in correct range.

Definition at line 595 of file virtqueue.

References L4virtio::Virtqueue::_desc, and L4virtio::Virtqueue::_idx_mask.

Referenced by L4virtio::Driver::Block_device::add_block(), init_queue(), L4virtio::Driver::Block_device::process_request(), L4virtio::Driver::Block_device::send_request(), and L4virtio::Driver::Block_device::start_request().

Here is the caller graph for this function:

15.345.2.3 enqueue_descriptor()

```
void L4virtio::Driver::Virtqueue::enqueue_descriptor (
            l4_uint16_t descno ) [inline]
```

Enqueue a descriptor in the available ring.

Parameters

descno	Index of the head descriptor to enqueue.

Definition at line 579 of file virtqueue.

References L4virtio::Virtqueue::_avail, L4virtio::Virtqueue::_idx_mask, L4virtio::Virtqueue::Avail::idx, and L4virtio::Virtqueue::Avail::ring.

Referenced by L4virtio::Driver::Device::send().

Here is the caller graph for this function:

15.345.2.4 find_next_used()

```
l4_uint16_t L4virtio::Driver::Virtqueue::find_next_used (
            l4_uint32_t * len = nullptr )  [inline]
```

Return the next finished block.

Parameters

out	*len*	(optional) Size of valid data in finished block. Note that this is the value reported by the device, which may set it to a value that is larger than the original buffer size.

Returns

Index of the head or Virtqueue::Eoq if no used element is currently available.

Definition at line 614 of file virtqueue.

References L4virtio::Virtqueue::_current_avail, L4virtio::Virtqueue::_idx_mask, L4virtio::Virtqueue::_used, L4virtio::Virtqueue::Used::idx, L4virtio::Virtqueue::Used_elem::len, and L4virtio::Virtqueue::Used::ring.

Referenced by L4virtio::Driver::Block_device::process_used_queue(), and L4virtio::Driver::Device::wait_for_next_used().

Here is the caller graph for this function:

15.345.2.5 free_descriptor()

```
void L4virtio::Driver::Virtqueue::free_descriptor (
            l4_uint16_t head,
            l4_uint16_t tail )  [inline]
```

Free a chained list of descriptors in the descriptor queue.

Parameters

head	Index of the first element in the descriptor chain.
tail	Index of the last element in the descriptor chain.

Simply takes the descriptor chain and prepends it to the beginning of the free list. Assumes that the list has been correctly chained.

Definition at line 636 of file virtqueue.

References L4virtio::Virtqueue::_desc, L4virtio::Virtqueue::_idx_mask, and L4virtio::Virtqueue::Desc::next.

15.345.2.6 init_queue() [1/2]

```
void L4virtio::Driver::Virtqueue::init_queue (
            unsigned num,
            void * base )  [inline]
```

Initialize this virtqueue.

Parameters

num	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
base	The base address for the queue data structure.

This function sets up the memory and initializes the freelist.

Definition at line 542 of file virtqueue.

References initialize_rings(), L4virtio::Virtqueue::num(), and L4virtio::Virtqueue::setup_simple().

Here is the call graph for this function:

15.345.2.7 init_queue() [2/2]

```
void L4virtio::Driver::Virtqueue::init_queue (
            unsigned num,
            void * desc,
            void * avail,
            void * used )  [inline]
```

Initialize this virtqueue.

Parameters

num	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
desc	The address of the descriptor table. (Must be Desc_align aligned and at least `desc_size(num)` bytes in size.)
avail	The address of the available ring. (Must be Avail_align aligned and at least `avail_size(num)` bytes in size.)
used	The address of the used ring. (Must be Used_align aligned and at least `used_size(num)` bytes in size.)

This function sets up the memory and initializes the freelist.

Definition at line 527 of file virtqueue.

References desc(), initialize_rings(), L4virtio::Virtqueue::num(), and L4virtio::Virtqueue::setup().

Referenced by L4virtio::Driver::Block_device::setup_device().

Here is the call graph for this function:

Here is the caller graph for this function:

15.345.2.8 initialize_rings()

```
void L4virtio::Driver::Virtqueue::initialize_rings (
            unsigned num )  [inline]
```

Initialize the descriptor table and the index structures of this queue.

Parameters

num	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).

Precondition

The queue must be set up correctly with setup() or setup_simple().

Definition at line 499 of file virtqueue.

References L4virtio::Virtqueue::_avail, L4virtio::Virtqueue::_desc, L4virtio::Virtqueue::_used, L4virtio::Virtqueue::Avail::idx, L4virtio::Virtqueue::Used::idx, L4virtio::Virtqueue::Desc::next, and L4virtio::Virtqueue::num().

Referenced by init_queue().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

15.346 L4virtio::Ptr< T > Class Template Reference

Pointer used in virtio descriptors.

Inheritance diagram for L4virtio::Ptr< T >:

Collaboration diagram for L4virtio::Ptr< T >:

Public Types

- enum Invalid_type { Invalid }

  Type for making an invalid (NULL) Ptr.

Public Member Functions

- Ptr (Invalid_type)

  Make and invalid Ptr.
- Ptr (l4_uint64_t vm_addr)

  Make a Ptr from a raw 64bit address.
- l4_uint64_t get () const
- bool is_valid () const

15.346.1 Detailed Description

template<typename T>
class L4virtio::Ptr< T >

Pointer used in virtio descriptors.

As the descriptor contain guest addresses these pointers cannot be dereferenced directly.

Definition at line 56 of file virtqueue.

15.346.2 Member Enumeration Documentation

15.346.2.1 Invalid_type

```
template<typename T >
enum L4virtio::Ptr::Invalid_type
```

Type for making an invalid (NULL) Ptr.

Enumerator

Invalid	Use to set a Ptr to invalid (NULL)

Definition at line 60 of file virtqueue.

15.346.3 Member Function Documentation

15.346.3.1 get()

```
template<typename T >
l4_uint64_t L4virtio::Ptr< T >::get ( ) const  [inline]
```

Returns

The raw 64bit address of the pointer.

Definition at line 71 of file virtqueue.

Referenced by L4virtio::Virtqueue::Desc::dump(), L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc(), and L4virtio::Svr::Driver_mem_region_t< DATA >::local().

Here is the caller graph for this function:

15.346.3.2 is_valid()

```
template<typename T >
bool L4virtio::Ptr< T >::is_valid ( ) const  [inline]
```

Returns

true if the pointer is invalid (NULL).

Definition at line 74 of file virtqueue.

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

15.347 L4virtio::Svr::Bad_descriptor Struct Reference

Exception used by Queue to indicate descriptor errors.

Collaboration diagram for L4virtio::Svr::Bad_descriptor:

Public Types

- enum Error {
  Bad_address , Bad_rights , Bad_flags , Bad_next ,
  Bad_size }

  The error code.

Public Member Functions

- Bad_descriptor (Request_processor const ∗proc, Error e)

  Make a bad descriptor exception.
- char const ∗ message () const

  Get a human readable description of the error code.

Data Fields

- Request_processor const ∗ proc

  The processor that triggered the exception.

15.347.1 Detailed Description

Exception used by Queue to indicate descriptor errors.

Definition at line 325 of file virtio.

15.347.2 Member Enumeration Documentation

15.347.2.1 Error

```
enum L4virtio::Svr::Bad_descriptor::Error
```

The error code.

Enumerator

Bad_address	Address cannot be translated.
Bad_rights	Missing access rights on memory.
Bad_flags	Invalid combination of descriptor flags.
Bad_next	Invalid next index.
Bad_size	Invalid size of memory block.

Definition at line 328 of file virtio.

15.347.3 Constructor & Destructor Documentation

15.347.3.1 Bad_descriptor()

```
L4virtio::Svr::Bad_descriptor::Bad_descriptor (
            Request_processor const * proc,
            Error e )  [inline]
```

Make a bad descriptor exception.

Parameters

proc	The request processor causing the exception
e	The error code.

Definition at line 348 of file virtio.

15.347.4 Member Function Documentation

15.347.4.1 message()

```
char const* L4virtio::Svr::Bad_descriptor::message ( ) const  [inline]
```

Get a human readable description of the error code.

Returns

Message describing the error.

Definition at line 357 of file virtio.

References Bad_address, Bad_flags, Bad_next, Bad_rights, and Bad_size.

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

15.348 L4virtio::Svr::Block_dev_base< Ds_data > Class Template Reference

Base class for virtio block devices.

Inheritance diagram for L4virtio::Svr::Block_dev_base< Ds_data >:

Collaboration diagram for L4virtio::Svr::Block_dev_base< Ds_data >:

Public Member Functions

- Block_dev_base (l4_uint32_t vendor, unsigned queue_size, l4_uint64_t capacity, bool read_only)

  Create a new virtio block device.
- virtual bool process_request (cxx::unique_ptr< Request > &&req)=0

  Implements the actual processing of data in the device.
- virtual void reset_device ()=0

Reset the actual hardware device.

- virtual bool queue_stopped ()=0

  Return true, if the queues should not be processed further.
- void finalize_request (cxx::unique_ptr< Request > req, unsigned sz, l4_uint8_t status=L4VIRTIO_BLOCK_S_OK)

  Releases resources related to a request and notifies the client.
- int reconfig_queue (unsigned idx) override

  callback for client queue-config request
- void reset () override

  reset callback, called for doing a device reset

Protected Member Functions

- void set_size_max (l4_uint32_t sz)

  Sets the maximum size of any single segment reported to client.
- void set_seg_max (l4_uint32_t sz)

  Sets the maximum number of segments in a request that is reported to client.
- void set_geometry (l4_uint16_t cylinders, l4_uint8_t heads, l4_uint8_t sectors)

  Set disk geometry that is reported to the client.
- void set_blk_size (l4_uint32_t sz)

  Sets block disk size to be reported to the client.
- void set_topology (l4_uint8_t physical_block_exp, l4_uint8_t alignment_offset, l4_uint32_t min_io_size, l4_uint32_t opt_io_size)

  Sets the I/O alignment information reported back to the client.
- void set_flush ()

  Enables the flush command.
- void set_config_wce (l4_uint8_t writeback)

  Sets cache mode and enables the writeback toggle.
- l4_uint8_t get_writeback ()

  Get the writeback field from the configuration space.
- void set_discard (l4_uint32_t max_discard_sectors, l4_uint32_t max_discard_seg, l4_uint32_t discard_↩ sector_alignment)

  Sets constraints for and enables the discard command.
- void set_write_zeroes (l4_uint32_t max_write_zeroes_sectors, l4_uint32_t max_write_zeroes_seg, l4_uint8_t write_zeroes_may_unmap)

  Sets constraints for and enables the write zeroes command.

Additional Inherited Members

15.348.1 Detailed Description

template<typename Ds_data>
class L4virtio::Svr::Block_dev_base< Ds_data >

Base class for virtio block devices.

Use this class as a base to implement your own specific block device.

Definition at line 257 of file virtio-block.

15.348.2 Constructor & Destructor Documentation

15.348.2.1 Block_dev_base()

```
template<typename Ds_data >
L4virtio::Svr::Block_dev_base< Ds_data >::Block_dev_base (
            l4_uint32_t vendor,
            unsigned queue_size,
            l4_uint64_t capacity,
            bool read_only )  [inline]
```

Create a new virtio block device.

Parameters

vendor	Vendor ID
queue_size	Number of entries to provide in avail and used queue.
capacity	Size of the device in 512-byte sectors.
read_only	True, if the device should not be writable.

Definition at line 462 of file virtio-block.

References L4virtio::Svr::Device_t< Ds_data >::reset_queue_config().

Here is the call graph for this function:

15.348.3 Member Function Documentation

15.348.3.1 finalize_request()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::finalize_request (
            cxx::unique_ptr< Request > req,
            unsigned sz,
            l4_uint8_t status = L4VIRTIO_BLOCK_S_OK )  [inline]
```

Releases resources related to a request and notifies the client.

Parameters

req	Pointer to request that has finished.
sz	Number of bytes consumed.
status	Status of request (see L4virtio_block_status).

This function must be called when an asynchronous request finishes, either successfully or with an error. The status byte in the request must have been set prior to calling it.

Definition at line 515 of file virtio-block.

References L4virtio::Svr::Device_t< Ds_data >::device_error(), and L4virtio::Virtqueue::ready().

Here is the call graph for this function:

15.348.3.2 get_writeback()

```
template<typename Ds_data >
l4_uint8_t L4virtio::Svr::Block_dev_base< Ds_data >::get_writeback ( ) [inline], [protected]
```

Get the writeback field from the configuration space.

Returns

Value of the writeback field.

Definition at line 406 of file virtio-block.

15.348.3.3 process_request()

```
template<typename Ds_data >
virtual bool L4virtio::Svr::Block_dev_base< Ds_data >::process_request (
            cxx::unique_ptr< Request > && req ) [pure virtual]
```

Implements the actual processing of data in the device.

Parameters

req	The request to be processed.

Returns

If false, no further requests will be scheduled.

Synchronous and asynchronous processing of the data is supported. For asynchronous mode, the function should set up the worker and then return false. In synchronous mode, the function should return true, once processing is complete. If there is an error and processing is aborted, the status flag of `req` needs to be set accordingly and the request immediately finished with finish_request() if the client is to be answered.

15.348.3.4 set_blk_size()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_blk_size (
            l4_uint32_t sz )  [inline], [protected]
```

Sets block disk size to be reported to the client.

Setting this does not change the logical sector size used for addressing the device.

Definition at line 350 of file virtio-block.

15.348.3.5 set_config_wce()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_config_wce (
            l4_uint8_t writeback )  [inline], [protected]
```

Sets cache mode and enables the writeback toggle.

Parameters

writeback	Mode of the cache (0 for writethrough, 1 for writeback).

Definition at line 393 of file virtio-block.

15.348.3.6 set_discard()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_discard (
            l4_uint32_t max_discard_sectors,
```

```
l4_uint32_t max_discard_seg,
l4_uint32_t discard_sector_alignment )  [inline], [protected]
```

Sets constraints for and enables the discard command.

Parameters

max_discard_sectors	Maximum discard sectors size.
max_discard_seg	Maximum discard segment number.
discard_sector_alignment	Can be used by the driver when splitting a request based on alignment.

Definition at line 420 of file virtio-block.

15.348.3.7 set_size_max()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_size_max (
            l4_uint32_t sz )  [inline], [protected]
```

Sets the maximum size of any single segment reported to client.

The limit is also applied to any incoming requests. Requests with larger segments result in an IO error being reported to the client. That means that process_request() can safely make the assumption that all segments in the received request are smaller.

Definition at line 308 of file virtio-block.

15.348.3.8 set_topology()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_topology (
            l4_uint8_t physical_block_exp,
            l4_uint8_t alignment_offset,
            l4_uint32_t min_io_size,
            l4_uint32_t opt_io_size )  [inline], [protected]
```

Sets the I/O alignment information reported back to the client.

Parameters

physical_block_exp	Number of logical blocks per physical block(log2)
alignment_offset	Offset of the first aligned logical block
min_io_size	Suggested minimum I/O size in blocks
opt_io_size	Optimal I/O size in blocks

Definition at line 366 of file virtio-block.

15.348.3.9 set_write_zeroes()

```
template<typename Ds_data >
```

```
void L4virtio::Svr::Block_dev_base< Ds_data >::set_write_zeroes (
            l4_uint32_t max_write_zeroes_sectors,
            l4_uint32_t max_write_zeroes_seg,
            l4_uint8_t write_zeroes_may_unmap )  [inline], [protected]
```

Sets constraints for and enables the write zeroes command.

Parameters

max_write_zeroes_sectors	Maximum write zeroes sectors size.
max_write_zeroes_seg	maximum write zeroes segment number.
write_zeroes_may_unmap	Set if a write zeroes request can result in deallocating one or more sectors.

Definition at line 440 of file virtio-block.

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio-block

15.349 L4virtio::Svr::Block_request< Ds_data > Class Template Reference

A request to read or write data.

Collaboration diagram for L4virtio::Svr::Block_request< Ds_data >:

Public Member Functions

- unsigned data_size () const

  Compute the total size of the data in the request.
- bool has_more ()

  Check if the request contains more data blocks.
- Data_block next_block ()

  Return next block in scatter-gather list.
- l4virtio_block_header_t const & header () const

  Return the block request header.

15.349.1 Detailed Description

template<typename Ds_data>
class L4virtio::Svr::Block_request< Ds_data >

A request to read or write data.

Definition at line 28 of file virtio-block.

15.349.2 Member Function Documentation

15.349.2.1 data_size()

```
template<typename Ds_data >
unsigned L4virtio::Svr::Block_request< Ds_data >::data_size ( ) const  [inline]
```

Compute the total size of the data in the request.

Return values

Size	in bytes or 0 if there was an error.

Exceptions

L4::Runtime_error(-L4_EIO)	Request has a bad format.

Note that this operation is relatively expensive as it has to iterate over the complete list of blocks.

Definition at line 63 of file virtio-block.

References L4virtio::Svr::Request_processor::has_more(), L4_EIO, L4virtio::Svr::Request_processor::next(), and L4virtio::Svr::Request_processor::start().

Here is the call graph for this function:

15.349.2.2 next_block()

```
template<typename Ds_data >
Data_block L4virtio::Svr::Block_request< Ds_data >::next_block ( ) [inline]
```

Return next block in scatter-gather list.

Returns

Information about the next data block.

Exceptions

L4::Runtime_error	No more data block is available.
Bad_descriptor	Virtio request is corrupted.

Definition at line 113 of file virtio-block.

References L4virtio::Svr::Bad_descriptor::Bad_size, L4virtio::Svr::Request_processor::has_more(), L4_EEXIST, and L4virtio::Svr::Request_processor::next().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio-block

15.350 L4virtio::Svr::Data_buffer Struct Reference

Abstract data buffer.

Collaboration diagram for L4virtio::Svr::Data_buffer:

Public Member Functions

- template<typename T >
  Data_buffer (T *p)

  Create buffer for object p.
- template<typename T >
  void set (T *p)

  Set buffer for object p.
- l4_uint32_t copy_to (Data_buffer *dst, l4_uint32_t max=UINT_MAX)

  Copy contents from this buffer to the destination buffer.
- l4_uint32_t skip (l4_uint32_t bytes)

  Skip given number of bytes in this buffer.
- bool done () const

  Check if there are no more bytes left in the buffer.

Data Fields

- char * pos

  Current buffer position.
- l4_uint32_t left

  Bytes left in buffer.

15.350.1 Detailed Description

Abstract data buffer.

Definition at line 239 of file virtio.

15.350.2 Constructor & Destructor Documentation

15.350.2.1 Data_buffer()

```
template<typename T >
L4virtio::Svr::Data_buffer::Data_buffer (
            T * p )  [inline], [explicit]
```

Create buffer for object *p*.

Template Parameters

T	type of object (implicit)

Parameters

p	pointer to object.

The buffer shall point to the start of the object *p* and the size left is sizeof(T).

Definition at line 255 of file virtio.

15.350.3 Member Function Documentation

15.350.3.1 copy_to()

```
l4_uint32_t L4virtio::Svr::Data_buffer::copy_to (
            Data_buffer * dst,
            l4_uint32_t max = UINT_MAX )  [inline]
```

Copy contents from this buffer to the destination buffer.

Parameters

dst	Destination buffer.
max	(optional) Maximum number of bytes to copy.

Returns

the number of bytes copied.

This function copies at most `max` bytes from this to `dst`. If `max` is omitted, copies the maximum number of bytes available that fit `dst`.

Definition at line 284 of file virtio.

References left, cxx::max(), cxx::min(), and pos.

Here is the call graph for this function:

15.350.3.2 done()

```
bool L4virtio::Svr::Data_buffer::done ( ) const  [inline]
```

Check if there are no more bytes left in the buffer.

Returns

true if there are no more bytes left in the buffer.

Definition at line 316 of file virtio.

References left.

15.350.3.3 set()

```
template<typename T >
void L4virtio::Svr::Data_buffer::set (
            T * p )  [inline]
```

Set buffer for object *p*.

Template Parameters

T	type of object (implicit)

Parameters

p	pointer to object.

The buffer shall point to the start of the object *p* and the size left is sizeof(T).

Definition at line 268 of file virtio.

References left, and pos.

15.350.3.4 skip()

```
l4_uint32_t L4virtio::Svr::Data_buffer::skip (
            l4_uint32_t bytes ) [inline]
```

Skip given number of bytes in this buffer.

Parameters

bytes	Number of bytes that shall be skipped.

Returns

The number of bytes skipped.

Try to skip the given number of bytes in this buffer, if there are less bytes left in the buffer that given then at most left bytes are skipped and the amount is returned.

Definition at line 304 of file virtio.

References left, cxx::min(), and pos.

Here is the call graph for this function:

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

15.351 L4virtio::Svr::Dev_config Class Reference

Abstraction for L4-Virtio device config memory.

Inherited by L4virtio::Svr::Dev_config_t< l4virtio_block_config_t >, and L4virtio::Svr::Dev_config_t< PRIV_CONFIG >.

Collaboration diagram for L4virtio::Svr::Dev_config:

Public Member Functions

- Dev_config (l4_uint32_t vendor, l4_uint32_t device, unsigned cfg_size, l4_uint32_t num_queues=0)

  Create a L4-Virtio config data space.
- Dev_config (Cfg_cap const &cfg, l4_addr_t cfg_offset, l4_uint32_t vendor, l4_uint32_t device, unsigned cfg_size, l4_uint32_t num_queues=0)

  Setup an L4-Virtio config space in an existing data space.
- l4_uint32_t guest_features (unsigned idx) const

  Return a specific set of guest features.
- l4_uint32_t negotiated_features (unsigned idx) const

  Compute a specific set of negotiated features.
- Status status () const

  Get current device status (trusted).
- l4_uint32_t get_cmd () const

  Get the value from the `cmd` register.
- void reset_cmd ()

  Reset the `cmd` register after execution of a command.
- void set_status (Status status)

  Set device status register.
- void set_device_needs_reset ()

Set DEVICE_NEEDS_RESET bit in device status register.

- bool change_queue_config (l4_uint32_t num_queues)

  Setup new queue configuration.
- l4virtio_config_queue_t volatile const ∗ qconfig (unsigned index) const

  Get queue read-only config data for queue with the given index.
- void reset_hdr (bool inc_generation=false) const

  Reset the config header to the initial contents.
- bool reset_queue (unsigned index, unsigned num_max, bool inc_generation=false) const

  Reset queue config for the given queue.
- l4virtio_config_hdr_t const volatile ∗ hdr () const

  Get a read-only pointer to the config header.
- L4::Cap< L4Re::Dataspace > ds () const

  Get data-space capability for the shared config data space.
- l4_addr_t ds_offset () const

  Return the offset into the config dataspace where the device configuration starts.

15.351.1 Detailed Description

Abstraction for L4-Virtio device config memory.

Virtio defines a device configuration mechanism, L4-Virtio implements this mechanism based on shared memory a set_status() and a config_queue() call. This class provides an abstraction for L4-Virtio host implementations to establish such a shared memory data space and providing the necessary contents and access functions.

Definition at line 48 of file l4virtio.

15.351.2 Constructor & Destructor Documentation

15.351.2.1 Dev_config() [1/2]

```
L4virtio::Svr::Dev_config::Dev_config (
            l4_uint32_t vendor,
            l4_uint32_t device,
            unsigned cfg_size,
            l4_uint32_t num_queues = 0 )  [inline]
```

Create a L4-Virtio config data space.

Parameters

vendor	The vendor ID to store in config header.
device	The device ID to store in config header.
cfg_size	The size of the device-specific config data in bytes.
num_queues	The number of queues provided by the device.

This constructor allocates a data space used for L4-virtio config attaches the data space to the local address space

and writes the initial contents to the config header.

Definition at line 108 of file l4virtio.

References L4Re::chkcap(), L4Re::chksys(), L4Re::Env::env(), and L4_PAGESIZE.

Here is the call graph for this function:

15.351.2.2 Dev_config() [2/2]

```
L4virtio::Svr::Dev_config::Dev_config (
            Cfg_cap const & cfg,
            l4_addr_t cfg_offset,
            l4_uint32_t vendor,
            l4_uint32_t device,
            unsigned cfg_size,
            l4_uint32_t num_queues = 0 )  [inline]
```

Setup an L4-Virtio config space in an existing data space.

Parameters

cfg	Dataspace that should hold the L4-Virtio configuration.
cfg_offset	Offset into the dataspace where the configuration starts.
vendor	The vendor ID to store in config header.
device	The device ID to store in config header.
cfg_size	The size of the device-specific config data in bytes.
num_queues	The number of queues provided by the device.

Definition at line 142 of file l4virtio.

References L4_PAGESIZE.

15.351.3 Member Function Documentation

15.351.3.1 change_queue_config()

```
bool L4virtio::Svr::Dev_config::change_queue_config (
            l4_uint32_t num_queues ) [inline]
```

Setup new queue configuration.

Parameters

num_queues	The number of queues provided by the device.

Definition at line 265 of file l4virtio.

References L4_PAGESIZE, and reset_hdr().

Here is the call graph for this function:

15.351.3.2 ds()

```
L4::Cap<L4Re::Dataspace> L4virtio::Svr::Dev_config::ds ( ) const [inline]
```

Get data-space capability for the shared config data space.

Returns

Capability for the shared config data space.

Definition at line 354 of file l4virtio.

15.351.3.3 get_cmd()

```
l4_uint32_t L4virtio::Svr::Dev_config::get_cmd ( ) const  [inline]
```

Get the value from the `cmd` register.

Note, the most significant eight bits are the command (0 is nothing to do). The upper eight bit are reset to zero after the command was handled.

Definition at line 220 of file l4virtio.

References l4virtio_config_hdr_t::cmd, and hdr().

Referenced by L4virtio::Svr::Device_t< DATA >::handle_mem_cmd_write().

Here is the call graph for this function:

Here is the caller graph for this function:

15.351.3.4 guest_features()

```
l4_uint32_t L4virtio::Svr::Dev_config::guest_features (
            unsigned idx ) const  [inline]
```

Return a specific set of guest features.

Parameters

idx	Index into the guest features array.

Return values

The	selected set of guest features.

This function returns a specific 32bit set of features enabled by the guest/driver. `idx` is the index in the guest features array, resp. the 32 bit set to return.

Definition at line 188 of file l4virtio.

15.351.3.5 hdr()

```
l4virtio_config_hdr_t const volatile* L4virtio::Svr::Dev_config::hdr ( ) const  [inline]
```

Get a read-only pointer to the config header.

Returns

Read-only pointer to the shared config header.

Definition at line 347 of file l4virtio.

Referenced by get_cmd(), reset_cmd(), set_device_needs_reset(), and set_status().

Here is the caller graph for this function:

15.351.3.6 negotiated_features()

```
l4_uint32_t L4virtio::Svr::Dev_config::negotiated_features (
            unsigned idx ) const  [inline]
```

Compute a specific set of negotiated features.

Parameters

idx	Index into the guest/host features array.

Return values

The	selected set of negotiated features.

This function returns a specific 32-bit set of features negotiated by the guest/driver and host/device. `idx` is the index in the guest/host features array, resp. the 32-bit set to return.

Definition at line 202 of file l4virtio.

15.351.3.7 qconfig()

```
l4virtio_config_queue_t volatile const* L4virtio::Svr::Dev_config::qconfig (
            unsigned index ) const  [inline]
```

Get queue read-only config data for queue with the given *index*.

Parameters

index	The index of the queue.

Returns

Read-only pointer to the config of the queue with the given *index*, or NULL if *index* is out of range.

Definition at line 282 of file l4virtio.

References L4_UNLIKELY.

Referenced by reset_queue(), and L4virtio::Svr::Device_t< DATA >::setup_queue().

Here is the caller graph for this function:

15.351.3.8 reset_cmd()

```
void L4virtio::Svr::Dev_config::reset_cmd ( ) [inline]
```

Reset the `cmd` register after execution of a command.

This function resets the `cmd` register in order for the client to detect that the command was executed by the device.

Definition at line 231 of file l4virtio.

References l4virtio_config_hdr_t::cmd, and hdr().

Referenced by L4virtio::Svr::Device_t< DATA >::handle_mem_cmd_write().

Here is the call graph for this function:

Here is the caller graph for this function:

15.351.3.9 reset_queue()

```
bool L4virtio::Svr::Dev_config::reset_queue (
            unsigned index,
            unsigned num_max,
            bool inc_generation = false ) const [inline]
```

Reset queue config for the given queue.

Parameters

index	The index of the queue to reset.
num_max	The maximum number of descriptors supported by this queue.
inc_generation	The config generation will be incremented when this is true.

Returns

true on success, or false when *index* is out of range.

Definition at line 324 of file l4virtio.

References L4_UNLIKELY, l4virtio_config_queue_t::num, l4virtio_config_queue_t::num_max, qconfig(), and l4virtio_config_queue_t::ready.

Referenced by L4virtio::Svr::Device_t< DATA >::reset_queue_config().

Here is the call graph for this function:

Here is the caller graph for this function:

15.351.3.10 set_device_needs_reset()

```
void L4virtio::Svr::Dev_config::set_device_needs_reset ( ) [inline]
```

Set DEVICE_NEEDS_RESET bit in device status register.

This function sets the internal status register and also the status register in the shared memory to DEVICE_↩ NEEDS_RESET.

Definition at line 255 of file l4virtio.

References L4virtio::Svr::Dev_status::device_needs_reset(), hdr(), L4virtio::Svr::Dev_status::raw, and l4virtio_config_hdr_t::status.

Referenced by L4virtio::Svr::Device_t< DATA >::device_error().

Here is the call graph for this function:

Here is the caller graph for this function:

15.351.3.11 set_status()

```
void L4virtio::Svr::Dev_config::set_status (
            Status status )  [inline]
```

Set device status register.

Parameters

status	The new value for the device status register.

This function sets the internal status register and also the status register in the shared memory to *status*.

Definition at line 243 of file l4virtio.

References hdr(), L4virtio::Svr::Dev_status::raw, status(), and l4virtio_config_hdr_t::status.

Here is the call graph for this function:

15.351.3.12 status()

```
Status L4virtio::Svr::Dev_config::status ( ) const  [inline]
```

Get current device status (trusted).

Returns

Current device status register (trusted).

The status returned by this function is value stored internally and cannot be written by the guest (i.e., the value can be taken as trusted.)

Definition at line 212 of file l4virtio.

Referenced by L4virtio::Svr::Device_t< DATA >::device_error(), and set_status().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/server/l4virtio

15.352 L4virtio::Svr::Dev_features Struct Reference

Type for device feature bitmap.

Inherited by L4virtio::Svr::Block_features.

Collaboration diagram for L4virtio::Svr::Dev_features:

Public Member Functions

- Dev_features (l4_uint32_t v)

  Make Features from a raw bitmap.

Data Fields

- l4_uint32_t raw

  The raw value of the features bitmap.

- typedef cxx::Bitfield< decltype(raw), 28, 28 > ring_indirect_desc_bfm_t

  Type to access the `ring_indirect_desc` bits (28 to 28) of `raw`.
- ring_indirect_desc_bfm_t::Val ring_indirect_desc () const

  Get the `ring_indirect_desc` bits (28 to 28) of `raw`.
- ring_indirect_desc_bfm_t::Ref ring_indirect_desc ()

  Get a reference to the `ring_indirect_desc` bits (28 to 28) of `raw`.

- typedef cxx::Bitfield< decltype(raw), 29, 29 > ring_event_idx_bfm_t

  Type to access the `ring_event_idx` bits (29 to 29) of `raw`.
- ring_event_idx_bfm_t::Val ring_event_idx () const

  Get the `ring_event_idx` bits (29 to 29) of `raw`.
- ring_event_idx_bfm_t::Ref ring_event_idx ()

  Get a reference to the `ring_event_idx` bits (29 to 29) of `raw`.

15.352.1 Detailed Description

Type for device feature bitmap.

Definition at line 66 of file virtio.

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

15.353 L4virtio::Svr::Dev_status Struct Reference

Type of the device status register.

Collaboration diagram for L4virtio::Svr::Dev_status:

Public Member Functions

- Dev_status (l4_uint32_t v)

  Make Status from raw value.
- bool running () const

  Check if the device is in running state.

Data Fields

- unsigned char raw

  Raw value of the VIRTIO device status register.

- typedef cxx::Bitfield< decltype(raw), 0, 0 > acked_bfm_t

  Type to access the `acked` bits (0 to 0) of `raw`.
- acked_bfm_t::Val acked () const

  Get the `acked` bits (0 to 0) of `raw`.
- acked_bfm_t::Ref acked ()

  Get a reference to the `acked` bits (0 to 0) of `raw`.

- typedef cxx::Bitfield< decltype(raw), 1, 1 > driver_bfm_t

  Type to access the `driver` bits (1 to 1) of `raw`.
- driver_bfm_t::Val driver () const

  Get the `driver` bits (1 to 1) of `raw`.
- driver_bfm_t::Ref driver ()

  Get a reference to the `driver` bits (1 to 1) of `raw`.

- typedef cxx::Bitfield< decltype(raw), 2, 2 > driver_ok_bfm_t

  Type to access the `driver_ok` bits (2 to 2) of `raw`.
- driver_ok_bfm_t::Val driver_ok () const

  Get the `driver_ok` bits (2 to 2) of `raw`.
- driver_ok_bfm_t::Ref driver_ok ()

  Get a reference to the `driver_ok` bits (2 to 2) of `raw`.

- typedef cxx::Bitfield< decltype(raw), 3, 3 > features_ok_bfm_t

  Type to access the `features_ok` bits (3 to 3) of `raw`.
- features_ok_bfm_t::Val features_ok () const

  Get the `features_ok` bits (3 to 3) of `raw`.
- features_ok_bfm_t::Ref features_ok ()

  Get a reference to the `features_ok` bits (3 to 3) of `raw`.

- typedef cxx::Bitfield< decltype(raw), 6, 7 > fail_state_bfm_t

  Type to access the `fail_state` bits (6 to 7) of `raw`.
- fail_state_bfm_t::Val fail_state () const

  Get the `fail_state` bits (6 to 7) of `raw`.
- fail_state_bfm_t::Ref fail_state ()

  Get a reference to the `fail_state` bits (6 to 7) of `raw`.

- typedef cxx::Bitfield< decltype(raw), 6, 6 > device_needs_reset_bfm_t

Type to access the `device_needs_reset` bits (6 to 6) of `raw`.

- device_needs_reset_bfm_t::Val device_needs_reset () const

  Get the `device_needs_reset` bits (6 to 6) of `raw`.
- device_needs_reset_bfm_t::Ref device_needs_reset ()

  Get a reference to the `device_needs_reset` bits (6 to 6) of `raw`.

- typedef cxx::Bitfield< decltype(raw), 7, 7 > failed_bfm_t

  Type to access the `failed` bits (7 to 7) of `raw`.
- failed_bfm_t::Val failed () const

  Get the `failed` bits (7 to 7) of `raw`.
- failed_bfm_t::Ref failed ()

  Get a reference to the `failed` bits (7 to 7) of `raw`.

15.353.1 Detailed Description

Type of the device status register.

Definition at line 32 of file virtio.

15.353.2 Member Function Documentation

15.353.2.1 running()

```
bool L4virtio::Svr::Dev_status::running ( ) const  [inline]
```

Check if the device is in running state.

Returns

true if the device is in running state.

The device is in running state when acked(), driver(), features_ok(), and driver_ok() return true, and device_needs_reset() and failed() return false.

Definition at line 57 of file virtio.

References raw.

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

15.354 L4virtio::Svr::Device_t< DATA > Class Template Reference

Server-side L4-VIRTIO device stub.

Inheritance diagram for L4virtio::Svr::Device_t< DATA >:

Collaboration diagram for L4virtio::Svr::Device_t< DATA >:

Public Member Functions

- virtual void reset ()=0

  reset callback, called for doing a device reset
- virtual bool check_features ()

  callback for checking the subset of accepted features
- virtual bool check_queues ()=0

  callback for checking if the queues at DRIVER_OK transition
- virtual int reconfig_queue (unsigned idx)=0

  callback for client queue-config request
- virtual void register_single_driver_irq ()

  callback for registering a single guest IRQ for all queues (old-style)
- virtual void trigger_driver_config_irq () const =0

  callback for triggering configuration change notification IRQ
- virtual L4::Cap< L4::Irq > device_notify_irq () const

  callback to gather the device notification IRQ (old-style)
- virtual void register_driver_irq (unsigned idx)

  Callback for registering an notification IRQ (multi IRQ).
- virtual L4::Cap< L4::Irq > device_notify_irq (unsigned idx)

  Callback to gather the device notification IRQ (multi IRQ).
- virtual unsigned num_events_supported () const

  Return the highest notification index supported.
- Device_t (Dev_config *dev_config)

  Make a device for the given config.
- Mem_list const * mem_info () const

  Get the memory region list used for this device.

- void reset_queue_config (unsigned idx, unsigned num_max, bool inc_generation=false)

  Trigger reset for the configuration space for queue idx.
- void init_mem_info (unsigned num)

  Initialize the memory region list to the given maximum.
- void device_error ()

  Transition device into DEVICE_NEEDS_RESET state.
- bool setup_queue (Virtqueue *q, unsigned qn, unsigned num_max)

  Enable/disable the specified queue.
- bool handle_mem_cmd_write ()

  Check for a value in the `cmd` *register and handle a write.*

Protected Attributes

- Mem_list _mem_info

  Memory region list.

15.354.1 Detailed Description

template<typename DATA>
class L4virtio::Svr::Device_t< DATA >

Server-side L4-VIRTIO device stub.

This stub supports old-style device registration with single IRQs (via register_iface()) and new-style multi-event registration (using get_device_config(), bind() and get_device_notification_irq()).

In their default implementation the callbacks provide a wrapper from old-style to new-style functions, so that legacy devices provide both interfaces without any changes to their implementation.

New devices should always implement the new-style interface. If required, they can also provide a backward-compatibility mode by implementing the old-style interface as well.

The old-style interface is considered deprecated and will be removed at some point.

Definition at line 783 of file l4virtio.

15.354.2 Member Function Documentation

15.354.2.1 device_error()

```
template<typename DATA >
void L4virtio::Svr::Device_t< DATA >::device_error ( ) [inline]
```

Transition device into DEVICE_NEEDS_RESET state.

This function does a full reset, sets the DEVICE_NEEDS_RESET bit in the device status register, triggering a guest config IRQ if necessary. The driver still needs to perform its own reset and initialization sequence.

Definition at line 1020 of file l4virtio.

References L4virtio::Svr::Dev_status::driver_ok(), L4virtio::Svr::Device_t< DATA >::reset(), L4virtio::Svr::Dev_config::set_device_nee L4virtio::Svr::Dev_config::status(), and L4virtio::Svr::Device_t< DATA >::trigger_driver_config_irq().

Here is the call graph for this function:

15.354.2.2 device_notify_irq()

```
template<typename DATA >
virtual L4::Cap<L4::Irq> L4virtio::Svr::Device_t< DATA >::device_notify_irq (
            unsigned idx ) [inline], [virtual]
```

Callback to gather the device notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 846 of file l4virtio.

References L4Re::chksys(), L4virtio::Svr::Device_t< DATA >::device_notify_irq(), and L4_ENOSYS.

Here is the call graph for this function:

15.354.2.3 handle_mem_cmd_write()

```
template<typename DATA >
bool L4virtio::Svr::Device_t< DATA >::handle_mem_cmd_write ( ) [inline]
```

Check for a value in the `cmd` register and handle a write.

This function checks for a value in the `cmd` register and executes the command if there is any, or returns false if there was no command.

Execution of the command is signaled by a zero in the `cmd` register.

Definition at line 1154 of file l4virtio.

References L4virtio::Svr::Dev_config::get_cmd(), L4_LIKELY, L4VIRTIO_CMD_CFG_QUEUE, L4VIRTIO_CMD_MASK, L4VIRTIO_CMD_SET_STATUS, L4virtio::Svr::Device_t< DATA >::reconfig_queue(), and L4virtio::Svr::Dev_config::reset_cmd().

Here is the call graph for this function:

15.354.2.4 init_mem_info()

```
template<typename DATA >
void L4virtio::Svr::Device_t< DATA >::init_mem_info (
            unsigned num ) [inline]
```

Initialize the memory region list to the given maximum.

Parameters

num	Maximum number of memory regions that can be managed.

Definition at line 1008 of file l4virtio.

References L4virtio::Svr::Device_t< DATA >::_mem_info, and L4virtio::Svr::Driver_mem_list_t< DATA >::init().

Here is the call graph for this function:

15.354.2.5 register_driver_irq()

```
template<typename DATA >
virtual void L4virtio::Svr::Device_t< DATA >::register_driver_irq (
            unsigned idx )  [inline], [virtual]
```

Callback for registering an notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 832 of file l4virtio.

References L4Re::chksys(), L4_ENOSYS, and L4virtio::Svr::Device_t< DATA >::register_single_driver_irq().

Here is the call graph for this function:

15.354.2.6 reset_queue_config()

```
template<typename DATA >
void L4virtio::Svr::Device_t< DATA >::reset_queue_config (
            unsigned idx,
            unsigned num_max,
            bool inc_generation = false )  [inline]
```

Trigger reset for the configuration space for queue *idx*.

Parameters

idx	The queue index to reset.
num_max	Maximum number of entries in this queue.
inc_generation	The config generation will be incremented when this is true.

This function resets the driver-readable configuration space for the queue with the given index. The queue configuration is reset to all 0, and the maximum number of entries in the queue is set to *num_max*.

Definition at line 998 of file l4virtio.

References L4virtio::Svr::Dev_config::reset_queue().

Here is the call graph for this function:

15.354.2.7 setup_queue()

```
template<typename DATA >
bool L4virtio::Svr::Device_t< DATA >::setup_queue (
            Virtqueue * q,
            unsigned qn,
            unsigned num_max )  [inline]
```

Enable/disable the specified queue.

Parameters

q	Pointer to the ring that represents the virtqueue internally.
qn	Index of the queue.
num_max	Maximum number of supported entries in this queue.

Returns

true for success.

- This function calculates the parameters of the virtqueue from the clients configuration space values, checks the accessibility of the queue data structures and initializes *q* to ready state when all checks succeeded.

Definition at line 1047 of file l4virtio.

References L4virtio::Svr::Device_t< DATA >::_mem_info, l4virtio_config_queue_t::avail_addr, L4virtio::Virtqueue::avail_size(), l4virtio_config_queue_t::desc_addr, L4virtio::Virtqueue::desc_size(), L4virtio::Virtqueue::disable(), L4virtio::Svr::Driver_mem_list_t< L4_UNLIKELY, l4virtio_config_queue_t::num, L4virtio::Svr::Dev_config::qconfig(), l4virtio_config_queue_t::ready, L4virtio::Virtqueue::setup(), l4virtio_config_queue_t::used_addr, and L4virtio::Virtqueue::used_size().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/server/l4virtio

15.355 L4virtio::Svr::Driver_mem_list_t< DATA > Class Template Reference

List of driver memory regions assigned to a single L4-VIRTIO transport instance.

Inheritance diagram for L4virtio::Svr::Driver_mem_list_t< DATA >:

Collaboration diagram for L4virtio::Svr::Driver_mem_list_t< DATA >:

Public Types

- typedef L4Re::Util::Unique_cap< L4Re::Dataspace > Ds_cap

  type for storing a data-space capability internally

Public Member Functions

- Driver_mem_list_t ()

  Make an empty, zero capacity list.
- void init (unsigned max)

  Make a fresh list with capacity max.
- bool full () const
- Mem_region const ∗ add (l4_uint64_t drv_base, l4_umword_t size, l4_addr_t offset, Ds_cap &&ds)

  Add a new region to the list.
- void remove (Mem_region const ∗r)

  Remove the given region from the list.
- Mem_region ∗ find (l4_uint64_t base, l4_umword_t size) const

  Find memory region containing the given driver address region.
- void load_desc (Virtqueue::Desc const &desc, Request_processor const ∗p, Virtqueue::Desc const ∗∗table) const

  Default implementation for loading an indirect descriptor.
- void load_desc (Virtqueue::Desc const &desc, Request_processor const ∗p, Mem_region const ∗∗data) const

  Default implementation returning the Driver_mem_region.
- template<typename ARG >
  void load_desc (Virtqueue::Desc const &desc, Request_processor const ∗p, ARG ∗data) const

  Default implementation returning generic information.

15.355.1 Detailed Description

template<typename DATA>
class L4virtio::Svr::Driver_mem_list_t< DATA >

List of driver memory regions assigned to a single L4-VIRTIO transport instance.

Note

The regions added to this list *must* never overlap.

Definition at line 605 of file l4virtio.

15.355.2 Member Function Documentation

15.355.2.1 add()

```
template<typename DATA >
Mem_region const* L4virtio::Svr::Driver_mem_list_t< DATA >::add (
            l4_uint64_t drv_base,
            l4_umword_t size,
            l4_addr_t offset,
            Ds_cap && ds )  [inline]
```

Add a new region to the list.

Parameters

drv_base	Driver base address of the region.
size	Size of the region in bytes.
offset	Offset within the data space attached to drv_base.
ds	Data space backing the driver memory.

Returns

A pointer to the new region.

Definition at line 645 of file l4virtio.

References L4Re::chksys(), L4virtio::Svr::Driver_mem_list_t< DATA >::full(), and L4_ENOMEM.

Here is the call graph for this function:

15.355.2.2 find()

```
template<typename DATA >
Mem_region* L4virtio::Svr::Driver_mem_list_t< DATA >::find (
            l4_uint64_t base,
            l4_umword_t size ) const  [inline]
```

Find memory region containing the given driver address region.

Parameters

base	Driver base address.
size	Size of the region.

Returns

Pointer to the region containing the given region, NULL if none is found.

Definition at line 679 of file l4virtio.

Referenced by L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc(), and L4virtio::Svr::Device_t< DATA >::setup_queue().

Here is the caller graph for this function:

15.355.2.3 full()

```
template<typename DATA >
bool L4virtio::Svr::Driver_mem_list_t< DATA >::full ( ) const  [inline]
```

Returns

True if the remaining capacity is 0.

Definition at line 634 of file l4virtio.

Referenced by L4virtio::Svr::Driver_mem_list_t< DATA >::add().

Here is the caller graph for this function:

15.355.2.4 init()

```
template<typename DATA >
void L4virtio::Svr::Driver_mem_list_t< DATA >::init (
            unsigned max )  [inline]
```

Make a fresh list with capacity *max*.

Parameters

max	The capacity of this vector.

Definition at line 626 of file l4virtio.

References cxx::max().

Referenced by L4virtio::Svr::Device_t< DATA >::init_mem_info().

Here is the call graph for this function:

Here is the caller graph for this function:

15.355.2.5 load_desc() [1/3]

```
template<typename DATA >
template<typename ARG >
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
            Virtqueue::Desc const & desc,
            Request_processor const * p,
            ARG * data ) const  [inline]
```

Default implementation returning generic information.

Template Parameters

ARG	Abstract argument type used with Request_processor::start() and Request_processor::next() to deliver the result of loading a descriptor. This type must provide a constructor taking three arguments: (1) pointer to a Driver_mem_region, (2) the Virtqueue::Desc descriptor, and (3) a pointer to the calling Request_processor.

Parameters

	desc	The descriptor to load
	p	The request processor calling us
out	*data*	Shall be assigned to ARG(mem, desc, p)

Exceptions

Bad_descriptor	The descriptor address could not be translated.

Definition at line 740 of file l4virtio.

References L4virtio::Virtqueue::Desc::addr, L4virtio::Svr::Bad_descriptor::Bad_address, L4virtio::Svr::Driver_mem_list_t< DATA >::fi
L4virtio::Ptr< T >::get(), L4_UNLIKELY, and L4virtio::Virtqueue::Desc::len.

Here is the call graph for this function:

15.355.2.6 load_desc() [2/3]

```
template<typename DATA >
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
            Virtqueue::Desc const & desc,
            Request_processor const * p,
            Mem_region const ** data ) const  [inline]
```

Default implementation returning the Driver_mem_region.

Parameters

	desc	The descriptor to load
	p	The request processor calling us
out	*data*	Shall be set to a pointer to the Driver_mem_region that covers the descriptor.

Exceptions

Bad_descriptor	The descriptor address could not be translated.

Definition at line 713 of file l4virtio.

References L4virtio::Virtqueue::Desc::addr, L4virtio::Svr::Bad_descriptor::Bad_address, L4virtio::Svr::Driver_mem_list_t< DATA >::fi
L4virtio::Ptr< T >::get(), L4_UNLIKELY, and L4virtio::Virtqueue::Desc::len.

Here is the call graph for this function:

15.355.2.7 load_desc() [3/3]

```
template<typename DATA >
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
            Virtqueue::Desc const & desc,
            Request_processor const * p,
            Virtqueue::Desc const ** table ) const  [inline]
```

Default implementation for loading an indirect descriptor.

Parameters

	desc	The descriptor to load
	p	The request processor calling us
out	*table*	Shall be set to the loaded descriptor table

Exceptions

Bad_descriptor	The descriptor address could not be translated.

Definition at line 693 of file l4virtio.

References L4virtio::Virtqueue::Desc::addr, L4virtio::Svr::Bad_descriptor::Bad_address, L4virtio::Svr::Driver_mem_list_t< DATA >::fi
L4virtio::Ptr< T >::get(), L4_UNLIKELY, L4virtio::Virtqueue::Desc::len, and L4virtio::Svr::Driver_mem_region_t< DATA >::local().

Here is the call graph for this function:

15.355.2.8 remove()

```
template<typename DATA >
void L4virtio::Svr::Driver_mem_list_t< DATA >::remove (
            Mem_region const * r )  [inline]
```

Remove the given region from the list.

Parameters

r	The region to remove (result from add(), or find()).

Definition at line 659 of file l4virtio.

References L4Re::chksys(), and L4_ERANGE.

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/server/l4virtio

15.356 L4virtio::Svr::Driver_mem_region_t< DATA > Class Template Reference

Region of driver memory, that shall be managed locally.

Inheritance diagram for L4virtio::Svr::Driver_mem_region_t< DATA >:

Collaboration diagram for L4virtio::Svr::Driver_mem_region_t< DATA >:

Public Member Functions

- Driver_mem_region_t ()

  Make default empty memory region.
- Driver_mem_region_t (l4_uint64_t drv_base, l4_umword_t size, l4_addr_t offset, Ds_cap &&ds)

  Make a local memory region for the given driver values.
- bool is_writable () const
- Flags flags () const
- bool empty () const
- l4_uint64_t drv_base () const
- void ∗ local_base () const
- l4_umword_t size () const
- l4_addr_t ds_offset () const
- L4::Cap< L4Re::Dataspace > ds () const
- bool contains (l4_uint64_t base, l4_umword_t size) const

  Test if the given driver address range is within this region.
- template<typename T >
  T ∗ local (Ptr< T > p) const

  Get the local address for driver address p.

15.356.1 Detailed Description

template<typename DATA>
class L4virtio::Svr::Driver_mem_region_t< DATA >

Region of driver memory, that shall be managed locally.

Template Parameters

DATA	Class defining additional information

Definition at line 429 of file l4virtio.

15.356.2 Constructor & Destructor Documentation

15.356.2.1 Driver_mem_region_t()

```
template<typename DATA >
L4virtio::Svr::Driver_mem_region_t< DATA >::Driver_mem_region_t (
            l4_uint64_t drv_base,
            l4_umword_t size,
            l4_addr_t offset,
            Ds_cap && ds )  [inline]
```

Make a local memory region for the given driver values.

Parameters

drv_base	Base address of the memory region used by the driver.
size	Size of the memory region.
offset	Offset within the data space that is mapped to *drv_base* within the driver.
ds	Data space capability backing the memory.

This constructor attaches the region of given data space to the local address space and stores the corresponding data for later reference.

Definition at line 474 of file l4virtio.

References L4Re::chksys().

Here is the call graph for this function:

15.356.3 Member Function Documentation

15.356.3.1 contains()

```
template<typename DATA >
bool L4virtio::Svr::Driver_mem_region_t< DATA >::contains (
            l4_uint64_t base,
            l4_umword_t size ) const [inline]
```

Test if the given driver address range is within this region.

Parameters

base	The driver base address.
size	The size of the region to lookup.

Returns

true if the given driver address region is contained in this region, false else.

Definition at line 568 of file l4virtio.

References L4virtio::Svr::Driver_mem_region_t< DATA >::size().

Here is the call graph for this function:

15.356.3.2 drv_base()

```
template<typename DATA >
l4_uint64_t L4virtio::Svr::Driver_mem_region_t< DATA >::drv_base ( ) const [inline]
```

Returns

The base address used by the driver.

Definition at line 547 of file l4virtio.

15.356.3.3 ds()

```
template<typename DATA >
L4::Cap<L4Re::Dataspace> L4virtio::Svr::Driver_mem_region_t< DATA >::ds ( ) const  [inline]
```

Returns

The data space capability for this region.

Definition at line 559 of file l4virtio.

15.356.3.4 ds_offset()

```
template<typename DATA >
l4_addr_t L4virtio::Svr::Driver_mem_region_t< DATA >::ds_offset ( ) const  [inline]
```

Returns

The offset within the data space.

Definition at line 556 of file l4virtio.

15.356.3.5 empty()

```
template<typename DATA >
bool L4virtio::Svr::Driver_mem_region_t< DATA >::empty ( ) const  [inline]
```

Returns

True if the region is empty (size == 0), false otherwise.

Definition at line 543 of file l4virtio.

15.356.3.6 flags()

```
template<typename DATA >
Flags L4virtio::Svr::Driver_mem_region_t< DATA >::flags ( ) const  [inline]
```

Returns

The flags for this region.

Definition at line 540 of file l4virtio.

15.356.3.7 is_writable()

```
template<typename DATA >
bool L4virtio::Svr::Driver_mem_region_t< DATA >::is_writable ( ) const  [inline]
```

Returns

True if the region is writable, false otherwise.

Definition at line 537 of file l4virtio.

15.356.3.8 local()

```
template<typename DATA >
template<typename T >
T* L4virtio::Svr::Driver_mem_region_t< DATA >::local (
            Ptr< T > p ) const  [inline]
```

Get the local address for driver address *p*.

Parameters

p	Driver address to translate.

Precondition

p must be contained in this region.

Returns

Local address for the given driver address *p*.

Definition at line 592 of file l4virtio.

References L4virtio::Ptr< T >::get().

Referenced by L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc().

Here is the call graph for this function:

Here is the caller graph for this function:

15.356.3.9 local_base()

```
template<typename DATA >
void* L4virtio::Svr::Driver_mem_region_t< DATA >::local_base ( ) const  [inline]
```

Returns

The local base address.

Definition at line 550 of file l4virtio.

15.356.3.10 size()

```
template<typename DATA >
l4_umword_t L4virtio::Svr::Driver_mem_region_t< DATA >::size ( ) const  [inline]
```

Returns

The size of the region in bytes.

Definition at line 553 of file l4virtio.

Referenced by L4virtio::Svr::Driver_mem_region_t< DATA >::contains().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/server/l4virtio

15.357 L4virtio::Svr::Request_processor Class Reference

Encapsulate the state for processing a VIRTIO request.

Collaboration diagram for L4virtio::Svr::Request_processor:

Public Member Functions

- template<typename DESC_MAN , typename ... ARGS>
  void start (DESC_MAN *dm, Virtqueue *ring, Virtqueue::Head_desc const &request, ARGS... args)

  Start processing a new request.
- template<typename DESC_MAN , typename ... ARGS>
  Virtqueue::Request const & start (DESC_MAN *dm, Virtqueue::Request const &request, ARGS... args)

  Start processing a new request.
- Virtqueue::Desc::Flags current_flags () const

  Get the flags of the currently processed descriptor.
- bool has_more () const

  Are there more chained descriptors?
- template<typename DESC_MAN , typename ... ARGS>
  bool next (DESC_MAN *dm, ARGS... args)

  Switch to the next descriptor in a descriptor chain.

15.357.1 Detailed Description

Encapsulate the state for processing a VIRTIO request.

A VIRTIO request is a possibly chained list of descriptors retrieved from the available ring of a virtqueue, using Virtqueue::next_avail().

The descriptor processing depends on helper (DESC_MAN) for interpreting the descriptors in the context of the device implementation.

DESC_MAN has to provide the functionality to safely dereference a descriptor from a descriptor list.

The following methods must be provided by DESC_MAN:

- `DESC_MAN::load_desc(Virtqueue::Desc const &desc, Request_processor const *proc, Virtqueue::Desc const **table)`

  This function is used to dereference *desc* as an indirect descriptor table, and must return a pointer to an indirect descriptor table.
- `DESC_MAN::load_desc(Virtqueue::Desc const &desc, Request_processor const *proc, ...)`

  This function is used to dereference a descriptor as a normal data buffer, and '...' are the arguments that are passed to start() and next().

Definition at line 398 of file virtio.

15.357.2 Member Function Documentation

15.357.2.1 current_flags()

`Virtqueue::Desc::Flags L4virtio::Svr::Request_processor::current_flags ( ) const [inline]`

Get the flags of the currently processed descriptor.

Returns

The flags of the currently processed descriptor.

Definition at line 468 of file virtio.

References L4virtio::Virtqueue::Desc::flags.

15.357.2.2 has_more()

`bool L4virtio::Svr::Request_processor::has_more ( ) const [inline]`

Are there more chained descriptors?

Returns

true if there are more chained descriptors in the current request.

Definition at line 475 of file virtio.

References L4virtio::Virtqueue::Desc::flags, and L4virtio::Virtqueue::Desc::Flags::next().

Referenced by L4virtio::Svr::Block_request< Ds_data >::data_size(), L4virtio::Svr::Block_request< Ds_data >::has_more(), and L4virtio::Svr::Block_request< Ds_data >::next_block().

Here is the call graph for this function:

Here is the caller graph for this function:

15.357.2.3 next()

```
template<typename DESC_MAN , typename ... ARGS>
bool L4virtio::Svr::Request_processor::next (
            DESC_MAN * dm,
            ARGS... args ) [inline]
```

Switch to the next descriptor in a descriptor chain.

Template Parameters

DESC_MAN	Type of descriptor manager (implicit).

Parameters

dm	Descriptor manager that is used to translate VIRTIO descriptor addresses.
args	Extra arguments passed to dm->load_desc()

Return values

true	A next descriptor is available.
false	No descriptor available.

Exceptions

Bad_descriptor	The `next` index of this descriptor is invalid.

Definition at line 492 of file virtio.

References L4virtio::Svr::Bad_descriptor::Bad_flags, L4virtio::Svr::Bad_descriptor::Bad_next, L4virtio::Virtqueue::Desc::flags, L4virtio::Virtqueue::Desc::Flags::indirect(), L4_UNLIKELY, L4virtio::Virtqueue::Desc::Flags::next(), and L4virtio::Virtqueue::Desc::next

Referenced by L4virtio::Svr::Block_request< Ds_data >::data_size(), L4virtio::Svr::Block_request< Ds_data >::has_more(), and L4virtio::Svr::Block_request< Ds_data >::next_block().

Here is the call graph for this function:

Here is the caller graph for this function:

15.357.2.4 start() [1/2]

```
template<typename DESC_MAN , typename ... ARGS>
void L4virtio::Svr::Request_processor::start (
            DESC_MAN * dm,
            Virtqueue * ring,
            Virtqueue::Head_desc const & request,
            ARGS... args ) [inline]
```

Start processing a new request.

Template Parameters

DESC_MAN	Type of descriptor manager (implicit).

Parameters

dm	Descriptor manager that is used to translate VIRTIO descriptor addresses.
ring	VIRTIO ring of the request.
request	VIRTIO request from Virtqueue::next_avail()
args	Extra arguments passed to dm->load_desc()

Precondition

The given request must be valid.

Exceptions

Bad_descriptor	The descriptor has an invalid size or load_desc() has thrown an exception by itself.

Definition at line 427 of file virtio.

References L4virtio::Svr::Bad_descriptor::Bad_size, L4virtio::Svr::Virtqueue::Head_desc::desc(), L4virtio::Svr::Virtqueue::desc(), L4virtio::Virtqueue::Desc::flags, L4virtio::Virtqueue::Desc::Flags::indirect(), L4_UNLIKELY, L4virtio::Virtqueue::Desc::len, and L4virtio::Virtqueue::num().

Referenced by L4virtio::Svr::Block_request< Ds_data >::data_size(), and start().

Here is the call graph for this function:

Here is the caller graph for this function:

15.357.2.5 start() [2/2]

```
template<typename DESC_MAN , typename ...  ARGS>
Virtqueue::Request const& L4virtio::Svr::Request_processor::start (
            DESC_MAN * dm,
            Virtqueue::Request const & request,
            ARGS...  args )  [inline]
```

Start processing a new request.

Template Parameters

DESC_MAN	Type of descriptor manager (implicit).

Parameters

dm	Descriptor manager that is used to translate VIRTIO descriptor addresses.
request	VIRTIO request from Virtqueue::next_avail()
args	Extra arguments passed to dm->load_desc()

Precondition

The given request must be valid.

Definition at line 458 of file virtio.

References start().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio

15.358 L4virtio::Svr::Virtqueue Class Reference

Virtqueue implementation for the device.

Inheritance diagram for L4virtio::Svr::Virtqueue:

Collaboration diagram for L4virtio::Svr::Virtqueue:

Data Structures

- class Head_desc

  VIRTIO request, essentially a descriptor from the available ring.

Public Member Functions

- Request next_avail ()

Get the next available descriptor from the available ring.

- bool desc_avail () const

  Test for available descriptors.
- void consumed (Head_desc const &r, l4_uint32_t len=0)

  Put the given descriptor into the used ring.
- void disable_notify ()

  Set the 'no notify' flag for this queue.
- void enable_notify ()

  Clear the 'no notify' flag for this queue.
- Desc const * desc (unsigned idx) const

  Get a descriptor from the descriptor list.

Additional Inherited Members

15.358.1 Detailed Description

Virtqueue implementation for the device.

This class represents a single virtqueue, with a local running available index.

Note

The Virtqueue implementation is not thread-safe.

Definition at line 87 of file virtio.

15.358.2 Member Function Documentation

15.358.2.1 consumed()

```
void L4virtio::Svr::Virtqueue::consumed (
            Head_desc const & r,
            l4_uint32_t len = 0 )  [inline]
```

Put the given descriptor into the used ring.

Parameters

r	request that shall be marked as finished.
len	the total number of bytes written.

Precondition

queue must be in working state.

r must be a valid request from this queue.

Definition at line 166 of file virtio.

References L4virtio::Virtqueue::_desc, L4virtio::Virtqueue::_idx_mask, L4virtio::Virtqueue::_used, L4virtio::Virtqueue::Used::idx, and L4virtio::Virtqueue::Used::ring.

15.358.2.2 desc()

```
Desc const* L4virtio::Svr::Virtqueue::desc (
            unsigned idx ) const  [inline]
```

Get a descriptor from the descriptor list.

Parameters

idx	the index of the descriptor.

Precondition

$idx < num$

queue must be in working state

Definition at line 231 of file virtio.

References L4virtio::Virtqueue::_desc.

Referenced by L4virtio::Svr::Request_processor::start().

Here is the caller graph for this function:

15.358.2.3 desc_avail()

```
bool L4virtio::Svr::Virtqueue::desc_avail ( ) const  [inline]
```

Test for available descriptors.

Returns

true if there are descriptors available, false if not.

Precondition

The queue must be in working state.

Definition at line 154 of file virtio.

References L4virtio::Virtqueue::_avail, L4virtio::Virtqueue::_current_avail, and L4virtio::Virtqueue::Avail::idx.

15.358.2.4 disable_notify()

```
void L4virtio::Svr::Virtqueue::disable_notify ( ) [inline]
```

Set the 'no notify' flag for this queue.

This function may be called on a disabled queue.

Definition at line 208 of file virtio.

References L4virtio::Virtqueue::_used, L4virtio::Virtqueue::Used::flags, L4_LIKELY, L4virtio::Virtqueue::Used::Flags::no_notify(), and L4virtio::Virtqueue::ready().

Here is the call graph for this function:

15.358.2.5 enable_notify()

```
void L4virtio::Svr::Virtqueue::enable_notify ( ) [inline]
```

Clear the 'no notify' flag for this queue.

This function may be called on a disabled queue.

Definition at line 219 of file virtio.

References L4virtio::Virtqueue::_used, L4virtio::Virtqueue::Used::flags, L4_LIKELY, L4virtio::Virtqueue::Used::Flags::no_notify(), and L4virtio::Virtqueue::ready().

Here is the call graph for this function:

15.358.2.6 next_avail()

```
Request L4virtio::Svr::Virtqueue::next_avail ( ) [inline]
```

Get the next available descriptor from the available ring.

Precondition

The queue must be in working state.

Returns

A Request for the next available descriptor, the Request is invalid if there are no descriptors in the available ring.

Note

The return value must be checked even when a previous desc_avail() returned true.

Definition at line 137 of file virtio.

References L4virtio::Virtqueue::_avail, L4virtio::Virtqueue::_current_avail, L4virtio::Virtqueue::_idx_mask, L4virtio::Virtqueue::Avail::idx, L4_LIKELY, and L4virtio::Virtqueue::Avail::ring.

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio

15.359 L4virtio::Svr::Virtqueue::Head_desc Class Reference

VIRTIO request, essentially a descriptor from the available ring.

Inherited by L4virtio::Svr::Virtqueue::Request.

Collaboration diagram for L4virtio::Svr::Virtqueue::Head_desc:

Public Member Functions

- Head_desc ()

  Make invalid (NULL) request.
- bool valid () const
- operator Null_ptr_check const ∗ () const
- Desc const ∗ desc () const

15.359.1 Detailed Description

VIRTIO request, essentially a descriptor from the available ring.

Definition at line 93 of file virtio.

15.359.2 Member Function Documentation

15.359.2.1 desc()

```
Desc const* L4virtio::Svr::Virtqueue::Head_desc::desc ( ) const  [inline]
```

Returns

Pointer to the head descriptor of the request.

Definition at line 114 of file virtio.

Referenced by L4virtio::Svr::Request_processor::start().

Here is the caller graph for this function:

15.359.2.2 operator Null_ptr_check const *()

```
L4virtio::Svr::Virtqueue::Head_desc::operator Null_ptr_check const * ( ) const  [inline]
```

Returns

True if the request is valid (not NULL).

Definition at line 110 of file virtio.

15.359.2.3 valid()

```
bool L4virtio::Svr::Virtqueue::Head_desc::valid ( ) const  [inline]
```

Returns

True if the request is valid (not NULL).

Definition at line 107 of file virtio.

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio

15.360 L4virtio::Virtqueue Class Reference

Low-level Virtqueue.

Inheritance diagram for L4virtio::Virtqueue:

Collaboration diagram for L4virtio::Virtqueue:

Data Structures

- class Avail

  Type of available ring, this is read-only for the host.
- class Desc

  Descriptor in the descriptor table.
- class Used

  Used ring.
- struct Used_elem

  Type of an element of the used ring.

Public Types

- enum

  Fixed alignment values for different parts of a virtqueue.

Public Member Functions

- void disable ()

  Completely disable the queue.
- unsigned long total_size () const

  Calculate the total size of this virtqueue.
- unsigned long avail_offset () const

  Get the offset of the available ring from the descriptor table.
- unsigned long used_offset () const

  Get the offset of the used ring from the descriptor table.
- void setup (unsigned num, void ∗desc, void ∗avail, void ∗used)

  Enable this queue.
- void setup_simple (unsigned num, void ∗ring)

  Enable this queue.
- void dump (Desc const ∗d) const

  Dump descriptors for this queue.
- bool ready () const

  Test if this queue is in working state.
- unsigned num () const
- bool no_notify_guest () const

  Get the no IRQ flag of this queue.
- bool no_notify_host () const

  Get the no notify flag of this queue.
- void no_notify_host (bool value)

  Set the no-notify flag for this queue.
- l4_uint16_t get_avail_idx () const

  Get available index from available ring (for debugging).
- l4_uint16_t get_tail_avail_idx () const

  Get tail-available index stored in local state (for debugging).

Static Public Member Functions

- static unsigned long total_size (unsigned num)

  Calculate the total size for a virtqueue of the given dimensions.
- static unsigned long desc_size (unsigned num)

  Calculate the size of the descriptor table for `num` *entries.*
- static unsigned long desc_align ()

  Get the alignment in zero LSBs needed for the descriptor table.
- static unsigned long avail_size (unsigned num)

  Calculate the size of the available ring for `num` *entries.*
- static unsigned long avail_align ()

  Get the alignment in zero LSBs needed for the available ring.
- static unsigned long used_size (unsigned num)

  Calculate the size of the used ring for `num` *entries.*
- static unsigned long used_align ()

  Get the alignment in zero LSBs needed for the used ring.

Protected Member Functions

- Virtqueue ()

  Create a disabled virtqueue.

Protected Attributes

- Desc * _desc

  pointer to descriptor table, NULL if queue is off.
- Avail * _avail

  pointer to available ring.
- Used * _used

  pointer to used ring.
- l4_uint16_t _current_avail

  The life counter for the queue.
- l4_uint16_t _idx_mask

  mask used for indexing into the descriptor table and the rings.

15.360.1 Detailed Description

Low-level Virtqueue.

This class represents a single virtqueue, with a local running available index.

Note

The Virtqueue implementation is not thread-safe.

Definition at line 89 of file virtqueue.

15.360.2 Member Function Documentation

15.360.2.1 avail_align()

```
static unsigned long L4virtio::Virtqueue::avail_align ( ) [inline], [static]
```

Get the alignment in zero LSBs needed for the available ring.

Returns

The alignment in zero LSBs needed for an available ring.

Definition at line 293 of file virtqueue.

15.360.2.2 avail_size()

```
static unsigned long L4virtio::Virtqueue::avail_size (
            unsigned num ) [inline], [static]
```

Calculate the size of the available ring for `num` entries.

Parameters

num	The number of entries in the available ring.

Returns

The size in bytes needed for an available ring with `num` entries.

Definition at line 285 of file virtqueue.

References num().

Referenced by L4virtio::Svr::Device_t< DATA >::setup_queue(), setup_simple(), and total_size().

Here is the call graph for this function:

Here is the caller graph for this function:

15.360.2.3 desc_align()

```
static unsigned long L4virtio::Virtqueue::desc_align ( ) [inline], [static]
```

Get the alignment in zero LSBs needed for the descriptor table.

Returns

The alignment in zero LSBs needed for a descriptor table.

Definition at line 275 of file virtqueue.

15.360.2.4 desc_size()

```
static unsigned long L4virtio::Virtqueue::desc_size (
            unsigned num )  [inline], [static]
```

Calculate the size of the descriptor table for `num` entries.

Parameters

num	The number of entries in the descriptor table.

Returns

The size in bytes needed for a descriptor table with `num` entries.

Definition at line 267 of file virtqueue.

References num().

Referenced by L4virtio::Svr::Device_t< DATA >::setup_queue(), setup_simple(), and total_size().

Here is the call graph for this function:

Here is the caller graph for this function:

15.360.2.5 disable()

```
void L4virtio::Virtqueue::disable ( ) [inline]
```

Completely disable the queue.

setup() must be used to enable the queue again.

Definition at line 230 of file virtqueue.

References _desc.

Referenced by L4virtio::Svr::Block_dev_base< Ds_data >::reset(), and L4virtio::Svr::Device_t< DATA >::setup_queue().

Here is the caller graph for this function:

15.360.2.6 dump()

```
void L4virtio::Virtqueue::dump (
            Desc const * d ) const [inline]
```

Dump descriptors for this queue.

Precondition

the queue must be in working state.

Definition at line 397 of file virtqueue.

References _desc, and L4virtio::Virtqueue::Desc::dump().

Here is the call graph for this function:

15.360.2.7 get_avail_idx()

```
l4_uint16_t L4virtio::Virtqueue::get_avail_idx ( ) const  [inline]
```

Get available index from available ring (for debugging).

Precondition

Queue must be in a working state.

Returns

current index in the available ring (shared between device model and device driver).

Definition at line 454 of file virtqueue.

References _avail, and L4virtio::Virtqueue::Avail::idx.

15.360.2.8 get_tail_avail_idx()

```
l4_uint16_t L4virtio::Virtqueue::get_tail_avail_idx ( ) const  [inline]
```

Get tail-available index stored in local state (for debugging).

Returns

current tail index for the the available ring.

Definition at line 461 of file virtqueue.

References _current_avail.

15.360.2.9 no_notify_guest()

```
bool L4virtio::Virtqueue::no_notify_guest ( ) const  [inline]
```

Get the no IRQ flag of this queue.

Precondition

queue must be in working state.

Returns

true if the guest does not want to get IRQs (currently).

Definition at line 419 of file virtqueue.

References _avail, L4virtio::Virtqueue::Avail::flags, and L4virtio::Virtqueue::Avail::Flags::no_irq().

Here is the call graph for this function:

15.360.2.10 no_notify_host() [1/2]

```
bool L4virtio::Virtqueue::no_notify_host ( ) const  [inline]
```

Get the no notify flag of this queue.

Precondition

queue must be in working state.

Returns

true if the host does not want to get IRQs (currently).

Definition at line 431 of file virtqueue.

References _used, L4virtio::Virtqueue::Used::flags, and L4virtio::Virtqueue::Used::Flags::no_notify().

Referenced by L4virtio::Driver::Device::send().

Here is the call graph for this function:

Here is the caller graph for this function:

15.360.2.11 no_notify_host() [2/2]

```
void L4virtio::Virtqueue::no_notify_host (
            bool value )  [inline]
```

Set the no-notify flag for this queue.

Precondition

Queue must be in a working state.

Definition at line 441 of file virtqueue.

References _used, L4virtio::Virtqueue::Used::flags, and L4virtio::Virtqueue::Used::Flags::no_notify().

Here is the call graph for this function:

15.360.2.12 num()

```
unsigned L4virtio::Virtqueue::num ( ) const  [inline]
```

Returns

The number of entries in the ring.

Definition at line 409 of file virtqueue.

References _idx_mask.

Referenced by avail_size(), desc_size(), L4virtio::Driver::Virtqueue::init_queue(), L4virtio::Driver::Virtqueue::initialize_rings(), L4virtio::Driver::Block_device::process_used_queue(), setup(), setup_simple(), L4virtio::Svr::Request_processor::start(), total_size(), and used_size().

Here is the caller graph for this function:

15.360.2.13 ready()

```
bool L4virtio::Virtqueue::ready ( ) const  [inline]
```

Test if this queue is in working state.

Returns

true when the queue is in working state, false else.

Definition at line 405 of file virtqueue.

References _desc, and L4_LIKELY.

Referenced by L4virtio::Svr::Virtqueue::disable_notify(), L4virtio::Svr::Virtqueue::enable_notify(), and L4virtio::Svr::Block_dev_base<

Here is the caller graph for this function:

15.360.2.14 setup()

```
void L4virtio::Virtqueue::setup (
            unsigned num,
            void * desc,
            void * avail,
            void * used )  [inline]
```

Enable this queue.

Parameters

num	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
desc	The address of the descriptor table. (Must be Desc_align aligned and at least `desc_size(num)` bytes in size.)
avail	The address of the available ring. (Must be Avail_align aligned and at least `avail_size(num)` bytes in size.)
used	The address of the used ring. (Must be Used_align aligned and at least `used_size(num)` bytes in size.)

Due to the data type of the descriptors, the queue can have a maximum size of 2^16.

Definition at line 355 of file virtqueue.

References _avail, _current_avail, _desc, _idx_mask, _used, L4_EINVAL, and num().

Referenced by L4virtio::Driver::Virtqueue::init_queue(), L4virtio::Svr::Device_t< DATA >::setup_queue(), and setup_simple().

Here is the call graph for this function:

Here is the caller graph for this function:

15.360.2.15 setup_simple()

```
void L4virtio::Virtqueue::setup_simple (
            unsigned num,
            void * ring )  [inline]
```

Enable this queue.

Parameters

num	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
ring	The base address for the queue data structure. The memory block at `ring` must be at least `total_size(num)` bytes in size and have an alignment of Desc_align (desc_align()) bits.

Due to the data type of the descriptors, the queue can have a maximum size of 2^16.

Definition at line 384 of file virtqueue.

References avail_size(), desc_size(), l4_round_size(), num(), and setup().

Referenced by L4virtio::Driver::Virtqueue::init_queue().

Here is the call graph for this function:

Here is the caller graph for this function:

15.360.2.16 total_size() [1/2]

```
unsigned long L4virtio::Virtqueue::total_size ( ) const  [inline]
```

Calculate the total size of this virtqueue.

Precondition

The queue has been set up.

Definition at line 320 of file virtqueue.

References _desc, _used, num(), and used_size().

Here is the call graph for this function:

15.360.2.17 total_size() [2/2]

```
static unsigned long L4virtio::Virtqueue::total_size (
            unsigned num )  [inline], [static]
```

Calculate the total size for a virtqueue of the given dimensions.

Parameters

num	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).

Returns

The total size in bytes of the queue data structures.

Definition at line 251 of file virtqueue.

References avail_size(), desc_size(), l4_round_size(), num(), and used_size().

Referenced by L4virtio::Driver::Block_device::setup_device().

Here is the call graph for this function:

Here is the caller graph for this function:

15.360.2.18 used_align()

```
static unsigned long L4virtio::Virtqueue::used_align ( ) [inline], [static]
```

Get the alignment in zero LSBs needed for the used ring.

Returns

The alignment in zero LSBs needed for an used ring.

Definition at line 312 of file virtqueue.

15.360.2.19 used_size()

```
static unsigned long L4virtio::Virtqueue::used_size (
            unsigned num ) [inline], [static]
```

Calculate the size of the used ring for `num` entries.

Parameters

num	The number of entries in the used ring.

Returns

The size in bytes needed for an used ring with `num` entries.

Definition at line 304 of file virtqueue.

References num().

Referenced by L4virtio::Svr::Device_t< DATA >::setup_queue(), and total_size().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

15.361 L4virtio::Virtqueue::Avail Class Reference

Type of available ring, this is read-only for the host.

Collaboration diagram for L4virtio::Virtqueue::Avail:

Data Structures

- struct Flags

  Flags of the available ring.

Data Fields

- Flags flags

  flags of available ring
- l4_uint16_t idx

  available index written by guest
- l4_uint16_t ring [ ]

  array of available descriptor indexes.

15.361.1 Detailed Description

Type of available ring, this is read-only for the host.

Definition at line 136 of file virtqueue.

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

15.362 L4virtio::Virtqueue::Avail::Flags Struct Reference

Flags of the available ring.

Collaboration diagram for L4virtio::Virtqueue::Avail::Flags:

Public Member Functions

- Flags (l4_uint16_t v)

  Make Flags from the raw value.

Data Fields

- l4_uint16_t raw

  raw 16bit flags value of the available ring.

- typedef cxx::Bitfield< decltype(raw), 0, 0 > no_irq_bfm_t

  Guest does not want to receive interrupts when requests are finished.
- no_irq_bfm_t::Val no_irq () const

  Get the `no_irq` bits (0 to 0) of `raw`.
- no_irq_bfm_t::Ref no_irq ()

  Get a reference to the `no_irq` bits (0 to 0) of `raw`.

15.362.1 Detailed Description

Flags of the available ring.

Definition at line 142 of file virtqueue.

15.362.2 Member Typedef Documentation

15.362.2.1 no_irq_bfm_t

```
typedef cxx::Bitfield<decltype( raw ), 0 , 0 > L4virtio::Virtqueue::Avail::Flags::no_irq_bfm_t
```

Guest does not want to receive interrupts when requests are finished.

Type to access the `no_irq` bits (0 to 0) of `raw`.

Definition at line 151 of file virtqueue.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtqueue

15.363 L4virtio::Virtqueue::Desc Class Reference

Descriptor in the descriptor table.

Collaboration diagram for L4virtio::Virtqueue::Desc:

Data Structures

- struct Flags

  Type for descriptor flags.

Public Member Functions

- void dump (unsigned idx) const

  Dump a single descriptor.

Data Fields

- Ptr< void > addr

  Address stored in descriptor.
- l4_uint32_t len

  Length of described buffer.
- Flags flags

  Descriptor flags.
- l4_uint16_t next

  Index of the next chained descriptor.

15.363.1 Detailed Description

Descriptor in the descriptor table.

Definition at line 95 of file virtqueue.

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

15.364 L4virtio::Virtqueue::Desc::Flags Struct Reference

Type for descriptor flags.

Collaboration diagram for L4virtio::Virtqueue::Desc::Flags:

Public Member Functions

- Flags (l4_uint16_t v)

  Make Flags from raw 16bit value.

Data Fields

- l4_uint16_t raw

  raw flags value of a virtio descriptor.

- typedef cxx::Bitfield< decltype(raw), 0, 0 > next_bfm_t

  Part of a descriptor chain which is continued with the next field.
- next_bfm_t::Val next () const

  Get the `next` bits (0 to 0) of `raw`.
- next_bfm_t::Ref next ()

  Get a reference to the `next` bits (0 to 0) of `raw`.

- typedef cxx::Bitfield< decltype(raw), 1, 1 > write_bfm_t

  Block described by this descriptor is writeable.
- write_bfm_t::Val write () const

  Get the `write` bits (1 to 1) of `raw`.
- write_bfm_t::Ref write ()

  Get a reference to the `write` bits (1 to 1) of `raw`.

- typedef cxx::Bitfield< decltype(raw), 2, 2 > indirect_bfm_t

  Indirect descriptor, block contains a list of descriptors.
- indirect_bfm_t::Val indirect () const

  Get the `indirect` bits (2 to 2) of `raw`.
- indirect_bfm_t::Ref indirect ()

  Get a reference to the `indirect` bits (2 to 2) of `raw`.

15.364.1 Detailed Description

Type for descriptor flags.

Definition at line 101 of file virtqueue.

15.364.2 Member Typedef Documentation

15.364.2.1 indirect_bfm_t

```
typedef cxx::Bitfield<decltype( raw ), 2 , 2 > L4virtio::Virtqueue::Desc::Flags::indirect_bfm_t
```

Indirect descriptor, block contains a list of descriptors.

Type to access the `indirect` bits (2 to 2) of `raw`.

Definition at line 114 of file virtqueue.

15.364.2.2 next_bfm_t

```
typedef cxx::Bitfield<decltype( raw ), 0 , 0 > L4virtio::Virtqueue::Desc::Flags::next_bfm_t
```

Part of a descriptor chain which is continued with the next field.

Type to access the `next` bits (0 to 0) of `raw`.

Definition at line 110 of file virtqueue.

15.364.2.3 write_bfm_t

```
typedef cxx::Bitfield<decltype( raw ), 1 , 1 > L4virtio::Virtqueue::Desc::Flags::write_bfm_t
```

Block described by this descriptor is writeable.

Type to access the `write` bits (1 to 1) of `raw`.

Definition at line 112 of file virtqueue.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtqueue

15.365 L4virtio::Virtqueue::Used Class Reference

Used ring.

Collaboration diagram for L4virtio::Virtqueue::Used:

Data Structures

- struct Flags

  flags for the used ring.

Data Fields

- Flags flags

  flags of the used ring.
- l4_uint16_t idx

  index of the last entry in the ring.
- Used_elem ring [ ]

  array of used descriptors.

15.365.1 Detailed Description

Used ring.

Definition at line 181 of file virtqueue.

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

15.366 L4virtio::Virtqueue::Used::Flags Struct Reference

flags for the used ring.

Collaboration diagram for L4virtio::Virtqueue::Used::Flags:

Public Member Functions

- Flags (l4_uint16_t v)

  make Flags from raw value

Data Fields

- l4_uint16_t raw

  raw flags value as specified by virtio.

- typedef cxx::Bitfield< decltype(raw), 0, 0 > no_notify_bfm_t

  host does not want to be notified when new requests have been queued.
- no_notify_bfm_t::Val no_notify () const

  Get the `no_notify` bits (0 to 0) of `raw`.
- no_notify_bfm_t::Ref no_notify ()

  Get a reference to the `no_notify` bits (0 to 0) of `raw`.

15.366.1 Detailed Description

flags for the used ring.

Definition at line 187 of file virtqueue.

15.366.2 Member Typedef Documentation

15.366.2.1 no_notify_bfm_t

```
typedef cxx::Bitfield<decltype( raw ), 0 , 0 > L4virtio::Virtqueue::Used::Flags::no_notify_bfm_t
```

host does not want to be notified when new requests have been queued.

Type to access the `no_notify` bits (0 to 0) of `raw`.

Definition at line 196 of file virtqueue.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtqueue

15.367 L4virtio::Virtqueue::Used_elem Struct Reference

Type of an element of the used ring.

Collaboration diagram for L4virtio::Virtqueue::Used_elem:

Public Member Functions

- Used_elem (l4_uint16_t id, l4_uint32_t len)

  Initialize a used ring element.

Data Fields

- l4_uint32_t id

  descriptor index
- l4_uint32_t len

  length field

15.367.1 Detailed Description

Type of an element of the used ring.

Definition at line 162 of file virtqueue.

15.367.2 Constructor & Destructor Documentation

15.367.2.1 Used_elem()

```
L4virtio::Virtqueue::Used_elem::Used_elem (
            l4_uint16_t id,
            l4_uint32_t len )  [inline]
```

Initialize a used ring element.

Parameters

id	The index of the descriptor to be marked as used.
len	The total bytes written into the buffer of the descriptor chain.

Definition at line 173 of file virtqueue.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtqueue

15.368 l4virtio_block_config_t Struct Reference

Device configuration for block devices.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio_block_config_t:

Data Fields

- l4_uint64_t capacity

  Capacity of device in 512-byte sectors.
- l4_uint32_t size_max

  Maximum size of a single segment.
- l4_uint32_t seg_max

  Maximum number of segments per request.
- l4_uint32_t blk_size

  Block size of underlying disk.

15.368.1 Detailed Description

Device configuration for block devices.

Definition at line 80 of file virtio_block.h.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_block.h

15.369 l4virtio_block_discard_t Struct Reference

Structure used for the write zeroes and discard commands.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio_block_discard_t:

15.369.1 Detailed Description

Structure used for the write zeroes and discard commands.

Definition at line 70 of file virtio_block.h.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_block.h

15.370 l4virtio_block_header_t Struct Reference

Header structure of a request for a block device.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio_block_header_t:

Data Fields

- l4_uint32_t type

  Kind of request, see L4virtio_block_operations.
- l4_uint32_t ioprio

  Priority (unused)
- l4_uint64_t sector

  First sector to read/write.

15.370.1 Detailed Description

Header structure of a request for a block device.

Definition at line 54 of file virtio_block.h.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_block.h

15.371 l4virtio_config_hdr_t Struct Reference

L4-VIRTIO config header, provided in shared data space.

```
#include <virtio.h>
```

Inherited by L4virtio::Device::Config_hdr.

Collaboration diagram for l4virtio_config_hdr_t:

Data Fields

- l4_uint32_t magic

  magic value (must be 'virt').
- l4_uint32_t version

  VIRTIO version.
- l4_uint32_t device

  device ID
- l4_uint32_t vendor

  vendor ID
- l4_uint32_t dev_features

  device features windows selected by device_feature_sel
- l4_uint32_t num_queues

  number of virtqueues
- l4_uint32_t queues_offset

  offset of virtqueue config array
- l4_uint32_t status

  Device status register (read-only).
- l4_uint32_t cfg_driver_notify_index

  W: Event index to be used for config notifications (device to driver)
- l4_uint32_t cfg_device_notify_index

  R: Event index to be used for config notifications (driver to device)
- l4_uint32_t cmd

  L4 specific command register polled by the driver iff supported.

15.371.1 Detailed Description

L4-VIRTIO config header, provided in shared data space.

Definition at line 122 of file virtio.h.

15.371.2 Field Documentation

15.371.2.1 status

```
l4_uint32_t l4virtio_config_hdr_t::status
```

Device status register (read-only).

The register must be written using l4virtio_set_status().

must be at offset 0x70 (virtio-mmio)

Definition at line 161 of file virtio.h.

Referenced by L4virtio::Svr::Dev_config::set_device_needs_reset(), and L4virtio::Svr::Dev_config::set_status().

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio.h

15.372 l4virtio_config_queue_t Struct Reference

Queue configuration entry.

```
#include <virtio.h>
```

Collaboration diagram for l4virtio_config_queue_t:

Data Fields

- l4_uint16_t num_max

  R: maximum number of descriptors supported by this queue.
- l4_uint16_t num

  RW: number of descriptors configured for this queue.
- l4_uint16_t ready

  RW: queue ready flag (read-write)
- l4_uint16_t driver_notify_index

  W: Event index to be used for device notifications (device to driver)
- l4_uint64_t desc_addr

  W: address of descriptor table.
- l4_uint64_t avail_addr

  W: address of available ring.
- l4_uint64_t used_addr

  W: address of used ring.
- l4_uint16_t device_notify_index

  R: Event index to be used by the driver (driver to device)

15.372.1 Detailed Description

Queue configuration entry.

An array of such entries is available at the l4virtio_config_hdr_t::queues_offset in the config data space.

Consistency rules for the queue config are:

- A driver might read `num_max` at any time.
- A driver must write to `num`, `desc_addr`, `avail_addr`, and `used_addr` only when `ready` is zero (0). Values in these fields are validated and used by the device only after successfully setting `ready` to one (1), either by the IPC or by L4VIRTIO_CMD_CFG_QUEUE.
- The value of `device_notify_index` is valid only when `ready` is one.
- The driver might write to `device_notify_index` at any time, however the change is guaranteed to take effect after a successful L4VIRTIO_CMD_CFG_QUEUE or after a config_queue IPC. Note, the change might also have immediate effect, depending on the device implementation.

Definition at line 203 of file virtio.h.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio.h

15.373 l4virtio_input_absinfo_t Struct Reference

Information about the absolute axis in the underlying evdev implementation.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio_input_absinfo_t:

15.373.1 Detailed Description

Information about the absolute axis in the underlying evdev implementation.

Definition at line 44 of file virtio_input.h.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_input.h

15.374 l4virtio_input_config_t Struct Reference

Device configuration for input devices.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio_input_config_t:

15.374.1 Detailed Description

Device configuration for input devices.

Definition at line 67 of file virtio_input.h.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_input.h

15.375 l4virtio_input_devids_t Struct Reference

Device ID information for the device.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio_input_devids_t:

15.375.1 Detailed Description

Device ID information for the device.

Definition at line 56 of file virtio_input.h.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_input.h

15.376 l4virtio_input_event_t Struct Reference

Single event in event or status queue.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio_input_event_t:

15.376.1 Detailed Description

Single event in event or status queue.

Definition at line 85 of file virtio_input.h.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_input.h

Chapter 16

File Documentation

16.1 amd64/l4/util/apic.h File Reference

APIC for AMD64.

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
```

Include dependency graph for apic.h:

16.1.1 Detailed Description

APIC for AMD64.

Definition in file apic.h.

16.2 apic.h

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */
#ifndef __L4_UTIL_APIC_H
#define __L4_UTIL_APIC_H

/*
 * Local APIC programming library
 *
 * For documentation, see
 *
 * "Intel Architecture Software Developer's Manual", Volume 3, chapter 7.5:
 * "Advanced Programmable Interrupt Controller (APIC)"
 *
 * Local APIC is present since
 * - INTEL P6 (PPro)
 * - AMD K7 (Athlon), Model 2
 *
 * In non-SMP-boards, local APIC is disabled, but
 * can be activated by writing to a MSR register.
 * For using APIC see packets cpufreq and l4rtl.
 *
 * See linux/include/asm-i386/i82489.h for further details.
 */

#define APIC_PHYS_BASE              0xFEE00000
#define APIC_MAP_BASE               0xA0200000
#define APIC_BASE_MSR               0x1b

#define APIC_ID                     0x20
#define   GET_APIC_ID(x)              (((x)>>24)&0x0F)
#define APIC_LVR                    0x30
#define   GET_APIC_VERSION(x)         ((x)&0xFF)
#define APIC_TASKPRI                0x80
#define   APIC_TPRI_MASK              0xFF
#define APIC_EOI                    0xB0
#define APIC_LDR                    0xD0
#define   APIC_LDR_MASK               (0xFF<<24)
#define APIC_DFR                    0xE0
#define   SET_APIC_DFR(x)             ((x)<<28)
#define APIC_SPIV                   0xF0
#define APIC_LVTT                   0x320
#define APIC_LVTPC                  0x340
#define APIC_LVT0                   0x350
#define   SET_APIC_TIMER_BASE(x)      (((x)<<18))
#define   APIC_TIMER_BASE_DIV         0x2
#define APIC_LVT1                   0x360
#define APIC_LVTERR                 0x370
#define APIC_TMICT                  0x380
#define APIC_TMCCT                  0x390
#define APIC_TDCR                   0x3E0

#define APIC_LVT_MASKED             (1<<16)
#define APIC_LVT_TIMER_PERIODIC     (1<<17)
#define APIC_TDR_DIV_1              0xB
#define APIC_TDR_DIV_2              0x0
#define APIC_TDR_DIV_4              0x1
#define APIC_TDR_DIV_8              0x2
#define APIC_TDR_DIV_16             0x3
#define APIC_TDR_DIV_32             0x8
#define APIC_TDR_DIV_64             0x9
#define APIC_TDR_DIV_128            0xA

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>

EXTERN_C_BEGIN

/* prototypes */
extern unsigned long apic_map_base;
extern unsigned long apic_timer_divisor;

extern unsigned long l4_scaler_apic_to_ms;

L4_CV void apic_show_registers(void);
L4_CV int  apic_check_working(void);
L4_CV void apic_activate_by_io(void);
L4_CV void apic_timer_set_divisor(int divisor);
```

```
L4_CV unsigned long l4_calibrate_apic(void);

EXTERN_C_END

L4_INLINE void apic_write(unsigned long reg, unsigned long v);
L4_INLINE unsigned long apic_read(unsigned long reg);
L4_INLINE void apic_activate_by_msr(void);
L4_INLINE void apic_deactivate_by_msr(void);
L4_INLINE unsigned long apic_read_phys_address(void);
L4_INLINE int  apic_test_present(void);
L4_INLINE void apic_soft_enable(void);
L4_INLINE void apic_init(unsigned long map_addr);
L4_INLINE void apic_done(void);
L4_INLINE void apic_irq_ack(void);

L4_INLINE void apic_lvt0_disable_irq(void);
L4_INLINE void apic_lvt0_enable_irq(void);
L4_INLINE void apic_lvt1_disable_irq(void);
L4_INLINE void apic_lvt1_enable_irq(void);

L4_INLINE void apic_timer_write(unsigned long value);
L4_INLINE unsigned long apic_timer_read(void);
L4_INLINE void apic_timer_disable_irq(void);
L4_INLINE void apic_timer_enable_irq(void);
L4_INLINE void apic_timer_assign_irq(unsigned long vector);
L4_INLINE void apic_timer_set_periodic(void);
L4_INLINE void apic_timer_set_one_shot(void);

L4_INLINE void apic_perf_disable_irq(void);
L4_INLINE void apic_perf_enable_irq(void);
L4_INLINE void apic_perf_assign_irq(unsigned long vector);


/* write APIC register */
L4_INLINE void
apic_write(unsigned long reg, unsigned long v)
{
  *((volatile unsigned long *)(apic_map_base+reg))=v;
}


/* read APIC register */
L4_INLINE unsigned long
apic_read(unsigned long reg)
{
  return *((volatile unsigned long *)(apic_map_base+reg));
}


/* disable LINT0 */
L4_INLINE void
apic_lvt0_disable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVT0);
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVT0, tmp_val);
}


/* enable LINT0 */
L4_INLINE void
apic_lvt0_enable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVT0);
  tmp_val &= ~(APIC_LVT_MASKED);
  apic_write(APIC_LVT0, tmp_val);
}


/* disable LINT1 */
L4_INLINE void
apic_lvt1_disable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVT1);
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVT1, tmp_val);
}


/* enable LINT1 */
L4_INLINE void
apic_lvt1_enable_irq(void)
{
  unsigned long tmp_val;
```

```
  tmp_val = apic_read(APIC_LVT1);
  tmp_val &= ~(APIC_LVT_MASKED);
  apic_write(APIC_LVT1, tmp_val);
}


/* write APIC timer register */
L4_INLINE void
apic_timer_write(unsigned long value)
{
  apic_read(APIC_TMICT);
  apic_write(APIC_TMICT,value);
}


/* read APIC timer register */
L4_INLINE unsigned long
apic_timer_read(void)
{
  return apic_read(APIC_TMCCT);
}


/* disable IRQ when APIC timer passes 0 */
L4_INLINE void
apic_timer_disable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTT);
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVTT, tmp_val);
}


/* enable IRQ when APIC timer passes 0 */
L4_INLINE void
apic_timer_enable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTT);
  tmp_val &= ~(APIC_LVT_MASKED);
  apic_write(APIC_LVTT, tmp_val);
}


L4_INLINE void
apic_timer_set_periodic(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTT);
  tmp_val |= APIC_LVT_TIMER_PERIODIC;
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVTT, tmp_val);
}


L4_INLINE void
apic_timer_set_one_shot(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTT);
  tmp_val &= ~APIC_LVT_TIMER_PERIODIC;
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVTT, tmp_val);
}


/* set vector of APIC timer irq */
L4_INLINE void
apic_timer_assign_irq(unsigned long vector)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTT);
  tmp_val &= 0xffffff00;
  tmp_val |= vector;
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVTT, tmp_val);
}


/* disable IRQ when performance counter passes 0 */
L4_INLINE void
apic_perf_disable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTPC);
  tmp_val |= APIC_LVT_MASKED;
```

```
  apic_write(APIC_LVTPC, tmp_val);
}


/* enable IRQ when performance counter passes 0 */
L4_INLINE void
apic_perf_enable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTPC);
  tmp_val &= ~(APIC_LVT_MASKED);
  apic_write(APIC_LVTPC, tmp_val);
}


/* set vector of performance counter irq */
L4_INLINE void
apic_perf_assign_irq(unsigned long vector)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTPC);
  tmp_val &= 0xffffff00;
  tmp_val |= vector;
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVTPC, tmp_val);
}


/* activate APIC by writing to appropriate MSR */
L4_INLINE void
apic_activate_by_msr(void)
{
  unsigned long low;
  unsigned long high;

  /* rdmsr */
  asm volatile(".byte 0xf; .byte 0x32\n"
               :"=a" (low),
                "=d" (high)
               :"c" (APIC_BASE_MSR)
               );

  low |= 0x800;                          /* activate APIC */
  low &= 0x00000fff;
  low |= (APIC_PHYS_BASE & 0xfffff000); /* set address */

  /* wrmsr */
  asm volatile(".byte 0xf; .byte 0x30\n"
               :
               :"c" (APIC_BASE_MSR),
                "a" (low),
                "d" (high)
               );
}


/* deactivate APIC by writing to appropriate MSR */
L4_INLINE void
apic_deactivate_by_msr(void)
{
  unsigned long low;
  unsigned long high;

  /* rdmsr */
  asm volatile(".byte 0xf; .byte 0x32\n"
               :"=a" (low),
                "=d" (high)
               :"c" (APIC_BASE_MSR)
               );

  low  &= 0xfffff7ff;                     /* deactivate APIC */

  /* wrmsr */
  asm volatile(".byte 0xf; .byte 0x30\n"
               :
               :"c" (APIC_BASE_MSR),
                "a" (low),
                "d" (high)
               );
}


/* read memory mapped address of apic */
L4_INLINE unsigned long
apic_read_phys_address(void)
{
  unsigned long low;
```

```
  unsigned long high;

  /* rdmsr */
  asm volatile(".byte 0xf; .byte 0x32\n"
               :"=a" (low),
                "=d" (high)
               :"c" (APIC_BASE_MSR)
               );

  return (low  &= 0xfffff000);
}


/* test if APIC present */
L4_INLINE int
apic_test_present(void)
{
  unsigned int dummy;
  unsigned int capability;

  asm volatile("cpuid"
             : "=a" (dummy),
               "=d" (capability)
             : "a" (0x00000001)
             : "cc", "rbx", "rcx");

  return ((capability & 1«9) !=0);
}


L4_INLINE void
apic_soft_enable(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_SPIV);
  tmp_val |= (1«8);          /* enable APIC */
  tmp_val &= ~(1«9);         /* enable Focus Processor Checking */
  tmp_val |= 0xff;            /* Set spurious IRQ vector to 0xff */
  apic_write(APIC_SPIV, tmp_val);
}


L4_INLINE void
apic_init(unsigned long base_addr)
{
  apic_map_base = base_addr;
}


L4_INLINE void
apic_done(void)
{
  apic_map_base = 0;
}


L4_INLINE void
apic_irq_ack(void)
{
  apic_read(APIC_SPIV);
  apic_write(APIC_EOI, 0);
}


#endif /* __L4_UTIL_APIC_H */
```

16.3 x86/l4/util/apic.h File Reference

APIC for X86.

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
```

Include dependency graph for apic.h:

16.3.1 Detailed Description

APIC for X86.

Definition in file apic.h.

16.4 apic.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #ifndef __L4_UTIL_APIC_H
00014 #define __L4_UTIL_APIC_H
00015
00016 /*
00017  * Local APIC programming library
00018  *
00019  * For documentation, see
00020  *
00021  * "Intel Architecture Software Developer's Manual", Volume 3, chapter 7.5:
00022  * "Advanced Programmable Interrupt Controller (APIC)"
00023  *
00024  * Local APIC is present since
00025  * - INTEL P6 (PPro)
00026  * - AMD K7 (Athlon), Model 2
00027  *
00028  * In non-SMP-boards, local APIC is disabled, but
00029  * can be activated by writing to a MSR register.
00030  * For using APIC see packets cpufreq and l4rtl.
00031  *
00032  * See linux/include/asm-i386/i82489.h for further details.
00033  */
00034
00035 #define APIC_PHYS_BASE              0xFEE00000
00036 #define APIC_MAP_BASE               0xA0200000
00037 #define APIC_BASE_MSR               0x1b
00038
00039 #define APIC_ID                     0x20
00040 #define   GET_APIC_ID(x)              (((x)»24)&0x0F)
```

```
#define APIC_LVR                    0x30
#define   GET_APIC_VERSION(x)         ((x)&0xFF)
#define APIC_TASKPRI                0x80
#define   APIC_TPRI_MASK              0xFF
#define APIC_EOI                    0xB0
#define APIC_LDR                    0xD0
#define   APIC_LDR_MASK               (0xFF<<24)
#define APIC_DFR                    0xE0
#define   SET_APIC_DFR(x)             ((x)<<28)
#define APIC_SPIV                   0xF0
#define APIC_LVTT                   0x320
#define APIC_LVTPC                  0x340
#define APIC_LVT0                   0x350
#define   SET_APIC_TIMER_BASE(x)      (((x)<<18))
#define   APIC_TIMER_BASE_DIV         0x2
#define APIC_LVT1                   0x360
#define APIC_LVTERR                 0x370
#define APIC_TMICT                  0x380
#define APIC_TMCCT                  0x390
#define APIC_TDCR                   0x3E0

#define APIC_LVT_MASKED             (1<<16)
#define APIC_LVT_TIMER_PERIODIC     (1<<17)
#define APIC_TDR_DIV_1              0xB
#define APIC_TDR_DIV_2              0x0
#define APIC_TDR_DIV_4              0x1
#define APIC_TDR_DIV_8              0x2
#define APIC_TDR_DIV_16             0x3
#define APIC_TDR_DIV_32             0x8
#define APIC_TDR_DIV_64             0x9
#define APIC_TDR_DIV_128            0xA

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>

EXTERN_C_BEGIN

/* prototypes */
extern unsigned long apic_map_base;
extern unsigned long apic_timer_divisor;

extern unsigned long l4_scaler_apic_to_ms;

L4_CV void apic_show_registers(void);
L4_CV int  apic_check_working(void);
L4_CV void apic_activate_by_io(void);
L4_CV void apic_timer_set_divisor(int divisor);

L4_CV unsigned long l4_calibrate_apic(void);

EXTERN_C_END

L4_INLINE void apic_write(unsigned long reg, unsigned long v);
L4_INLINE unsigned long apic_read(unsigned long reg);
L4_INLINE void apic_activate_by_msr(void);
L4_INLINE void apic_deactivate_by_msr(void);
L4_INLINE unsigned long apic_read_phys_address(void);
L4_INLINE int  apic_test_present(void);
L4_INLINE void apic_soft_enable(void);
L4_INLINE void apic_init(unsigned long map_addr);
L4_INLINE void apic_done(void);
L4_INLINE void apic_irq_ack(void);

L4_INLINE void apic_lvt0_disable_irq(void);
L4_INLINE void apic_lvt0_enable_irq(void);
L4_INLINE void apic_lvt1_disable_irq(void);
L4_INLINE void apic_lvt1_enable_irq(void);

L4_INLINE void apic_timer_write(unsigned long value);
L4_INLINE unsigned long apic_timer_read(void);
L4_INLINE void apic_timer_disable_irq(void);
L4_INLINE void apic_timer_enable_irq(void);
L4_INLINE void apic_timer_assign_irq(unsigned long vector);
L4_INLINE void apic_timer_set_periodic(void);
L4_INLINE void apic_timer_set_one_shot(void);

L4_INLINE void apic_perf_disable_irq(void);
L4_INLINE void apic_perf_enable_irq(void);
L4_INLINE void apic_perf_assign_irq(unsigned long vector);


/* write APIC register */
L4_INLINE void
apic_write(unsigned long reg, unsigned long v)
{
  *((volatile unsigned long *)(apic_map_base+reg))=v;
}
```

```
/* read APIC register */
L4_INLINE unsigned long
apic_read(unsigned long reg)
{
  return *((volatile unsigned long *)(apic_map_base+reg));
}


/* disable LINT0 */
L4_INLINE void
apic_lvt0_disable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVT0);
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVT0, tmp_val);
}


/* enable LINT0 */
L4_INLINE void
apic_lvt0_enable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVT0);
  tmp_val &= ~(APIC_LVT_MASKED);
  apic_write(APIC_LVT0, tmp_val);
}


/* disable LINT1 */
L4_INLINE void
apic_lvt1_disable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVT1);
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVT1, tmp_val);
}


/* enable LINT1 */
L4_INLINE void
apic_lvt1_enable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVT1);
  tmp_val &= ~(APIC_LVT_MASKED);
  apic_write(APIC_LVT1, tmp_val);
}


/* write APIC timer register */
L4_INLINE void
apic_timer_write(unsigned long value)
{
  apic_read(APIC_TMICT);
  apic_write(APIC_TMICT,value);
}


/* read APIC timer register */
L4_INLINE unsigned long
apic_timer_read(void)
{
  return apic_read(APIC_TMCCT);
}


/* disable IRQ when APIC timer passes 0 */
L4_INLINE void
apic_timer_disable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTT);
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVTT, tmp_val);
}


/* enable IRQ when APIC timer passes 0 */
L4_INLINE void
apic_timer_enable_irq(void)
{
  unsigned long tmp_val;
```

```
  tmp_val = apic_read(APIC_LVTT);
  tmp_val &= ~(APIC_LVT_MASKED);
  apic_write(APIC_LVTT, tmp_val);
}


L4_INLINE void
apic_timer_set_periodic(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTT);
  tmp_val |= APIC_LVT_TIMER_PERIODIC;
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVTT, tmp_val);
}


L4_INLINE void
apic_timer_set_one_shot(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTT);
  tmp_val &= ~APIC_LVT_TIMER_PERIODIC;
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVTT, tmp_val);
}


/* set vector of APIC timer irq */
L4_INLINE void
apic_timer_assign_irq(unsigned long vector)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTT);
  tmp_val &= 0xffffff00;
  tmp_val |= vector;
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVTT, tmp_val);
}


/* disable IRQ when performance counter passes 0 */
L4_INLINE void
apic_perf_disable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTPC);
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVTPC, tmp_val);
}


/* enable IRQ when performance counter passes 0 */
L4_INLINE void
apic_perf_enable_irq(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTPC);
  tmp_val &= ~(APIC_LVT_MASKED);
  apic_write(APIC_LVTPC, tmp_val);
}


/* set vector of performance counter irq */
L4_INLINE void
apic_perf_assign_irq(unsigned long vector)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_LVTPC);
  tmp_val &= 0xffffff00;
  tmp_val |= vector;
  tmp_val |= APIC_LVT_MASKED;
  apic_write(APIC_LVTPC, tmp_val);
}


/* activate APIC by writing to appropriate MSR */
L4_INLINE void
apic_activate_by_msr(void)
{
  unsigned long low;
  unsigned long high;

  /* rdmsr */
  asm volatile(".byte 0xf; .byte 0x32\n"
               :"=a" (low),
                "=d" (high)
```

```
              :"c" (APIC_BASE_MSR)
              );

  low |= 0x800;                        /* activate APIC */
  low &= 0x00000fff;
  low |= (APIC_PHYS_BASE & 0xfffff000); /* set address */

  /* wrmsr */
  asm volatile(".byte 0xf; .byte 0x30\n"
              :
              :"c" (APIC_BASE_MSR),
               "a" (low),
               "d" (high)
              );
}


/* deactivate APIC by writing to appropriate MSR */
L4_INLINE void
apic_deactivate_by_msr(void)
{
  unsigned long low;
  unsigned long high;

  /* rdmsr */
  asm volatile(".byte 0xf; .byte 0x32\n"
              :"=a" (low),
               "=d" (high)
              :"c" (APIC_BASE_MSR)
              );

  low  &= 0xfffff7ff;                   /* deactivate APIC */

  /* wrmsr */
  asm volatile(".byte 0xf; .byte 0x30\n"
              :
              :"c" (APIC_BASE_MSR),
               "a" (low),
               "d" (high)
              );
}


/* read memory mapped address of apic */
L4_INLINE unsigned long
apic_read_phys_address(void)
{
  unsigned long low;
  unsigned long high;

  /* rdmsr */
  asm volatile(".byte 0xf; .byte 0x32\n"
              :"=a" (low),
               "=d" (high)
              :"c" (APIC_BASE_MSR)
              );

  return (low  &= 0xfffff000);
}


/* test if APIC present */
L4_INLINE int
apic_test_present(void)
{
  unsigned int dummy;
  unsigned int capability;

  asm volatile("pushl %%ebx ; cpuid ; popl %%ebx"
             : "=a" (dummy),
               "=c" (dummy),
               "=d" (capability)
             : "a" (0x00000001)
             : "cc");

  return ((capability & 1«9) !=0);
}


L4_INLINE void
apic_soft_enable(void)
{
  unsigned long tmp_val;
  tmp_val = apic_read(APIC_SPIV);
  tmp_val |= (1«8);          /* enable APIC */
  tmp_val &= ~(1«9);         /* enable Focus Processor Checking */
  tmp_val |= 0xff;            /* Set spurious IRQ vector to 0xff */
```

```
  apic_write(APIC_SPIV, tmp_val);
}


L4_INLINE void
apic_init(unsigned long base_addr)
{
  apic_map_base = base_addr;
}


L4_INLINE void
apic_done(void)
{
  apic_map_base = 0;
}


L4_INLINE void
apic_irq_ack(void)
{
  apic_read(APIC_SPIV);
  apic_write(APIC_EOI, 0);
}


#endif /* __L4_UTIL_APIC_H */
```

16.5 amd64/l4/util/idt.h File Reference

IDT related functions.

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for idt.h:

Data Structures

- struct l4util_idt_desc_t

  IDT entry.
- struct l4util_idt_header_t

  Header of an IDT table.

16.5.1 Detailed Description

IDT related functions.

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file idt.h.

16.6 idt.h

```
00001
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016
00017 #ifndef __L4UTIL_IDT_H
00018 #define __L4UTIL_IDT_H
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00033 typedef struct
00034 {
00035   l4_uint64_t       a, b;
00036 } __attribute__ ((packed)) l4util_idt_desc_t;
00037
00040 typedef struct
00041 {
00042   l4_uint16_t       limit;
00043   void              *base;
00044   l4util_idt_desc_t desc[0];
00045 } __attribute__ ((packed)) l4util_idt_header_t;
00046
00052 static inline void
00053 l4util_idt_entry(l4util_idt_header_t *idt, int nr, void(*handler)(void))
00054 {
00055   idt->desc[nr].a = (l4_uint64_t)handler & 0x0000ffff;
00056   idt->desc[nr].b = 0x0000ef00 | ((l4_uint64_t)handler & 0xffff0000);
00057 }
00058
00063 static inline void
00064 l4util_idt_init(l4util_idt_header_t *idt, int entries)
00065 {
00066   int i;
00067   idt->limit = entries*8 - 1;
00068   idt->base  = &idt->desc;
00069
00070   for (i=0; i<entries; i++)
00071     l4util_idt_entry(idt, i, 0);
00072 }
00073
00077 static inline void
00078 l4util_idt_load(l4util_idt_header_t *idt)
00079 {
00080   asm volatile ("lidt (%%rax) \n\t" : : "a" (idt));
00081 }
00083 EXTERN_C_END
00084
00085 #endif
00086
```

16.7 x86/l4/util/idt.h File Reference

IDT related functions.

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for idt.h:

Data Structures

- struct l4util_idt_desc_t

  IDT entry.
- struct l4util_idt_header_t

  Header of an IDT table.

16.7.1 Detailed Description

IDT related functions.

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file idt.h.

16.8 idt.h

```
00001
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016
00017 #ifndef __L4UTIL_IDT_H
00018 #define __L4UTIL_IDT_H
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00033 typedef struct
00034 {
00035   l4_uint32_t       a, b;
00036 } __attribute__ ((packed)) l4util_idt_desc_t;
00037
00040 typedef struct
00041 {
00042   l4_uint16_t       limit;
00043   void              *base;
00044   l4util_idt_desc_t desc[0];
00045 } __attribute__ ((packed)) l4util_idt_header_t;
00046
00052 static inline void
00053 l4util_idt_entry(l4util_idt_header_t *idt, int nr, void(*handler)(void))
00054 {
00055   idt->desc[nr].a = (l4_uint32_t)handler & 0x0000ffff;
00056   idt->desc[nr].b = 0x0000ef00 | ((l4_uint32_t)handler & 0xffff0000);
00057 }
00058
00063 static inline void
00064 l4util_idt_init(l4util_idt_header_t *idt, int entries)
00065 {
00066   int i;
00067   idt->limit = entries*8 - 1;
00068   idt->base  = &idt->desc;
00069
00070   for (i=0; i<entries; i++)
00071     l4util_idt_entry(idt, i, 0);
00072 }
00073
00077 static inline void
00078 l4util_idt_load(l4util_idt_header_t *idt)
00079 {
00080   asm volatile ("lidt (%%eax) \n\t" : : "a" (idt));
00081 }
00082
00084 EXTERN_C_END
00085
00086 #endif
00087
```

16.9 amd64/l4/util/perform.h File Reference

Perfomance Monitoring using P5/P6 Measurement Counters.

```
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for perform.h:

16.9.1 Detailed Description

Perfomance Monitoring using P5/P6 Measurement Counters.

Define either CPU_PENTIUM or CPU_P6

Definition in file perform.h.

16.10 perform.h

```
00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015 #ifndef __L4UTIL_PERFORM_H
00016 #define __L4UTIL_PERFORM_H
00017
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020
00021 EXTERN_C_BEGIN
00022
00023 extern const char*strp6pmc_event(l4_uint32_t event);
00024
00025 #ifndef CONFIG_PERFORM_ONLY_PROTOTYPES
00026
00027 #if ! (defined CPU_PENTIUM  ^ defined CPU_P6 ^ defined CPU_K7)
00028
00029 #error You must define your target architecture.
00030 #error Define EITHER CPU_PENTIUM for Intel Pentium or CPU_P6 for Intel PPro/PII/PIII.
00031
00032 #else
00033
00034 /* P5/P6/K7 section */
00035
00036 /* Makros for access to model specific registers (MSR) */
00037
```

```
/* Write the 64-Bit Model Specific Register. First argument is the register,
   second the 64-Bit value. This can only be called at priviledge level 0.
   With L4, the kernel emulates the WRMSR when calling in PL 3.
   */
static inline void l4_i586_wrmsr(unsigned reg,unsigned long long*val){
  unsigned long dummyeax, dummyecx, dummyedx;

  asm volatile(
        ".byte 0xf; .byte 0x30\n" /* wrmsr */
  : "=a" (dummyeax), "=d" (dummyedx), "=c" (dummyecx)
  : "2" (reg), "0" (*(unsigned *)val), "1" (*((unsigned *)val+1))
  );
}

/* Read the 64-Bit Model Specific Register. First argument is the register,
   second the address to a 64-Bit value. This can only be called at
   priviledge level 0.  With L4, the kernel emulates the RDMSR when calling
   in PL 3.
   */
static inline void l4_i586_rdmsr(unsigned reg,unsigned long long*val){
  unsigned dummy;

  asm volatile(
        ".byte 0xf; .byte 0x32\n" /* rdmsr */
  : "=a" (*(unsigned *)val), "=d" (*((unsigned *)val+1)), "=c" (dummy)
  : "2" (reg)
  );
}


#ifdef CPU_PENTIUM
/* Pentium section */

/* functions and events defined here are only usable at Pentium
   Processors. P6 architecture does NOT support this kind of measuring and
   these events. P6 architecture has its own counters and its own events.
   See P6-section for details. */

/* from l4linux/arch/l4-i386/include/perform.h */

static inline void
l4_i586_reset_event_counter(void){
   asm volatile("xor %%rax, %%rax\n"
    "xor %%rdx, %%rdx\n"
    "mov $0x12, %%rcx\n"
    ".byte 0x0f, 0x30\n"
    "movl $0x13, %%rcx\n"
    ".byte 0x0f, 0x30\n"
    : : : "cx", "ax", "dx"
    );
};

static inline void
l4_i586_read_event_counter_long(long long *counter0, long long *counter1)
{
  asm volatile(
         /*        "movl  $0, %%eax\n"
         "movl  $0x11, %%ecx\n"
         ".byte 0x0f, 0x30\n" */* stop event counting */
         "mov  $0x12, %%rcx\n"
         ".byte 0x0f, 0x32\n"
         "mov %%rax, (%%rbx)\n"
         "mov %%rdx, 4(%%rbx)\n"
         "mov $0x13, %%ecx\n"
         ".byte 0x0f, 0x32\n"
         "mov %%rax, (%%rsi)\n"
         "mov %%rdx, 4(%%rsi)\n"
         : /* no output */
         : "b" (counter0), "S" (counter1)
         : "ax", "cx", "dx"
         );
}

static inline void
l4_i586_read_event_counter(int *counter0, int *counter1)
{
  asm volatile("push %%rdx    \n"
         ".byte  0x0f, 0x30 \n"
         "mov  $0x12, %%rcx \n"
         ".byte 0x0f, 0x32  \n"
         "mov  %%rax, %%rbx \n"
         "movl $0x13, %%rcx \n"
         ".byte 0x0f, 0x32\n"
         "popl  %%edx\n"
         : "=b" (*counter0), "=a" (*counter1)
         : "1" (0), "c" (0x11)
         );
```

```
}

static inline void
l4_i586_select_event(int event0, int event1)
{
   asm volatile(".byte 0x0f, 0x30\n"
    :
    :
    "a" (event0 + (event1 « 16)),
    "d" (0),
    "c" (0x11)
    );
};

#define P5_RD_MISS          0x003 /* 000011B */
#define P5_WR_MISS          0x008 /* 000100B */
#define P5_RW_MISS          0x029 /* 101001B */
#define P5_EX_MISS          0x00e /* 001110B */

#define P5_D_WBACK          0x006 /* 000110B */

#define P5_RW_TLB           0x002 /* 00010B */
#define P5_EX_TLB           0x00d /* 01101B */

#define P5_A_STALL          0x01f /* 11111B */
#define P5_W_STALL          0x019 /* 11001B */
#define P5_R_STALL          0x01a /* 11010B */
#define P5_X_STALL          0x01b /* 11011B */

#define P5_AGI_STALL        0x01f /* 11111B */

#define P5_PIPLINE_FLUSH    0x015 /* 10101B */

#define P5_NON_CACHE_RD     0x01e /* 11110B */
#define P5_NCACHE_REFS      0x01e /* 11110B */
#define P5_LOCKED_BUS       0x01c /* 11100B */

#define P5_MEM2PIPE         0x009 /* 01001B */
#define P5_BANK_CONF        0x00a /* 01010B */


#define P5_INSTRS_EX        0x016 /* 10110B */
#define P5_INSTRS_EX_V      0x017 /* 10111B */


#define P5_CNT_NOTHING      (0x00 « 6) /* 00B « 6 */
#define P5_CNT_EVENT_PL0    (0x01 « 6) /* 01B « 6 */
#define P5_CNT_EVENT_PL3    (0x02 « 6) /* 10B « 6 */
#define P5_CNT_EVENT        (0x03 « 6) /* 11B « 6 */
#define P5_CNT_CLOCKS_PL0   (0x05 « 6) /* 101B « 6 */
#define P5_CNT_CLOCKS_PL3   (0x06 « 6) /* 110B « 6 */
#define P5_CNT_CLOCKS       (0x07 « 6) /* 111B « 6 */


#else
#if defined CPU_P6
/* PPro/PII/PIII section */

/*-
 * Copyright (c) 1997 The President and Fellows of Harvard College.
 * All rights reserved.
 * Copyright (c) 1997 Aaron B. Brown.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by Harvard University
 *      and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY HARVARD AND CONTRIBUTORS "AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL HARVARD UNIVERSITY OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
```

```
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */

/*********************************************************************
 ** Symbolic names for counter numbers (used in select_p6counter()) **
 *********************************************************************
 *
 * These correspond in order to the Pentium Pro counters. Add new counters at
 * the end. These agree with the mneumonics in the Pentium Pro Family
 * Developer's Manual, vol 3.
 *
 * Those events marked with a $ require a MESI unit field; those marked with
 * a @ require a self/any unit field. Those marked with a 0 are only supported
 * in counter 0; those marked with 1 are only supported in counter 1.
 */

/* Data cache unit */
#define P6_DATA_MEM_REFS  0x43  /* total memory refs */
#define P6_DCU_LINES_IN   0x45  /* all lines allocated in cache unit */
#define P6_DCU_M_LINES_IN 0x46  /* M lines allocated in cache unit */
#define P6_DCU_M_LINES_OUT  0x47  /* M lines evicted from cache */
#define P6_DCU_MISS_OUTSTANDING 0x48  /* #cycles a miss is outstanding */

/* Instruction fetch unit */
#define P6_IFU_IFETCH   0x80  /* instruction fetches */
#define P6_IFU_IFETCH_MISS  0x81  /* instruction fetch misses */
#define P6_ITLB_MISS    0x85  /* ITLB misses */
#define P6_IFU_MEM_STALL  0x86  /* number of cycles IFU is stalled */
#define P6_ILD_STALL    0x87  /* #stalls in instr length decode */

/* L2 Cache */
#define P6_L2_IFETCH    0x28  /* ($) l2 ifetches */
#define P6_L2_LD    0x29  /* ($) l2 data loads */
#define P6_L2_ST    0x2a  /* ($) l2 data stores */
#define P6_L2_LINES_IN    0x24  /* lines allocated in l2 */
#define P6_L2_LINES_OUT   0x26  /* lines removed from l2 */
#define P6_L2_M_LINES_INM 0x25  /* modified lines allocated in L2 */
#define P6_L2_M_LINES_OUTM  0x27  /* modified lines removed from L2 */
#define P6_L2_RQSTS   0x2e  /* ($) number of l2 requests */
#define P6_L2_ADS   0x21  /* number of l2 addr strobes */
#define P6_L2_DBUS_BUSY   0x22  /* number of data bus busy cycles */
#define P6_L2_DBUS_BUSY_RD  0x23  /* #bus cycles xferring l2->cpu */

/* External bus logic */
#define P6_BUS_DRDY_CLOCKS  0x62  /* (@) #clocks DRDY is asserted */
#define P6_BUS_LOCK_CLOCKS  0x63  /* (@) #clocks LOCK is asserted */
#define P6_BUS_REQ_OUTSTANDING  0x60  /* #bus requests outstanding */
#define P6_BUS_TRAN_BRD   0x65  /* (@) bus burst read txns */
#define P6_BUS_TRAN_RFO   0x66  /* (@) bus read for ownership txns */
#define P6_BUS_TRAN_WB    0x67  /* (@) bus writeback txns */
#define P6_BUS_TRAN_IFETCH  0x68  /* (@) bus instr fetch txns */
#define P6_BUS_TRAN_INVAL 0x69  /* (@) bus invalidate txns */
#define P6_BUS_TRAN_PWR   0x6a  /* (@) bus partial write txns */
#define P6_BUS_TRANS_P    0x6b  /* (@) bus partial txns */
#define P6_BUS_TRANS_IO   0x6c  /* (@) bus I/O txns */
#define P6_BUS_TRAN_DEF   0x6d  /* (@) bus deferred txns */
#define P6_BUS_TRAN_BURST 0x6e  /* (@) bus burst txns */
#define P6_BUS_TRAN_ANY   0x70  /* (@) total bus txns */
#define P6_BUS_TRAN_MEM   0x6f  /* (@) total memory txns */
#define P6_BUS_DATA_RCV   0x64  /* #busclocks CPU is receiving data */
#define P6_BUS_BNR_DRV    0x61  /* #busclocks CPU is driving BNR pin */
#define P6_BUS_HIT_DRV    0x7a  /* #busclocks CPU is driving HIT pin */
#define P6_BUS_HITM_DRV   0x7b  /* #busclocks CPU is driving HITM pin*/
#define P6_BUS_SNOOP_STALL  0x7e  /* #clkcycles bus is snoop-stalled */

/* FPU */
#define P6_FLOPS          0xc1  /* (0) number of FP ops retired */
#define P6_FP_COMP_OPS    0x10  /* (0) computational FPOPS exec'd */
#define P6_FP_ASSIST    0x11  /* (1) FP excep's handled in ucode */
#define P6_MUL      0x12  /* (1) number of FP multiplies */
#define P6_DIV      0x13  /* (1) number of FP divides */
#define P6_CYCLES_DIV_BUSY  0x14  /* (0) number of cycles divider busy */

/* Memory ordering */
#define P6_LD_BLOCKS    0x03  /* number of store buffer blocks */
#define P6_SB_DRAINS    0x04  /* # of store buffer drain cycles */
#define P6_MISALING_MEM_REF 0x05  /* # misaligned data memory refs */

/* Instruction decoding and retirement */
#define P6_INST_RETIRED   0xc0  /* number of instrs retired */
#define P6_UOPS_RETIRED   0xc2  /* number of micro-ops retired */
#define P6_INST_DECODER   0xd0  /* number of instructions decoded */

/* Interrupts */
#define P6_HW_INT_RX    0xc8  /* number of hardware interrupts */
```

```
#define P6_CYCLES_INT_MASKED  0xc6  /* number of cycles hardints masked */
#define P6_CYCLES_INT_PENDING_AND_MASKED 0xc7 /* #cycles masked but pending */

/* Branches */
#define P6_BR_INST_RETIRED  0xc4  /* number of branch instrs retired */
#define P6_BR_MISS_PRED_RETIRED 0xc5  /* number of mispred'd brs retired */
#define P6_BR_TAKEN_RETIRED 0xc9  /* number of taken branches retired */
#define P6_BR_MISS_PRED_TAKEN_RET 0xca  /* #taken mispredictions br's retired*/
#define P6_BR_INST_DECODED      0xe0  /* number of branch instrs decoded */
#define P6_BTB_MISSES   0xe2  /* # of branches that missed in BTB */
#define P6_BR_BOGUS   0xe4  /* number of bogus branches */
#define P6_BACLEARS   0xe6  /* # times BACLEAR is asserted */

/* Stalls */
#define P6_RESOURCE_STALLS  0xa2  /* # resource-related stall cycles */
#define P6_PARTIAL_RAT_STALLS 0xd2  /* # cycles/events for partial stalls*/

/* Segment register loads */
#define P6_SEGMENT_REG_LOADS  0x06  /* number of segment register loads */

/* Clocks */
#define P6_CPU_CLK_UNHALTED 0x79  /* #clocks CPU is not halted */

/* Unit field tags */
#define P6_UNIT_M   0x0800
#define P6_UNIT_E   0x0400
#define P6_UNIT_S   0x0200
#define P6_UNIT_I   0x0100
#define P6_UNIT_MESI    0x0f00

#define P6_UNIT_SELF    0x0000
#define P6_UNIT_ANY   0x2000

/*****************************************************************************
 ** Flag bit definitions (used for the 'flag' field in select_p6counter()) **
 *****************************************************************************
 *
 * The driver accepts fully-formed counter specifications from user-level.
 * The following flags are mneumonics for the bits that get set in the
 * PerfEvtSel0 and PerfEvtSel1 MSR's
 *
 */
#define P6CNT_U  0x010000 /* Monitor user-level events */
#define P6CNT_K  0x020000 /* Monitor kernel-level events */
#define P6CNT_E  0x040000 /* Edge detect: count state transitions */
#define P6CNT_PC 0x080000 /* Pin control: ?? */
#define P6CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
#define P6CNT_F  0x200000 /* Freeze counter (handled in software) */
#define P6CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
#define P6CNT_IV 0x800000 /* Invert counter mask comparison result */

/******************************
 ** Miscellaneous constants **
 ******************************
 *
 * Number of Pentium Pro programable hardware counters.
 */
#define NUM_P6HWC 2

/*****************************************************************************
*
* End of Copyright by Harvard College
*
*****************************************************************************/


#define MSR_P6_EVNTSEL0 0x186
#define MSR_P6_EVNTSEL1 0x187
#define MSR_P6_PERFCTR0 0xc1
#define MSR_P6_PERFCTR1 0xc2

/* P6-specific Makros to manipulate and read counters */

/* Read the 40 bit performance monitoring counter. This requires
   the PCE-flag in CR4 to be set. Otherwise GP0 is raised. Works only
   at P6.
   */
#define l4_i686_rdpmc(cntr, res_p) \
  __asm __volatile(           \
   "mov %2, %%rcx   # put counter number in   \n\
   .byte 0xf; .byte 0x33  # RDPMC instruction   \n\
         mov %%rdx, %1    # High order 32 bits    \n\
         mov %%rax, %0    # Low order 32 bits"    \
  : "=g" (*(int *)(res_p)), "=g" (*(((int *)res_p)+1))  \
  : "g" (cntr)            \
  : "ecx", "eax", "edx")

```

```
static inline l4_uint32_t l4_i686_rdpmc_32(int cntr){
  l4_uint32_t x;

  __asm__ __volatile__(
   ".byte 0xf; .byte 0x33 # RDPMC instruction"
  : "=a" (x)
  : "c" (cntr)
  : "rcx", "rax", "rdx");
  return x;
}

static inline void l4_i686_select_perfctr_event(int counter,
                                                unsigned long long val){
  l4_i586_wrmsr(MSR_P6_EVNTSEL0+counter, &val);
}

static inline void l4_i686_select_perfctr0_event(long long *val){
  asm volatile(
               "mov $MSR_P6_EVNTSEL0, %%rcx\n"
               "mov (%%rbx), %%rax\n"
               "mov 4(%%rbx), %%rdx\n"
               //".byte 0xcc, 0xeb, 0x01, 0x21\n"
               ".byte 0x0f, 0x30\n" // wrmsr
               //".byte 0xcc, 0xeb, 0x01, 0x21\n"
               : /* no output */
               : "b" (val)
               : "ax", "cx", "dx", "bx"
               );

}

/* end of P6 section */
#else

#define K7CNT_U  0x010000 /* Monitor user-level events */
#define K7CNT_K  0x020000 /* Monitor kernel-level events */
#define K7CNT_E  0x040000 /* Edge detect: count state transitions */
#define K7CNT_PC 0x080000 /* Pin control: ?? */
#define K7CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
#define K7CNT_F  0x200000 /* Freeze counter (handled in software) */
#define K7CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
#define K7CNT_IV 0x800000 /* Invert counter mask comparison result */

#define MSR_K7_EVNTSEL0 0xC0010000
#define MSR_K7_EVNTSEL1 0xC0010001
#define MSR_K7_EVNTSEL2 0xC0010002
#define MSR_K7_EVNTSEL3 0xC0010003
#define MSR_K7_PERFCTR0 0xC0010004
#define MSR_K7_PERFCTR1 0xC0010005
#define MSR_K7_PERFCTR2 0xC0010006
#define MSR_K7_PERFCTR3 0xC0010007

#endif

#endif

/* end of P5/P6/K7 section*/
#endif

/* end of not only lib-prototypes section */
#endif

EXTERN_C_END

#endif
```

16.11 x86/l4/util/perform.h File Reference

Perfomance Monitoring using P5/P6 Measurement Counters.

```
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for perform.h:

16.11.1 Detailed Description

Perfomance Monitoring using P5/P6 Measurement Counters.

Define either CPU_PENTIUM or CPU_P6

Definition in file perform.h.

16.12 perform.h

```
00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00010  *               Lars Reuther <reuther@os.inf.tu-dresden.de>
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016
00017 #ifndef __L4UTIL_PERFORM_H
00018 #define __L4UTIL_PERFORM_H
00019
00020 #include <l4/sys/types.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 extern const char*strp6pmc_event(l4_uint32_t event);
00026
00027 #ifndef CONFIG_PERFORM_ONLY_PROTOTYPES
00028
00029 #if ! (defined CPU_PENTIUM  ^ defined CPU_P6 ^ defined CPU_K7)
00030
00031 #error You must define your target architecture.
00032 #error Define EITHER CPU_PENTIUM for Intel Pentium or CPU_P6 for Intel PPro/PII/PIII.
00033
00034 #else
00035
00036 /* P5/P6/K7 section */
00037
```

```
/* Makros for access to model specific registers (MSR) */

/* Write the 64-Bit Model Specific Register. First argument is the register,
   second the 64-Bit value. This can only be called at priviledge level 0.
   With L4, the kernel emulates the WRMSR when calling in PL 3.
   */
static inline void l4_i586_wrmsr(unsigned reg,unsigned long long*val){
  unsigned long dummyeax, dummyecx, dummyedx;

  asm volatile(
        ".byte 0xf; .byte 0x30\n" /* wrmsr */
  : "=a" (dummyeax), "=d" (dummyedx), "=c" (dummyecx)
  : "2" (reg), "0" (*(unsigned *)val), "1" (*((unsigned *)val+1))
  );
}

/* Read the 64-Bit Model Specific Register. First argument is the register,
   second the address to a 64-Bit value. This can only be called at
   priviledge level 0.  With L4, the kernel emulates the RDMSR when calling
   in PL 3.
   */
static inline void l4_i586_rdmsr(unsigned reg,unsigned long long*val){
  unsigned dummy;

  asm volatile(
        ".byte 0xf; .byte 0x32\n" /* rdmsr */
  : "=a" (*(unsigned *)val), "=d" (*((unsigned *)val+1)), "=c" (dummy)
  : "2" (reg)
  );
}


#ifdef CPU_PENTIUM
/* Pentium section */

/* functions and events defined here are only usable at Pentium
   Processors. P6 architecture does NOT support this kind of measuring and
   these events. P6 architecture has its own counters and its own events.
   See P6-section for details. */

/* from l4linux/arch/l4-i386/include/perform.h */

static inline void
l4_i586_reset_event_counter(void){
   asm volatile("xor %%eax, %%eax\n"
    "xor %%edx, %%edx\n"
    "movl $0x12, %%ecx\n"
    ".byte 0x0f, 0x30\n"
    "movl $0x13, %%ecx\n"
    ".byte 0x0f, 0x30\n"
    : : : "cx", "ax", "dx"
    );
};

static inline void
l4_i586_read_event_counter_long(long long *counter0, long long *counter1)
{
  asm volatile(
         /*        "movl  $0, %%eax\n"
         "movl  $0x11, %%ecx\n"
         ".byte 0x0f, 0x30\n" *//* stop event counting */
         "movl $0x12, %%ecx\n"
         ".byte 0x0f, 0x32\n"
         "movl %%eax, (%%ebx)\n"
         "movl %%edx, 4(%%ebx)\n"
         "movl $0x13, %%ecx\n"
         ".byte 0x0f, 0x32\n"
         "movl %%eax, (%%esi)\n"
         "movl %%edx, 4(%%esi)\n"
         : /* no output */
         : "b" (counter0), "S" (counter1)
         : "ax", "cx", "dx"
         );
}

static inline void
l4_i586_read_event_counter(int *counter0, int *counter1)
{
  asm volatile("pushl %%edx\n"
         ".byte 0x0f, 0x30\n"
         "movl $0x12, %%ecx\n"
         ".byte 0x0f, 0x32\n"
         "movl %%eax, %%ebx\n"
         "movl $0x13, %%ecx\n"
         ".byte 0x0f, 0x32\n"
         "popl  %%edx\n"
         : "=b" (*counter0), "=a" (*counter1)
```

```
          : "1" (0), "c" (0x11)
          );
}

static inline void
l4_i586_select_event(int event0, int event1)
{
   asm volatile(".byte 0x0f, 0x30\n"
    :
    :
    "a" (event0 + (event1 « 16)),
    "d" (0),
    "c" (0x11)
    );
};

#define P5_RD_MISS          0x003 /* 000011B */
#define P5_WR_MISS          0x008 /* 000100B */
#define P5_RW_MISS          0x029 /* 101001B */
#define P5_EX_MISS          0x00e /* 001110B */

#define P5_D_WBACK          0x006 /* 000110B */

#define P5_RW_TLB           0x002 /* 00010B */
#define P5_EX_TLB           0x00d /* 01101B */

#define P5_A_STALL          0x01f /* 11111B */
#define P5_W_STALL          0x019 /* 11001B */
#define P5_R_STALL          0x01a /* 11010B */
#define P5_X_STALL          0x01b /* 11011B */

#define P5_AGI_STALL        0x01f /* 11111B */

#define P5_PIPLINE_FLUSH    0x015 /* 10101B */

#define P5_NON_CACHE_RD     0x01e /* 11110B */
#define P5_NCACHE_REFS      0x01e /* 11110B */
#define P5_LOCKED_BUS       0x01c /* 11100B */

#define P5_MEM2PIPE         0x009 /* 01001B */
#define P5_BANK_CONF        0x00a /* 01010B */


#define P5_INSTRS_EX        0x016 /* 10110B */
#define P5_INSTRS_EX_V      0x017 /* 10111B */


#define P5_CNT_NOTHING      (0x00 « 6) /* 00B « 6 */
#define P5_CNT_EVENT_PL0    (0x01 « 6) /* 01B « 6 */
#define P5_CNT_EVENT_PL3    (0x02 « 6) /* 10B « 6 */
#define P5_CNT_EVENT        (0x03 « 6) /* 11B « 6 */
#define P5_CNT_CLOCKS_PL0   (0x05 « 6) /* 101B « 6 */
#define P5_CNT_CLOCKS_PL3   (0x06 « 6) /* 110B « 6 */
#define P5_CNT_CLOCKS       (0x07 « 6) /* 111B « 6 */


#else
#if defined CPU_P6
/* PPro/PII/PIII section */

/*-
 * Copyright (c) 1997 The President and Fellows of Harvard College.
 * All rights reserved.
 * Copyright (c) 1997 Aaron B. Brown.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by Harvard University
 *      and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY HARVARD AND CONTRIBUTORS "AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL HARVARD UNIVERSITY OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
```

```
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */

/*********************************************************************
 ** Symbolic names for counter numbers (used in select_p6counter()) **
 *********************************************************************
 *
 * These correspond in order to the Pentium Pro counters. Add new counters at
 * the end. These agree with the mneumonics in the Pentium Pro Family
 * Developer's Manual, vol 3.
 *
 * Those events marked with a $ require a MESI unit field; those marked with
 * a @ require a self/any unit field. Those marked with a 0 are only supported
 * in counter 0; those marked with 1 are only supported in counter 1.
 */

/* Data cache unit */
#define P6_DATA_MEM_REFS  0x43  /* total memory refs */
#define P6_DCU_LINES_IN   0x45  /* all lines allocated in cache unit */
#define P6_DCU_M_LINES_IN 0x46  /* M lines allocated in cache unit */
#define P6_DCU_M_LINES_OUT  0x47  /* M lines evicted from cache */
#define P6_DCU_MISS_OUTSTANDING 0x48  /* #cycles a miss is outstanding */

/* Instruction fetch unit */
#define P6_IFU_IFETCH   0x80  /* instruction fetches */
#define P6_IFU_IFETCH_MISS  0x81  /* instruction fetch misses */
#define P6_ITLB_MISS    0x85  /* ITLB misses */
#define P6_IFU_MEM_STALL  0x86  /* number of cycles IFU is stalled */
#define P6_ILD_STALL    0x87  /* #stalls in instr length decode */

/* L2 Cache */
#define P6_L2_IFETCH    0x28  /* ($) l2 ifetches */
#define P6_L2_LD    0x29  /* ($) l2 data loads */
#define P6_L2_ST    0x2a  /* ($) l2 data stores */
#define P6_L2_LINES_IN    0x24  /* lines allocated in l2 */
#define P6_L2_LINES_OUT   0x26  /* lines removed from l2 */
#define P6_L2_M_LINES_INM 0x25  /* modified lines allocated in L2 */
#define P6_L2_M_LINES_OUTM  0x27  /* modified lines removed from L2 */
#define P6_L2_RQSTS   0x2e  /* ($) number of l2 requests */
#define P6_L2_ADS   0x21  /* number of l2 addr strobes */
#define P6_L2_DBUS_BUSY   0x22  /* number of data bus busy cycles */
#define P6_L2_DBUS_BUSY_RD  0x23  /* #bus cycles xferring l2->cpu */

/* External bus logic */
#define P6_BUS_DRDY_CLOCKS  0x62  /* (@) #clocks DRDY is asserted */
#define P6_BUS_LOCK_CLOCKS  0x63  /* (@) #clocks LOCK is asserted */
#define P6_BUS_REQ_OUTSTANDING  0x60  /* #bus requests outstanding */
#define P6_BUS_TRAN_BRD   0x65  /* (@) bus burst read txns */
#define P6_BUS_TRAN_RFO   0x66  /* (@) bus read for ownership txns */
#define P6_BUS_TRAN_WB    0x67  /* (@) bus writeback txns */
#define P6_BUS_TRAN_IFETCH  0x68  /* (@) bus instr fetch txns */
#define P6_BUS_TRAN_INVAL 0x69  /* (@) bus invalidate txns */
#define P6_BUS_TRAN_PWR   0x6a  /* (@) bus partial write txns */
#define P6_BUS_TRANS_P    0x6b  /* (@) bus partial txns */
#define P6_BUS_TRANS_IO   0x6c  /* (@) bus I/O txns */
#define P6_BUS_TRAN_DEF   0x6d  /* (@) bus deferred txns */
#define P6_BUS_TRAN_BURST 0x6e  /* (@) bus burst txns */
#define P6_BUS_TRAN_ANY   0x70  /* (@) total bus txns */
#define P6_BUS_TRAN_MEM   0x6f  /* (@) total memory txns */
#define P6_BUS_DATA_RCV   0x64  /* #busclocks CPU is receiving data */
#define P6_BUS_BNR_DRV    0x61  /* #busclocks CPU is driving BNR pin */
#define P6_BUS_HIT_DRV    0x7a  /* #busclocks CPU is driving HIT pin */
#define P6_BUS_HITM_DRV   0x7b  /* #busclocks CPU is driving HITM pin*/
#define P6_BUS_SNOOP_STALL  0x7e  /* #clkcycles bus is snoop-stalled */

/* FPU */
#define P6_FLOPS          0xc1  /* (0) number of FP ops retired */
#define P6_FP_COMP_OPS    0x10  /* (0) computational FPOPS exec'd */
#define P6_FP_ASSIST    0x11  /* (1) FP excep's handled in ucode */
#define P6_MUL      0x12  /* (1) number of FP multiplies */
#define P6_DIV      0x13  /* (1) number of FP divides */
#define P6_CYCLES_DIV_BUSY  0x14  /* (0) number of cycles divider busy */

/* Memory ordering */
#define P6_LD_BLOCKS    0x03  /* number of store buffer blocks */
#define P6_SB_DRAINS    0x04  /* # of store buffer drain cycles */
#define P6_MISALING_MEM_REF 0x05  /* # misaligned data memory refs */

/* Instruction decoding and retirement */
#define P6_INST_RETIRED   0xc0  /* number of instrs retired */
#define P6_UOPS_RETIRED   0xc2  /* number of micro-ops retired */
#define P6_INST_DECODER   0xd0  /* number of instructions decoded */
```

```
/* Interrupts */
#define P6_HW_INT_RX    0xc8  /* number of hardware interrupts */
#define P6_CYCLES_INT_MASKED  0xc6  /* number of cycles hardints masked */
#define P6_CYCLES_INT_PENDING_AND_MASKED 0xc7 /* #cycles masked but pending */

/* Branches */
#define P6_BR_INST_RETIRED  0xc4  /* number of branch instrs retired */
#define P6_BR_MISS_PRED_RETIRED 0xc5  /* number of mispred'd brs retired */
#define P6_BR_TAKEN_RETIRED 0xc9  /* number of taken branches retired */
#define P6_BR_MISS_PRED_TAKEN_RET 0xca  /* #taken mispredictions br's retired*/
#define P6_BR_INST_DECODED      0xe0  /* number of branch instrs decoded */
#define P6_BTB_MISSES   0xe2  /* # of branches that missed in BTB */
#define P6_BR_BOGUS   0xe4  /* number of bogus branches */
#define P6_BACLEARS   0xe6  /* # times BACLEAR is asserted */

/* Stalls */
#define P6_RESOURCE_STALLS  0xa2  /* # resource-related stall cycles */
#define P6_PARTIAL_RAT_STALLS 0xd2  /* # cycles/events for partial stalls*/

/* Segment register loads */
#define P6_SEGMENT_REG_LOADS  0x06  /* number of segment register loads */

/* Clocks */
#define P6_CPU_CLK_UNHALTED 0x79  /* #clocks CPU is not halted */

/* Unit field tags */
#define P6_UNIT_M   0x0800
#define P6_UNIT_E   0x0400
#define P6_UNIT_S   0x0200
#define P6_UNIT_I   0x0100
#define P6_UNIT_MESI    0x0f00

#define P6_UNIT_SELF    0x0000
#define P6_UNIT_ANY   0x2000

/*****************************************************************************
 ** Flag bit definitions (used for the 'flag' field in select_p6counter()) **
 *****************************************************************************
 *
 * The driver accepts fully-formed counter specifications from user-level.
 * The following flags are mneumonics for the bits that get set in the
 * PerfEvtSel0 and PerfEvtSel1 MSR's
 *
 */
#define P6CNT_U  0x010000 /* Monitor user-level events */
#define P6CNT_K  0x020000 /* Monitor kernel-level events */
#define P6CNT_E  0x040000 /* Edge detect: count state transitions */
#define P6CNT_PC 0x080000 /* Pin control: ?? */
#define P6CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
#define P6CNT_F  0x200000 /* Freeze counter (handled in software) */
#define P6CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
#define P6CNT_IV 0x800000 /* Invert counter mask comparison result */

/******************************
 ** Miscellaneous constants **
 ******************************
 *
 * Number of Pentium Pro programable hardware counters.
 */
#define NUM_P6HWC 2

/*******************************************************************************
 *
 * End of Copyright by Harvard College
 *
 *******************************************************************************/


#define MSR_P6_EVNTSEL0 0x186
#define MSR_P6_EVNTSEL1 0x187
#define MSR_P6_PERFCTR0 0xc1
#define MSR_P6_PERFCTR1 0xc2

/* P6-specific Makros to manipulate and read counters */

/* Read the 40 bit performance monitoring counter. This requires
   the PCE-flag in CR4 to be set. Otherwise GP0 is raised. Works only
   at P6.
   */
#define l4_i686_rdpmc(cntr, res_p) \
  __asm __volatile(           \
   "movl %2, %%ecx  # put counter number in   \n\
   .byte 0xf; .byte 0x33  # RDPMC instruction   \n\
         movl %%edx, %1   # High order 32 bits    \n\
         movl %%eax, %0   # Low order 32 bits"    \
  : "=g" (*(int *)(res_p)), "=g" (*(((int *)res_p)+1))  \
  : "g" (cntr)            \
```

```
  : "ecx", "eax", "edx")

static inline l4_uint32_t l4_i686_rdpmc_32(int cntr){
  l4_uint32_t x;

  __asm__ __volatile__(
   ".byte 0xf; .byte 0x33 # RDPMC instruction"
  : "=a" (x)
  : "c" (cntr)
  : "ecx", "eax", "edx");
  return x;
}

static inline void l4_i686_select_perfctr_event(int counter,
                                                unsigned long long val){
  l4_i586_wrmsr(MSR_P6_EVNTSEL0+counter, &val);
}

static inline void l4_i686_select_perfctr0_event(long long *val){
  asm volatile(
               "movl $MSR_P6_EVNTSEL0, %%ecx\n"
               "movl (%%ebx), %%eax\n"
               "movl 4(%%ebx), %%edx\n"
               //".byte 0xcc, 0xeb, 0x01, 0x21\n"
               ".byte 0x0f, 0x30\n" // wrmsr
               //".byte 0xcc, 0xeb, 0x01, 0x21\n"
               : /* no output */
               : "b" (val)
               : "ax", "cx", "dx", "bx"
               );

}

/* end of P6 section */
#else

#define K7CNT_U  0x010000 /* Monitor user-level events */
#define K7CNT_K  0x020000 /* Monitor kernel-level events */
#define K7CNT_E  0x040000 /* Edge detect: count state transitions */
#define K7CNT_PC 0x080000 /* Pin control: ?? */
#define K7CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
#define K7CNT_F  0x200000 /* Freeze counter (handled in software) */
#define K7CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
#define K7CNT_IV 0x800000 /* Invert counter mask comparison result */

#define MSR_K7_EVNTSEL0 0xC0010000
#define MSR_K7_EVNTSEL1 0xC0010001
#define MSR_K7_EVNTSEL2 0xC0010002
#define MSR_K7_EVNTSEL3 0xC0010003
#define MSR_K7_PERFCTR0 0xC0010004
#define MSR_K7_PERFCTR1 0xC0010005
#define MSR_K7_PERFCTR2 0xC0010006
#define MSR_K7_PERFCTR3 0xC0010007

#endif

#endif

/* end of P5/P6/K7 section*/
#endif

/* end of not only lib-prototypes section */
#endif

EXTERN_C_END

#endif
```

16.13 amd64/l4/util/rdtsc.h File Reference

time stamp counter related functions

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/kip.h>
```

Include dependency graph for rdtsc.h:

Functions

- l4_cpu_time_t l4_rdtsc (void)

  Read current value of CPU-internal time stamp counter.
- l4_uint32_t l4_rdtsc_32 (void)

  Read the lest significant 32 bit of the TSC.
- l4_uint64_t l4_rdpmc (int ecx)

  Return current value of CPU-internal performance measurement counter.
- l4_uint32_t l4_rdpmc_32 (int ecx)

  Return the least significant 32 bit of a performance counter.
- l4_uint64_t l4_tsc_to_ns (l4_cpu_time_t tsc)

  Convert time stamp to ns value.
- l4_uint64_t l4_tsc_to_us (l4_cpu_time_t tsc)

  Convert time stamp into micro seconds value.
- void l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)

  Convert time stamp to s.ns value.
- l4_cpu_time_t l4_ns_to_tsc (l4_uint64_t ns)

  Convert nano seconds into CPU ticks.
- void l4_busy_wait_ns (l4_uint64_t ns)

  Wait busy for a small amount of time.
- void l4_busy_wait_us (l4_uint64_t us)

  Wait busy for a small amount of time.
- l4_uint32_t l4_calibrate_tsc (l4_kernel_info_t *kip)

  Determine scalers for time stamp calculations.
- l4_uint32_t l4_tsc_init (l4_kernel_info_t *kip)

  Initialize scaler for TSC calibrations from the kernel.
- l4_uint32_t l4_get_hz (void)

  Get CPU frequency in Hz.

16.13.1 Detailed Description

time stamp counter related functions

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file rdtsc.h.

16.14 rdtsc.h

```
00001
00009 /*
00010  * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00011  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00012  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00013  *     economic rights: Technische Universität Dresden (Germany)
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU Lesser General Public License 2.1.
00016  * Please see the COPYING-LGPL-2.1 file for details.
00017  */
00018
00019 #ifndef __l4_rdtsc_h
00020 #define __l4_rdtsc_h
00021
00027 #include <l4/sys/compiler.h>
00028 #include <l4/sys/l4int.h>
00029 #include <l4/sys/kip.h>
00030
00031 EXTERN_C_BEGIN
00032
00033 /* interface */
00039 extern l4_uint32_t l4_scaler_tsc_to_ns;
00040 extern l4_uint32_t l4_scaler_tsc_to_us;
00041 extern l4_uint32_t l4_scaler_ns_to_tsc;
00042 extern l4_uint32_t l4_scaler_tsc_linux;
00043
00048 L4_INLINE l4_cpu_time_t
00049 l4_rdtsc (void);
00050
00056 L4_INLINE
00057 l4_uint32_t l4_rdtsc_32(void);
00058
00065 L4_INLINE l4_uint64_t
00066 l4_rdpmc (int ecx);
00067
00073 L4_INLINE
00074 l4_uint32_t l4_rdpmc_32(int ecx);
00075
00080 L4_INLINE l4_uint64_t
00081 l4_tsc_to_ns (l4_cpu_time_t tsc);
00082
00087 L4_INLINE l4_uint64_t
00088 l4_tsc_to_us (l4_cpu_time_t tsc);
00089
00095 L4_INLINE void
00096 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns);
00097
00103 L4_INLINE l4_cpu_time_t
00104 l4_ns_to_tsc (l4_uint64_t ns);
00105
00111 L4_INLINE void
00112 l4_busy_wait_ns (l4_uint64_t ns);
00113
00119 L4_INLINE void
00120 l4_busy_wait_us (l4_uint64_t us);
00121
00128 L4_INLINE l4_uint32_t
00129 l4_calibrate_tsc (l4_kernel_info_t *kip);
00130
00144 L4_CV l4_uint32_t
00145 l4_tsc_init (l4_kernel_info_t *kip);
00146
00151 L4_CV l4_uint32_t
00152 l4_get_hz (void);
00153
```

```
EXTERN_C_END

/* implementation */

L4_INLINE l4_uint32_t
l4_calibrate_tsc (l4_kernel_info_t *kip)
{
  return l4_tsc_init(kip);
}

L4_INLINE l4_cpu_time_t
l4_rdtsc (void)
{
  l4_umword_t lo, hi;

  __asm__ __volatile__ ("rdtsc" : "=a"(lo), "=d"(hi));

  return ((l4_cpu_time_t)hi << 32) | lo;
}

L4_INLINE l4_uint64_t
l4_rdpmc (int ecx)
{
  l4_umword_t lo, hi;

  __asm__ __volatile__ ("rdpmc" : "=a"(lo), "=d"(hi) : "c"(ecx));

  return ((l4_cpu_time_t)hi << 32) | lo;
}

L4_INLINE
l4_uint32_t l4_rdtsc_32(void)
{
  l4_umword_t lo, hi;

  __asm__ __volatile__ ("rdtsc" : "=a"(lo), "=d"(hi));

  return lo;
}

L4_INLINE
l4_uint32_t l4_rdpmc_32(int ecx)
{
  l4_umword_t lo, hi;

  __asm__ __volatile__ ("rdpmc" : "=a"(lo), "=d"(hi) : "c"(ecx));

  return lo;
}

L4_INLINE l4_uint64_t
l4_tsc_to_ns (l4_cpu_time_t tsc)
{
    l4_uint64_t ns, dummy;
    __asm__
  ("        \n\t"
   "mulq  %3      \n\t"
         "shrd  $27, %%rdx, %%rax       \n\t"
  :"=a" (ns), "=d"(dummy)
  :"a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_ns)
  );
    return ns;
}

L4_INLINE l4_uint64_t
l4_tsc_to_us (l4_cpu_time_t tsc)
{
    l4_uint64_t ns, dummy;
    __asm__
  ("        \n\t"
   "mulq  %3      \n\t"
         "shrd  $32, %%rdx, %%rax       \n\t"
  :"=a" (ns), "=d" (dummy)
  :"a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_us)
  );
    return ns;
}

L4_INLINE void
l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)
{
    __asm__
  ("        \n\t"
   "mulq  %3      \n\t"
         "shrd  $27, %%rdx, %%rax       \n\t"
         "xorq  %%rdx, %%rdx            \n\t"
   "divq  %4      \n\t"
```

```
  :"=a" (*s), "=&d" (*ns)
  : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_ns),
          "rm"(1000000000ULL)
  );
}

L4_INLINE l4_cpu_time_t
l4_ns_to_tsc (l4_uint64_t ns)
{
    l4_uint64_t tsc, dummy;
    __asm__
        ("                              \n\t"
   "mulq  %3      \n\t"
         "shrd  $27, %%rdx, %%rax       \n\t"
  :"=a" (tsc), "=d" (dummy)
  :"a" (ns), "r" ((l4_uint64_t)l4_scaler_ns_to_tsc)
  );
    return tsc;
}

L4_INLINE void
l4_busy_wait_ns (l4_uint64_t ns)
{
  l4_cpu_time_t stop = l4_rdtsc();
  stop += l4_ns_to_tsc(ns);

  while (l4_rdtsc() < stop)
    ;
}

L4_INLINE void
l4_busy_wait_us (l4_uint64_t us)
{
  l4_cpu_time_t stop = l4_rdtsc ();
  stop += l4_ns_to_tsc(us*1000ULL);

  while (l4_rdtsc() < stop)
    ;
}

#endif /* __l4_rdtsc_h */
```

16.15 x86/l4/util/rdtsc.h File Reference

time stamp counter related functions

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/kip.h>
```

Include dependency graph for rdtsc.h:

Functions

- l4_cpu_time_t l4_rdtsc (void)

  Read current value of CPU-internal time stamp counter.
- l4_uint32_t l4_rdtsc_32 (void)

  Read the lest significant 32 bit of the TSC.
- l4_uint64_t l4_rdpmc (int ecx)

  Return current value of CPU-internal performance measurement counter.
- l4_uint32_t l4_rdpmc_32 (int ecx)

  Return the least significant 32 bit of a performance counter.
- l4_uint64_t l4_tsc_to_ns (l4_cpu_time_t tsc)

  Convert time stamp to ns value.
- l4_uint64_t l4_tsc_to_us (l4_cpu_time_t tsc)

  Convert time stamp into micro seconds value.
- void l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)

  Convert time stamp to s.ns value.
- l4_cpu_time_t l4_ns_to_tsc (l4_uint64_t ns)

  Convert nano seconds into CPU ticks.
- void l4_busy_wait_ns (l4_uint64_t ns)

  Wait busy for a small amount of time.
- void l4_busy_wait_us (l4_uint64_t us)

  Wait busy for a small amount of time.
- l4_uint32_t l4_calibrate_tsc (l4_kernel_info_t *kip)

  Determine scalers for time stamp calculations.
- l4_uint32_t l4_tsc_init (l4_kernel_info_t *kip)

  Initialize scaler for TSC calibrations from the kernel.
- l4_uint32_t l4_get_hz (void)

  Get CPU frequency in Hz.

16.15.1 Detailed Description

time stamp counter related functions

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file rdtsc.h.

16.16 rdtsc.h

```
00001
00009 /*
00010  * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00011  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00012  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00013  *               Jork Löser <jork@os.inf.tu-dresden.de>,
00014  *               Martin Pohlack <mp26@os.inf.tu-dresden.de>
00015  *     economic rights: Technische Universität Dresden (Germany)
00016  * This file is part of TUD:OS and distributed under the terms of the
00017  * GNU Lesser General Public License 2.1.
00018  * Please see the COPYING-LGPL-2.1 file for details.
00019  */
00020
00021 #ifndef __l4_rdtsc_h
00022 #define __l4_rdtsc_h
00023
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/l4int.h>
00031 #include <l4/sys/kip.h>
00032
00033 EXTERN_C_BEGIN
00034
00035 /* interface */
00041 extern l4_uint32_t l4_scaler_tsc_to_ns;
00042 extern l4_uint32_t l4_scaler_tsc_to_us;
00043 extern l4_uint32_t l4_scaler_ns_to_tsc;
00044 extern l4_uint32_t l4_scaler_tsc_linux;
00045
00050 L4_INLINE l4_cpu_time_t
00051 l4_rdtsc (void);
00052
00058 L4_INLINE
00059 l4_uint32_t l4_rdtsc_32(void);
00060
00067 L4_INLINE l4_uint64_t
00068 l4_rdpmc (int ecx);
00069
00075 L4_INLINE
00076 l4_uint32_t l4_rdpmc_32(int ecx);
00077
00082 L4_INLINE l4_uint64_t
00083 l4_tsc_to_ns (l4_cpu_time_t tsc);
00084
00089 L4_INLINE l4_uint64_t
00090 l4_tsc_to_us (l4_cpu_time_t tsc);
00091
00097 L4_INLINE void
00098 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns);
00099
00105 L4_INLINE l4_cpu_time_t
00106 l4_ns_to_tsc (l4_uint64_t ns);
00107
00113 L4_INLINE void
00114 l4_busy_wait_ns (l4_uint64_t ns);
00115
00121 L4_INLINE void
00122 l4_busy_wait_us (l4_uint64_t us);
00123
00130 L4_INLINE l4_uint32_t
00131 l4_calibrate_tsc (l4_kernel_info_t *kip);
00132
00146 L4_CV l4_uint32_t
00147 l4_tsc_init (l4_kernel_info_t *kip);
00148
00153 L4_CV l4_uint32_t
```

```
l4_get_hz (void);

EXTERN_C_END

/* implementation */

L4_INLINE l4_uint32_t
l4_calibrate_tsc (l4_kernel_info_t *kip)
{
  return l4_tsc_init(kip);
}

L4_INLINE l4_cpu_time_t
l4_rdtsc (void)
{
    l4_cpu_time_t v;

    __asm__ __volatile__
  ("        \n\t"
   ".byte 0x0f, 0x31    \n\t"
  /*"rdtsc\n\t"*/
  :
  "=A" (v)
  : /* no inputs */
  );

    return v;
}

/* the same, but only 32 bit. Useful for smaller differences,
   needs less cycles. */
L4_INLINE
l4_uint32_t l4_rdtsc_32(void)
{
  l4_uint32_t x;

  __asm__ __volatile__ (
       ".byte 0x0f, 0x31\n\t" // rdtsc
       : "=a" (x)
       :
       : "edx");

  return x;
}

L4_INLINE l4_uint64_t
l4_rdpmc (int ecx)
{
    l4_cpu_time_t v;

    __asm__ __volatile__ (
   "rdpmc       \n\t"
  :
  "=A" (v)
  : "c" (ecx)
  );

    return v;
}

/* the same, but only 32 bit. Useful for smaller differences,
   needs less cycles. */
L4_INLINE
l4_uint32_t l4_rdpmc_32(int ecx)
{
  l4_uint32_t x;

  __asm__ __volatile__ (
       "rdpmc       \n\t"
       : "=a" (x)
       : "c" (ecx)
       : "edx");

  return x;
}

L4_INLINE l4_uint64_t
l4_tsc_to_ns (l4_cpu_time_t tsc)
{
    l4_uint32_t dummy;
    l4_uint64_t ns;
    __asm__
  ("        \n\t"
   "movl  %%edx, %%ecx    \n\t"
   "mull  %3      \n\t"
   "movl  %%ecx, %%eax    \n\t"
   "movl  %%edx, %%ecx    \n\t"
```

```
    "mull  %3      \n\t"
    "addl  %%ecx, %%eax    \n\t"
    "adcl  $0, %%edx   \n\t"
    "shld  $5, %%eax, %%edx  \n\t"
    "shll  $5, %%eax   \n\t"
   :"=A" (ns),
    "=&c" (dummy)
   :"0" (tsc),
    "g" (l4_scaler_tsc_to_ns)
   );
     return ns;
}

L4_INLINE l4_uint64_t
l4_tsc_to_us (l4_cpu_time_t tsc)
{
     l4_uint32_t dummy;
     l4_uint64_t us;
     __asm__
   ("        \n\t"
    "movl  %%edx, %%ecx    \n\t"
    "mull  %3      \n\t"
    "movl  %%ecx, %%eax    \n\t"
    "movl  %%edx, %%ecx    \n\t"
    "mull  %3      \n\t"
    "addl  %%ecx, %%eax    \n\t"
    "adcl  $0, %%edx   \n\t"
   :"=A" (us),
    "=&c" (dummy)
   :"0" (tsc),
    "g" (l4_scaler_tsc_to_us)
   );
     return us;
}

L4_INLINE void
l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)
{
     l4_uint32_t dummy;
     __asm__
   ("        \n\t"
    "movl  %%edx, %%ecx    \n\t"
    "mull  %4      \n\t"
    "movl  %%ecx, %%eax    \n\t"
    "movl  %%edx, %%ecx    \n\t"
    "mull  %4      \n\t"
    "addl  %%ecx, %%eax    \n\t"
    "adcl  $0, %%edx   \n\t"
    "movl  $1000000000, %%ecx  \n\t"
    "shld  $5, %%eax, %%edx  \n\t"
    "shll  $5, %%eax   \n\t"
    "divl  %%ecx     \n\t"
   :"=a" (*s), "=d" (*ns), "=&c" (dummy)
   : "A" (tsc), "g" (l4_scaler_tsc_to_ns)
   );
}

L4_INLINE l4_cpu_time_t
l4_ns_to_tsc (l4_uint64_t ns)
{
     l4_uint32_t dummy;
     l4_cpu_time_t tsc;
     __asm__
   ("        \n\t"
    "movl  %%edx, %%ecx    \n\t"
    "mull  %3      \n\t"
    "movl  %%ecx, %%eax    \n\t"
    "movl  %%edx, %%ecx    \n\t"
    "mull  %3      \n\t"
    "addl  %%ecx, %%eax    \n\t"
    "adcl  $0, %%edx   \n\t"
    "shld  $5, %%eax, %%edx  \n\t"
    "shll  $5, %%eax   \n\t"
   :"=A" (tsc),
    "=&c" (dummy)
   :"0" (ns),
    "g" (l4_scaler_ns_to_tsc)
   );
     return tsc;
}

L4_INLINE void
l4_busy_wait_ns (l4_uint64_t ns)
{
   l4_cpu_time_t stop = l4_rdtsc();
   stop += l4_ns_to_tsc(ns);

```

```
  while (l4_rdtsc() < stop)
    ;
}

L4_INLINE void
l4_busy_wait_us (l4_uint64_t us)
{
  l4_cpu_time_t stop = l4_rdtsc ();
  stop += l4_ns_to_tsc(us*1000ULL);

  while (l4_rdtsc() < stop)
    ;
}

#endif /* __l4_rdtsc_h */
```

16.17 amd64/l4/util/spin.h File Reference

Spinning for amd64.

`#include <l4/sys/compiler.h>`

Include dependency graph for spin.h:

16.17.1 Detailed Description

Spinning for amd64.

Definition in file spin.h.

16.18 spin.h

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
```

```
 */
#ifndef __l4util_spin_h
#define __l4util_spin_h

#include <l4/sys/compiler.h>

EXTERN_C_BEGIN

L4_CV void l4_spin(int x,int y);
L4_CV void l4_spin_vga(int x,int y);
L4_CV void l4_spin_n_text(int x, int y, int len, const char*s);
L4_CV void l4_spin_n_text_vga(int x, int y, int len, const char*s);

/*****************************************************************************
*                                                                            *
* spin_text()      - spinning wheel at the hercules screen. The given text   *
*                    must be a text constant, no variables or arrays. Its    *
*                    size is determined with the sizeof operator, it's much  *
*                    faster than the strlen function.                        *
* spin_text_vga() - same for vga.                                            *
*                                                                            *
*****************************************************************************/
#define l4_spin_text(x, y, text) \
  l4_spin_n_text((x), (y), sizeof(text)-1, "" text)
#define l4_spin_text_vga(x, y, text) \
  l4_spin_n_text_vga((x), (y), sizeof(text)-1, "" text)

EXTERN_C_END

#endif
```

16.19 x86/l4/util/spin.h File Reference

Spinning for x86.

`#include <l4/sys/compiler.h>`

Include dependency graph for spin.h:

16.19.1 Detailed Description

Spinning for x86.

Definition in file spin.h.

16.20 spin.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #ifndef __l4util_spin_h
00014 #define __l4util_spin_h
00015
00016 #include <l4/sys/compiler.h>
00017
00018 EXTERN_C_BEGIN
00019
00020 L4_CV void l4_spin(int x,int y);
00021 L4_CV void l4_spin_vga(int x,int y);
00022 L4_CV void l4_spin_n_text(int x, int y, int len, const char*s);
00023 L4_CV void l4_spin_n_text_vga(int x, int y, int len, const char*s);
00024
00025 /****************************************************************************
00026 *                                                                          *
00027 * spin_text()     - spinning wheel at the hercules screen. The given text  *
00028 *                   must be a text constant, no variables or arrays. Its   *
00029 *                   size is determined with the sizeof operator, it's much *
00030 *                   faster than the strlen function.                       *
00031 * spin_text_vga() - same for vga.                                          *
00032 *                                                                          *
00033 ****************************************************************************/
00034 #define l4_spin_text(x, y, text) \
00035   l4_spin_n_text((x), (y), sizeof(text)-1, "" text)
00036 #define l4_spin_text_vga(x, y, text) \
00037   l4_spin_n_text_vga((x), (y), sizeof(text)-1, "" text)
00038
00039 EXTERN_C_END
00040
00041 #endif
```

16.21 amd64/l4/util/util.h File Reference

Utilities, amd64 version.

```
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for util.h:

Functions

- l4_timeout_s l4util_micros2l4to (unsigned int mus) L4_NOTHROW

  Calculate l4 timeouts.
- void l4_sleep (int ms) L4_NOTHROW

  Suspend thread for a period of ms milliseconds.
- void l4_sleep_forever (void) L4_NOTHROW)

  Go sleep and never wake up.

16.21.1 Detailed Description

Utilities, amd64 version.

Definition in file util.h.

16.21.2 Function Documentation

16.21.2.1 l4_sleep()

```
void l4_sleep (
            int ms )
```

Suspend thread for a period of *ms* milliseconds.

Parameters

ms	Time in milliseconds

Examples

examples/libs/libirq/async_isr.c, examples/sys/aliens/main.c, examples/sys/singlestep/main.c, and examples/sys/start-with-exc/r

Referenced by L4Re::Namespace::query().

Here is the caller graph for this function:

16.21.2.2 l4util_micros2l4to()

```
l4_timeout_s l4util_micros2l4to (
            unsigned int mus )
```

Calculate l4 timeouts.

Parameters

mus	time in microseconds. Special cases: • 0 - > timeout 0 • ~0U -> timeout NEVER

Returns

the corresponding l4_timeout value

16.22 util.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *              Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *              Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
```

```
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */
#ifndef __UTIL_H
#define __UTIL_H

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/ipc.h>

EXTERN_C_BEGIN

L4_CV l4_timeout_s l4util_micros2l4to(unsigned int mus) L4_NOTHROW;

L4_CV void l4_sleep(int ms) L4_NOTHROW;

/* Suspend thread for a period of \a us microseconds.
 * \param us Time in microseconds
 * WARNING: This function is mostly bogus since the timer resolution of
 *          current L4 implementations is about 1ms! */
L4_CV void l4_usleep(int us) L4_NOTHROW;

L4_INLINE void l4_sleep_forever(void) L4_NOTHROW __attribute__((noreturn));

L4_INLINE void
l4_sleep_forever(void) L4_NOTHROW
{
  for (;;)
    l4_ipc_sleep(L4_IPC_NEVER);
}

static inline void
l4_touch_ro(const void*addr, unsigned size) L4_NOTHROW
{
  const char *bptr, *eptr;

  bptr = (const char*)(((l4_addr_t)addr) & L4_PAGEMASK);
  eptr = (const char*)(((l4_addr_t)addr+size-1) & L4_PAGEMASK);
  for(;bptr<=eptr;bptr+=L4_PAGESIZE){
    asm volatile("or  %0,%%rax \n"
                 :
                 : "m" (*(const unsigned*)bptr)
                 : "rax" );
  }
}


static inline void
l4_touch_rw(const void*addr, unsigned size) L4_NOTHROW
{
  const char *bptr, *eptr;

  bptr = (const char*)(((l4_addr_t)addr) & L4_PAGEMASK);
  eptr = (const char*)(((l4_addr_t)addr+size-1) & L4_PAGEMASK);
  for(;bptr<=eptr;bptr+=L4_PAGESIZE){
    asm volatile("orb $0,%0 \n"
                 :
                 : "m" (*(const unsigned*)bptr)
                 );
  }
}

EXTERN_C_END

#endif
```

16.23 x86/l4/util/util.h File Reference

Utilities for x86.

```
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for util.h:

Functions

- l4_timeout_s l4util_micros2l4to (unsigned int mus) L4_NOTHROW

  Calculate l4 timeouts.
- void l4_sleep (int ms) L4_NOTHROW

  Suspend thread for a period of ms milliseconds.
- void l4_sleep_forever (void) L4_NOTHROW)

  Go sleep and never wake up.

16.23.1 Detailed Description

Utilities for x86.

Definition in file util.h.

16.23.2 Function Documentation

16.23.2.1 l4_sleep()

```
void l4_sleep (
            int ms )
```

Suspend thread for a period of *ms* milliseconds.

Parameters

ms	Time in milliseconds

16.23.2.2 l4util_micros2l4to()

```
l4_timeout_s l4util_micros2l4to (
            unsigned int mus )
```

Calculate l4 timeouts.

Parameters

mus	time in microseconds. Special cases: • 0 - > timeout 0 • ∼0U -> timeout NEVER

Returns

the corresponding l4_timeout value

16.24 util.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00009  *               Jork Löser <jork@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015 #ifndef __UTIL_H
00016 #define __UTIL_H
00017
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020 #include <l4/sys/ipc.h>
00021
00022 EXTERN_C_BEGIN
00023
00030 L4_CV l4_timeout_s l4util_micros2l4to(unsigned int mus) L4_NOTHROW;
00031
00035 L4_CV void l4_sleep(int ms) L4_NOTHROW;
00036
00037 /* Suspend thread for a period of \a us microseconds.
00038  * \param us Time in microseconds
00039  * WARNING: This function is mostly bogus since the timer resolution of
00040  *          current L4 implementations is about 1ms! */
00041 L4_CV void l4_usleep(int us) L4_NOTHROW;
00042
00048 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW __attribute__((noreturn));
00049
00050 L4_INLINE void
00051 l4_sleep_forever(void) L4_NOTHROW
00052 {
00053   for (;;)
```

```
    l4_ipc_sleep(L4_IPC_NEVER);
}

static inline void
l4_touch_ro(const void*addr, unsigned size) L4_NOTHROW
{
  const char *bptr, *eptr;

  bptr = (const char*)(((l4_addr_t)addr) & L4_PAGEMASK);
  eptr = (const char*)(((l4_addr_t)addr+size-1) & L4_PAGEMASK);
  for(;bptr<=eptr;bptr+=L4_PAGESIZE){
    asm volatile("or  %0,%%eax \n"
                 :
                 : "m" (*(const unsigned*)bptr)
                 : "eax" );
  }
}


static inline void
l4_touch_rw(const void*addr, unsigned size) L4_NOTHROW
{
  const char *bptr, *eptr;

  bptr = (const char*)(((l4_addr_t)addr) & L4_PAGEMASK);
  eptr = (const char*)(((l4_addr_t)addr+size-1) & L4_PAGEMASK);
  for(;bptr<=eptr;bptr+=L4_PAGESIZE){
    asm volatile("orb $0,%0 \n"
                 :
                 : "m" (*(const unsigned*)bptr)
                 );
  }
}

EXTERN_C_END

#endif

```

16.25 amd64/l4f/l4/sys/segment.h File Reference

l4f specific fs/gs manipulation

`#include <l4/sys/compiler.h>`

Include dependency graph for segment.h:

Functions

- long fiasco_amd64_set_fs (l4_cap_idx_t thread, l4_umword_t base, l4_utcb_t *utcb)

  Set the base address for the FS segment.
- long fiasco_amd64_set_segment_base (l4_cap_idx_t thread, enum L4_sys_segment segr, l4_umword_t base, l4_utcb_t *utcb)

  Set the base address for a segment.
- long fiasco_gdt_set (l4_cap_idx_t thread, void *desc, unsigned int size, unsigned int entry_number_start, l4_utcb_t *utcb)

  Set GDT segment descriptors.

16.25.1 Detailed Description

l4f specific fs/gs manipulation

Definition in file segment.h.

16.25.2 Function Documentation

16.25.2.1 fiasco_amd64_set_fs()

```
long fiasco_amd64_set_fs (
            l4_cap_idx_t thread,
            l4_umword_t base,
            l4_utcb_t * utcb )  [inline]
```

Set the base address for the FS segment.

Parameters

thread	Thread for which the FS base address shall be modified.
base	Base address.
utcb	UTCB of the caller.

Return values

L4_EOK	Success.
-L4_EINVAL	Invalid base address (`base`).
-L4_ENOSYS	Operation not supported with current kernel configuration.

Note

Calling this function is equivalent to calling `fiasco_amd64_set_segment_base(thread, L4_AMD64_SEGMENT_FS, base, utcb)`.

Definition at line 35 of file segment.h.

16.25.2.2 fiasco_amd64_set_segment_base()

```
long fiasco_amd64_set_segment_base (
            l4_cap_idx_t thread,
            enum L4_sys_segment segr,
            l4_umword_t base,
            l4_utcb_t * utcb )  [inline]
```

Set the base address for a segment.

Parameters

thread	Thread for which the base address of the selected segment shall be modified.
segr	Segment to modify (one of L4_sys_segment).
base	Base address.
utcb	UTCB of the caller.

Return values

L4_EOK	Success.
-L4_EINVAL	Invalid segment (`segr`) or base address (`base`).
-L4_ENOSYS	Operation not supported with current kernel configuration.

Definition at line 43 of file segment.h.

16.26 segment.h

```
00001 #include_next <l4/sys/segment.h>
00002
00008 /*
00009  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #ifndef __L4_SYS__ARCH_AMD64__L4API_L4F__SEGMENT_H__
00026 #define __L4_SYS__ARCH_AMD64__L4API_L4F__SEGMENT_H__
00027
00028 #include <l4/sys/compiler.h>
00029
00030 /****************************************************************************
00031  *** Implementation
00032  ****************************************************************************/
00033
00034 L4_INLINE long
00035 fiasco_amd64_set_fs(l4_cap_idx_t thread, l4_umword_t base, l4_utcb_t *utcb)
00036 {
00037   l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((l4_umword_t)L4_AMD64_SEGMENT_FS
       « 16);
00038   l4_utcb_mr_u(utcb)->mr[1] = base;
00039   return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER),
       utcb);
00040 }
00041
```

```
L4_INLINE long
fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum L4_sys_segment segr,
                              l4_umword_t base, l4_utcb_t *utcb)
{
  l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((l4_umword_t)segr << 16);
  l4_utcb_mr_u(utcb)->mr[1] = base;
  return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER),
 utcb);
}

L4_INLINE long
fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
               unsigned int entry_number_start, l4_utcb_t *utcb)
{
  l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
  l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
  __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
  return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2 + (size / 8), 0, 0),
 L4_IPC_NEVER), utcb);
}

#endif /* ! __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__ */
```

16.27 amd64/l4/sys/segment.h File Reference

Segment handling.

```
#include <l4/sys/ipc.h>
#include <l4/sys/task.h>
#include <l4/sys/thread.h>
```

Include dependency graph for segment.h:

Enumerations

- enum L4_task_ldt_x86_consts { L4_TASK_LDT_X86_ENTRY_SIZE = 8 , L4_TASK_LDT_X86_MAX_ENTRIES , L4_TASK_LDT_X86_ENTRY_SIZE = 8 , L4_TASK_LDT_X86_MAX_ENTRIES }

  Contants for LDT handling.
- enum L4_sys_segment { L4_AMD64_SEGMENT_FS = 0 , L4_AMD64_SEGMENT_GS = 1 }

  Constants for identifying segments.

Functions

- long fiasco_ldt_set (l4_cap_idx_t task, void *ldt, unsigned int num_desc, unsigned int entry_number_start, l4_utcb_t *utcb)

  Set LDT segments descriptors.
- long fiasco_gdt_set (l4_cap_idx_t thread, void *desc, unsigned int size, unsigned int entry_number_start, l4_utcb_t *utcb)

  Set GDT segment descriptors.
- unsigned fiasco_gdt_get_entry_offset (l4_cap_idx_t thread, l4_utcb_t *utcb)

  Return the offset of the entry in the GDT.
- long fiasco_amd64_set_fs (l4_cap_idx_t thread, l4_umword_t base, l4_utcb_t *utcb)

  Set the base address for the FS segment.
- long fiasco_amd64_set_segment_base (l4_cap_idx_t thread, enum L4_sys_segment segr, l4_umword_t base, l4_utcb_t *utcb)

  Set the base address for a segment.
- long fiasco_amd64_segment_info (l4_cap_idx_t thread, unsigned *user_ds, unsigned *user_cs, unsigned *user32_cs, l4_utcb_t *utcb)

  Get segment information.

16.27.1 Detailed Description

Segment handling.

Definition in file segment.h.

16.27.2 Enumeration Type Documentation

16.27.2.1 L4_sys_segment

`enum L4_sys_segment`

Constants for identifying segments.

Enumerator

L4_AMD64_SEGMENT_FS	Constant identifying the FS segment.
L4_AMD64_SEGMENT_GS	Constant identifying the GS segment.

Definition at line 116 of file segment.h.

16.27.2.2 L4_task_ldt_x86_consts

`enum L4_task_ldt_x86_consts`

Contants for LDT handling.

Enumerator

L4_TASK_LDT_X86_ENTRY_SIZE	Size of an LDT entry.
L4_TASK_LDT_X86_MAX_ENTRIES	Maximum number of LDT entries that can be written with one call.
L4_TASK_LDT_X86_ENTRY_SIZE	Size of an LDT entry.
L4_TASK_LDT_X86_MAX_ENTRIES	Maximum number of LDT entries that can be written with one call.

Definition at line 85 of file segment.h.

16.27.3 Function Documentation

16.27.3.1 fiasco_amd64_segment_info()

```
long fiasco_amd64_segment_info (
            l4_cap_idx_t thread,
            unsigned * user_ds,
            unsigned * user_cs,
            unsigned * user32_cs,
            l4_utcb_t * utcb )  [inline]
```

Get segment information.

Parameters

in	*thread*	Thread to get info from.
out	*user_ds*	DS segment selector.
out	*user_cs*	64-bit CS segment selector.
out	*user32_cs*	32-bit CS segment selector.
in	*utcb*	UTCB of the caller.

Returns

System call error

Definition at line 185 of file segment.h.

16.27.3.2 fiasco_amd64_set_fs()

```
long fiasco_amd64_set_fs (
            l4_cap_idx_t thread,
            l4_umword_t base,
            l4_utcb_t * utcb )  [inline]
```

Set the base address for the FS segment.

Parameters

thread	Thread for which the FS base address shall be modified.
base	Base address.
utcb	UTCB of the caller.

Return values

L4_EOK	Success.
-L4_EINVAL	Invalid base address (`base`).
-L4_ENOSYS	Operation not supported with current kernel configuration.

Note

Calling this function is equivalent to calling `fiasco_amd64_set_segment_base(thread, L4_AMD64_SEGMENT_FS, base, utcb)`.

Definition at line 35 of file segment.h.

16.27.3.3 fiasco_amd64_set_segment_base()

```
long fiasco_amd64_set_segment_base (
            l4_cap_idx_t thread,
            enum L4_sys_segment segr,
            l4_umword_t base,
            l4_utcb_t * utcb )  [inline]
```

Set the base address for a segment.

Parameters

thread	Thread for which the base address of the selected segment shall be modified.
segr	Segment to modify (one of L4_sys_segment).
base	Base address.
utcb	UTCB of the caller.

Return values

L4_EOK	Success.
-L4_EINVAL	Invalid segment (`segr`) or base address (`base`).
-L4_ENOSYS	Operation not supported with current kernel configuration.

Definition at line 43 of file segment.h.

16.28 segment.h

```
00001
```

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
/****************************************************************************/
#ifndef __L4_SYS__ARCH_X86__SEGMENT_H__
#define __L4_SYS__ARCH_X86__SEGMENT_H__

#ifndef L4API_l4f
#error This header file can only be used with a L4API version!
#endif

#include <l4/sys/ipc.h>

L4_INLINE long
fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
               unsigned int entry_number_start, l4_utcb_t *utcb);

L4_INLINE long
fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
               unsigned int entry_number_start, l4_utcb_t *utcb);

L4_INLINE unsigned
fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb);

enum L4_task_ldt_x86_consts
{
  L4_TASK_LDT_X86_ENTRY_SIZE = 8,
  L4_TASK_LDT_X86_MAX_ENTRIES
    = (L4_UTCB_GENERIC_DATA_SIZE - 2)
      / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
};

L4_INLINE long
fiasco_amd64_set_fs(l4_cap_idx_t thread, l4_umword_t base, l4_utcb_t *utcb);

enum L4_sys_segment
{
  L4_AMD64_SEGMENT_FS = 0,
  L4_AMD64_SEGMENT_GS = 1
};

L4_INLINE long
fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum L4_sys_segment segr,
                              l4_umword_t base, l4_utcb_t *utcb);

L4_INLINE long
fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
                          unsigned *user_cs, unsigned *user32_cs,
                          l4_utcb_t *utcb);

/****************************************************************************
 *** Implementation
 ****************************************************************************/

#include <l4/sys/task.h>
#include <l4/sys/thread.h>

L4_INLINE long
fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
               unsigned int entry_number_start, l4_utcb_t *utcb)
{
  if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
    return -L4_EINVAL;
  l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
  l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
  __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
                   num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
  return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(L4_PROTO_TASK, 2 + num_desc * 2, 0, 0),
 L4_IPC_NEVER), utcb);
}

L4_INLINE unsigned
fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb)
```

```
{
  l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
  if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
    return -1;
  return l4_utcb_mr_u(utcb)->mr[0];
}

L4_INLINE long
fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
                          unsigned *user_cs, unsigned *user32_cs,
                          l4_utcb_t *utcb)
{
  l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
  int r;

  m->mr[0] = L4_THREAD_AMD64_GET_SEGMENT_INFO_OP;

  r = l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0),
                             L4_IPC_NEVER), utcb);
  if (r < 0)
    return r;

  *user_ds   = m->mr[0];
  *user_cs   = m->mr[1];
  *user32_cs = m->mr[2];

  return 0;
}

#endif /* ! __L4_SYS__ARCH_X86__SEGMENT_H__ */
```

16.29 x86/l4f/l4/sys/segment.h File Reference

l4f specific segment manipulation

`#include <l4/sys/compiler.h>`

Include dependency graph for segment.h:

Functions

- long fiasco_gdt_set (l4_cap_idx_t thread, void *desc, unsigned int size, unsigned int entry_number_start, l4_utcb_t *utcb)

  Set GDT segment descriptors.

16.29.1 Detailed Description

l4f specific segment manipulation

Definition in file segment.h.

16.30 segment.h

```
#include_next <l4/sys/segment.h>

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#ifndef __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__
#define __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__

#include <l4/sys/compiler.h>

/******************************************************************************
 *** Implementation
 ******************************************************************************/

L4_INLINE long
fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
          unsigned int entry_number_start, l4_utcb_t *utcb)
{
  l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
  l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
  __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
  return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2 + (size >> 2), 0, 0),
 L4_IPC_NEVER), utcb);
}

#endif /* ! __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__ */
```

16.31 x86/l4/sys/segment.h File Reference

Segment handling.

```
#include <l4/sys/ipc.h>
#include <l4/sys/task.h>
#include <l4/sys/thread.h>
```

Include dependency graph for segment.h:

Enumerations

- enum L4_task_ldt_x86_consts { L4_TASK_LDT_X86_ENTRY_SIZE = 8 , L4_TASK_LDT_X86_MAX_ENTRIES , L4_TASK_LDT_X86_ENTRY_SIZE = 8 , L4_TASK_LDT_X86_MAX_ENTRIES }

  Contants for LDT handling.

Functions

- long fiasco_ldt_set (l4_cap_idx_t task, void ∗ldt, unsigned int num_desc, unsigned int entry_number_start, l4_utcb_t ∗utcb)

  Set LDT segments descriptors.
- long fiasco_gdt_set (l4_cap_idx_t thread, void ∗desc, unsigned int size, unsigned int entry_number_start, l4_utcb_t ∗utcb)

  Set GDT segment descriptors.
- unsigned fiasco_gdt_get_entry_offset (l4_cap_idx_t thread, l4_utcb_t ∗utcb)

  Return the offset of the entry in the GDT.

16.31.1 Detailed Description

Segment handling.

Definition in file segment.h.

16.31.2 Enumeration Type Documentation

16.31.2.1 L4_task_ldt_x86_consts

enum L4_task_ldt_x86_consts

Contants for LDT handling.

Enumerator

L4_TASK_LDT_X86_ENTRY_SIZE	Size of an LDT entry.
L4_TASK_LDT_X86_MAX_ENTRIES	Maximum number of LDT entries that can be written with one call.
L4_TASK_LDT_X86_ENTRY_SIZE	Size of an LDT entry.
L4_TASK_LDT_X86_MAX_ENTRIES	Maximum number of LDT entries that can be written with one call.

Definition at line 85 of file segment.h.

16.32 segment.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 /*****************************************************************************/
00024 #ifndef __L4_SYS__ARCH_X86__SEGMENT_H__
00025 #define __L4_SYS__ARCH_X86__SEGMENT_H__
00026
00027 #ifndef L4API_l4f
00028 #error This header file can only be used with a L4API version!
00029 #endif
00030
00031 #include <l4/sys/ipc.h>
00032
00050 L4_INLINE long
00051 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00052                unsigned int entry_number_start, l4_utcb_t *utcb);
00053
00069 L4_INLINE long
00070 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00071          unsigned int entry_number_start, l4_utcb_t *utcb);
00072
00079 L4_INLINE unsigned
00080 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb);
00081
00085 enum L4_task_ldt_x86_consts
00086 {
00088   L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00090   L4_TASK_LDT_X86_MAX_ENTRIES
00091     = (L4_UTCB_GENERIC_DATA_SIZE - 2)
```

```
00092       / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00093 };
00094
00095 /*****************************************************************************
00096  *** Implementation
00097  *****************************************************************************/
00098
00099 #include <l4/sys/task.h>
00100 #include <l4/sys/thread.h>
00101
00102 L4_INLINE long
00103 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00104                unsigned int entry_number_start, l4_utcb_t *utcb)
00105 {
00106   if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
00107     return -L4_EINVAL;
00108   l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
00109   l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00110   __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
00111                    num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
00112   return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(L4_PROTO_TASK, 2 + num_desc * 2, 0, 0),
       L4_IPC_NEVER), utcb);
00113 }
00114
00115 L4_INLINE unsigned
00116 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb)
00117 {
00118   l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00119   if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00120     return -1;
00121   return l4_utcb_mr_u(utcb)->mr[0];
00122 }
00123
00124 #endif /* ! __L4_SYS__ARCH_X86__SEGMENT_H__ */
```

16.33 amd64/l4f/l4/util/port_io.h File Reference

Port I/O functions.

`#include <x86/l4f/l4/util/port_io.h>`

Include dependency graph for port_io.h:

16.33.1 Detailed Description

Port I/O functions.

Definition in file port_io.h.

16.34 port_io.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #include <x86/l4f/l4/util/port_io.h>
```

16.35 amd64/l4/util/port_io.h File Reference

Port I/O functions.

`#include <x86/l4/util/port_io.h>`

Include dependency graph for port_io.h:

16.35.1 Detailed Description

Port I/O functions.

Definition in file port_io.h.

16.36 port_io.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #include <x86/l4/util/port_io.h>
```

16.37 x86/l4f/l4/util/port_io.h File Reference

port I/O functions

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <x86/l4/util/port_io.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for port_io.h:

This graph shows which files directly or indirectly include this file:

Functions

- int l4util_ioport_map (l4_cap_idx_t sigma0id, unsigned port_start, unsigned log2size)

  Map a range of I/O ports.

16.37.1 Detailed Description

port I/O functions

Date

06/2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file port_io.h.

16.37.2 Function Documentation

16.37.2.1 l4util_ioport_map()

```
int l4util_ioport_map (
            l4_cap_idx_t sigma0id,
            unsigned port_start,
            unsigned log2size )  [inline]
```

Map a range of I/O ports.

Parameters

sigma0id	I/O port service (sigma0).
port_start	(Start) Port to request.
log2size	Log2size of range to request.

Returns

IPC result: 0 if the range could be successfully mapped on error: IPC failure, or -L4_ENOENT if nothing mapped

Definition at line 56 of file port_io.h.

References l4_buf_regs_t::bdr, l4_buf_regs_t::br, L4_ENOENT, l4_iofpage(), l4_ipc_call(), l4_ipc_error(), L4_IPC_NEVER, L4_ITEM_MAP, l4_msgtag(), l4_msgtag_items(), L4_PROTO_IO_PAGE_FAULT, l4_utcb(), l4_utcb_br(), l4_utcb_mr(), l4_msg_regs_t::mr, and l4_fpage_t::raw.

Here is the call graph for this function:

16.38 port_io.h

```
/*****************************************************************************/
/*****************************************************************************/

/*
 * (c) 2003-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

#ifndef _L4UTIL_PORT_IO_API_H
#define _L4UTIL_PORT_IO_API_H

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>

#include <x86/l4/util/port_io.h>

EXTERN_C_BEGIN

L4_INLINE int
l4util_ioport_map(l4_cap_idx_t sigma0id,
                  unsigned port_start, unsigned log2size);

EXTERN_C_END


/****************************************************************************
 *** Implementation
 ****************************************************************************/

#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>


L4_INLINE int
l4util_ioport_map(l4_cap_idx_t sigma0id,
                  unsigned port_start, unsigned log2size)
{
  l4_fpage_t iofp;
  l4_msgtag_t tag;
```

```
  long err;

  iofp = l4_iofpage(port_start, log2size);
  l4_utcb_mr()->mr[0] = iofp.raw;
  l4_utcb_br()->bdr   = 0;
  l4_utcb_br()->br[0] = L4_ITEM_MAP;
  l4_utcb_br()->br[1] = iofp.raw;
  tag = l4_ipc_call(sigma0id, l4_utcb(),
              l4_msgtag(L4_PROTO_IO_PAGE_FAULT, 1, 0, 0),
                    L4_IPC_NEVER);

  if ((err = l4_ipc_error(tag, l4_utcb())))
    return err;

  return l4_msgtag_items(tag) > 0 ? 0 : -L4_ENOENT;
}

#endif

```

16.39 x86/l4/util/port_io.h File Reference

x86 port I/O

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for port_io.h:

This graph shows which files directly or indirectly include this file:

Functions

- l4_uint8_t l4util_in8 (l4_uint16_t port)

  Read byte from I/O port.
- l4_uint16_t l4util_in16 (l4_uint16_t port)

  Read 16-bit-value from I/O port.
- l4_uint32_t l4util_in32 (l4_uint16_t port)

  Read 32-bit-value from I/O port.
- void l4util_ins8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)

  Read a block of 8-bit-values from I/O ports.
- void l4util_ins16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)

  Read a block of 16-bit-values from I/O ports.
- void l4util_ins32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)

  Read a block of 32-bit-values from I/O ports.
- void l4util_out8 (l4_uint8_t value, l4_uint16_t port)

  Write byte to I/O port.
- void l4util_out16 (l4_uint16_t value, l4_uint16_t port)

  Write 16-bit-value to I/O port.
- void l4util_out32 (l4_uint32_t value, l4_uint16_t port)

  Write 32-bit-value to I/O port.
- void l4util_outs8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)

  Write a block of bytes to I/O port.
- void l4util_outs16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)

  Write a block of 16-bit-values to I/O port.
- void l4util_outs32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)

  Write block of 32-bit-values to I/O port.
- void l4util_iodelay (void)

  delay I/O port access by writing to port 0x80

16.39.1 Detailed Description

x86 port I/O

Date

06/2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file port_io.h.

16.40 port_io.h

```
00001 /*****************************************************************************/
00009 /*****************************************************************************/
00010
00011 /*
00012  * (c) 2003-2009 Author(s)
00013  *     economic rights: Technische Universität Dresden (Germany)
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU Lesser General Public License 2.1.
00016  * Please see the COPYING-LGPL-2.1 file for details.
00017  */
00018
00019 #ifndef _L4UTIL_PORT_IO_H
00020 #define _L4UTIL_PORT_IO_H
00021
00027 /* L4 includes */
00028 #include <l4/sys/l4int.h>
00029 #include <l4/sys/compiler.h>
00030
00031 /*****************************************************************************
00032  *** Prototypes
00033  *****************************************************************************/
00034
00035 EXTERN_C_BEGIN
00046 L4_INLINE l4_uint8_t
00047 l4util_in8(l4_uint16_t port);
00048
00055 L4_INLINE l4_uint16_t
00056 l4util_in16(l4_uint16_t port);
00057
00064 L4_INLINE l4_uint32_t
00065 l4util_in32(l4_uint16_t port);
00066
00074 L4_INLINE void
00075 l4util_ins8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00076
00084 L4_INLINE void
00085 l4util_ins16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00086
00094 L4_INLINE void
00095 l4util_ins32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00096
00103 L4_INLINE void
00104 l4util_out8(l4_uint8_t value, l4_uint16_t port);
00105
00113 L4_INLINE void
00114 l4util_out16(l4_uint16_t value, l4_uint16_t port);
00115
00122 L4_INLINE void
00123 l4util_out32(l4_uint32_t value, l4_uint16_t port);
00124
00132 L4_INLINE void
00133 l4util_outs8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00134
00143 L4_INLINE void
00144 l4util_outs16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00145
00153 L4_INLINE void
00154 l4util_outs32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
```

```
L4_INLINE void
l4util_iodelay(void);

EXTERN_C_END


/*****************************************************************************
 *** Implementation
 *****************************************************************************/

L4_INLINE l4_uint8_t
l4util_in8(l4_uint16_t port)
{
  l4_uint8_t value;
  asm volatile ("inb %w1, %b0" : "=a" (value) : "Nd" (port));
  return value;
}

L4_INLINE l4_uint16_t
l4util_in16(l4_uint16_t port)
{
  l4_uint16_t value;
  asm volatile ("inw %w1, %w0" : "=a" (value) : "Nd" (port));
  return value;
}

L4_INLINE l4_uint32_t
l4util_in32(l4_uint16_t port)
{
  l4_uint32_t value;
  asm volatile ("inl %w1, %0" : "=a" (value) : "Nd" (port));
  return value;
}

L4_INLINE void
l4util_ins8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
{
  l4_umword_t dummy1, dummy2;
  asm volatile ("rep insb" : "=D"(dummy1), "=c"(dummy2)
          : "d" (port), "D" (addr), "c"(count)
          : "memory");
}

L4_INLINE void
l4util_ins16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
{
  l4_umword_t dummy1, dummy2;
  asm volatile ("rep insw" : "=D"(dummy1), "=c"(dummy2)
          : "d" (port), "D" (addr), "c"(count)
          : "memory");
}

L4_INLINE void
l4util_ins32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
{
  l4_umword_t dummy1, dummy2;
  asm volatile ("rep insl" : "=D"(dummy1), "=c"(dummy2)
          : "d" (port), "D" (addr), "c"(count)
          : "memory");
}

L4_INLINE void
l4util_out8(l4_uint8_t value, l4_uint16_t port)
{
  asm volatile ("outb %b0, %w1" : : "a" (value), "Nd" (port));
}

L4_INLINE void
l4util_out16(l4_uint16_t value, l4_uint16_t port)
{
  asm volatile ("outw %w0, %w1" : : "a" (value), "Nd" (port));
}

L4_INLINE void
l4util_out32(l4_uint32_t value, l4_uint16_t port)
{
  asm volatile ("outl %0, %w1" : : "a" (value), "Nd" (port));
}

L4_INLINE void
l4util_outs8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
{
  l4_umword_t dummy1, dummy2;
  asm volatile ("rep outsb" : "=S"(dummy1), "=c"(dummy2)
          : "d" (port), "S" (addr), "c"(count)
          : "memory");
```

```
}

L4_INLINE void
l4util_outs16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
{
  l4_umword_t dummy1, dummy2;
  asm volatile ("rep outsw" : "=S"(dummy1), "=c"(dummy2)
          : "d" (port), "S" (addr), "c"(count)
          : "memory");
}

L4_INLINE void
l4util_outs32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
{
  l4_umword_t dummy1, dummy2;
  asm volatile ("rep outsl" : "=S"(dummy1), "=c"(dummy2)
          : "d" (port), "S" (addr), "c"(count)
          : "memory");
}

L4_INLINE void
l4util_iodelay(void)
{
  asm volatile ("outb %al,$0x80");
}

#endif
```

16.41 arm/l4/sys/linkage.h File Reference

Linkage.

Macros

- #define L4_CV

  Define calling convention.

16.41.1 Detailed Description

Linkage.

Definition in file linkage.h.

16.42 linkage.h

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#ifndef __L4__SYS__ARCH_ARM__LINKAGE_H__
#define __L4__SYS__ARCH_ARM__LINKAGE_H__
```

```

#ifdef __ASSEMBLY__
#ifndef ENTRY
#define ENTRY(name) \
  .globl name; \
  .p2align(2); \
  name:
#endif
#endif

#define L4_FASTCALL(x)  x
#define l4_fastcall

#define L4_CV

#ifdef __PIC__
# define L4_LONG_CALL
#else
# define L4_LONG_CALL __attribute__((long_call))
#endif

#endif /* ! __L4__SYS__ARCH_ARM__LINKAGE_H__ */
```

16.43 amd64/l4/sys/linkage.h File Reference

Linkage.

Macros

- #define L4_CV

  Define calling convention.

16.43.1 Detailed Description

Linkage.

Definition in file linkage.h.

16.44 linkage.h

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#ifndef __L4__SYS__ARCH_AMD64__LINKAGE_H__
#define __L4__SYS__ARCH_AMD64__LINKAGE_H__

#ifdef __ASSEMBLY__

#ifndef ENTRY
```

```
#define ENTRY(name) \
  .globl name; \
  .p2align(2); \
  name:

#endif /* __ASSEMBLY__ */
#endif /* ! ENTRY */

#define L4_FASTCALL(x)     x
#define l4_fastcall

#define L4_CV

#endif /* ! __L4__SYS__ARCH_AMD64__LINKAGE_H__ */
```

16.45 x86/l4/sys/linkage.h File Reference

Linkage.

Macros

- #define L4_CV __attribute__((regparm(0)))

  Define calling convention.

16.45.1 Detailed Description

Linkage.

Definition in file linkage.h.

16.46 linkage.h

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Frank Mehnert <fm3@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#ifndef __L4__SYS__ARCH_X86__LINKAGE_H__
#define __L4__SYS__ARCH_X86__LINKAGE_H__

#ifdef __ASSEMBLY__

#ifndef ENTRY
#define ENTRY(name) \
  .globl name; \
  .p2align(2); \
  name:

#endif /* ! ENTRY */
#endif /* __ASSEMBLY__ */

#define L4_FASTCALL(x)  x __attribute__((regparm(3)))
#define l4_fastcall __attribute__((regparm(3)))

#define L4_CV   __attribute__((regparm(0)))

#endif /* ! __L4__SYS__ARCH_X86__LINKAGE_H__ */
```

16.47 arm/l4/sys/mem_op.h File Reference

Memory access functions (ARM specific)

```
#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>
```

Include dependency graph for mem_op.h:

Enumerations

- enum L4_mem_op_widths { L4_MEM_WIDTH_1BYTE = 0 , L4_MEM_WIDTH_2BYTE = 1 , L4_MEM_WIDTH_4BYTE = 2 }

  Memory access width definitions.

Functions

- unsigned long l4_mem_read (unsigned long virtaddress, unsigned width)

  Read user task memory from kernel privilege level.
- void l4_mem_write (unsigned long virtaddress, unsigned width, unsigned long value)

  Write user task memory from kernel privilege level.
- unsigned long l4_mem_arm_op_call (unsigned long op, unsigned long va, unsigned long width, unsigned long value)

  Implementations.

16.47.1 Detailed Description

Memory access functions (ARM specific)

Date

2010-10

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file mem_op.h.

16.48 mem_op.h

```
00001
00009 /*
00010  * (c) 2010 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction.  Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License.  This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #ifndef __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__
00027 #define __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/syscall_defs.h>
00031
00032 EXTERN_C_BEGIN
00033
00051 enum L4_mem_op_widths
00052 {
00053   L4_MEM_WIDTH_1BYTE = 0,
00054   L4_MEM_WIDTH_2BYTE = 1,
00055   L4_MEM_WIDTH_4BYTE = 2,
00056 };
00057
00070 L4_INLINE unsigned long
00071 l4_mem_read(unsigned long virtaddress, unsigned width);
00072
00085 L4_INLINE void
00086 l4_mem_write(unsigned long virtaddress, unsigned width,
00087              unsigned long value);
00088
00089 enum L4_mem_ops
00090 {
00091   L4_MEM_OP_MEM_READ  = 0x10,
00092   L4_MEM_OP_MEM_WRITE = 0x11,
00093 };
00094
00098 L4_INLINE unsigned long
00099 l4_mem_arm_op_call(unsigned long op,
00100                    unsigned long va,
00101                    unsigned long width,
00102                    unsigned long value);
00103
00106 L4_INLINE unsigned long
00107 l4_mem_arm_op_call(unsigned long op,
00108                    unsigned long va,
00109                    unsigned long width,
00110                    unsigned long value)
00111 {
00112   register unsigned long _op    __asm__ ("r0") = op;
00113   register unsigned long _va    __asm__ ("r1") = va;
00114   register unsigned long _width __asm__ ("r2") = width;
00115   register unsigned long _value __asm__ ("r3") = value;
00116
00117   __asm__ __volatile__
00118     ("@ l4_cache_op_arm_call(start) \n\t"
00119      "mov     lr, pc              \n\t"
00120      "mov     pc, %[sc]             \n\t"
00121      "@ l4_cache_op_arm_call(end)   \n\t"
00122        :
00123   "=r" (_op),
00124   "=r" (_va),
00125   "=r" (_width),
00126         "=r" (_value)
00127        :
00128        [sc] "i" (L4_SYSCALL_MEM_OP),
00129   "0" (_op),
00130   "1" (_va),
00131   "2" (_width),
00132         "3" (_value)
00133        :
00134   "cc", "memory", "lr"
00135        );
00136
00137   return _value;
00138 }
```

```

L4_INLINE unsigned long
l4_mem_read(unsigned long virtaddress, unsigned width)
{
  return l4_mem_arm_op_call(L4_MEM_OP_MEM_READ, virtaddress, width, 0);
}

L4_INLINE void
l4_mem_write(unsigned long virtaddress, unsigned width,
             unsigned long value)
{
  l4_mem_arm_op_call(L4_MEM_OP_MEM_WRITE, virtaddress, width, value);
}

EXTERN_C_END

#endif /* ! __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__ */
```

16.49 arm/l4/sys/vm.h File Reference

ARM virtualization interface.

Data Structures

- struct l4_vm_tz_state

  state structure for TrustZone VMs

16.49.1 Detailed Description

ARM virtualization interface.

Definition in file vm.h.

16.50 vm.h

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

struct l4_vm_tz_state_mode
{
  l4_umword_t sp;
  l4_umword_t lr;
  l4_umword_t spsr;
};

struct l4_vm_tz_state_irq_inject
{
  l4_uint32_t group;
```

```
  l4_uint32_t irqs[8];
};

struct l4_vm_tz_state
{
  l4_umword_t r[13]; // r0 - r12

  l4_umword_t sp_usr;
  l4_umword_t lr_usr;

  struct l4_vm_tz_state_mode irq;

  l4_umword_t r_fiq[5]; // r8 - r12
  struct l4_vm_tz_state_mode fiq;
  struct l4_vm_tz_state_mode abt;
  struct l4_vm_tz_state_mode und;
  struct l4_vm_tz_state_mode svc;

  l4_umword_t pc;
  l4_umword_t cpsr;

  l4_umword_t pending_events;
  l4_uint32_t cpacr;
  l4_umword_t cp10_fpexc;

  l4_umword_t pfs;
  l4_umword_t pfa;
  l4_umword_t exit_reason;

  struct l4_vm_tz_state_irq_inject irq_inject;
};

enum L4_vm_exit_reason
{
  L4_vm_exit_reason_vmm_call   = 1,
  L4_vm_exit_reason_inst_abort = 2,
  L4_vm_exit_reason_data_abort = 3,
  L4_vm_exit_reason_irq        = 4,
  L4_vm_exit_reason_fiq        = 5,
  L4_vm_exit_reason_undef      = 6,
};

L4_INLINE int
l4_vm_tz_irq_inject(struct l4_vm_tz_state *state, unsigned irq);

L4_INLINE int
l4_vm_tz_irq_inject(struct l4_vm_tz_state *state, unsigned irq)
{
  if (irq > sizeof(state->irq_inject.irqs) * 8)
    return -L4_EINVAL;

  unsigned g = irq / 32;
  state->irq_inject.group   |= 1 << g;
  state->irq_inject.irqs[g] |= 1 << (irq & 31);

  return 0;
}
```

16.51 arm/l4/util/bitops_arch.h File Reference

ARM specific implementation of bitops functions.

16.51.1 Detailed Description

ARM specific implementation of bitops functions.

Definition in file bitops_arch.h.

16.52 bitops_arch.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #ifndef __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__
00013 #define __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__
00014
00015
00016 #endif /* ! __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__ */
```

16.53 amd64/l4/util/bitops_arch.h File Reference

amd64 bit manipulation functions

16.53.1 Detailed Description

amd64 bit manipulation functions

Date

07/03/2001

Author

Lars Reuther reuther@os.inf.tu-dresden.de Torsten Frenzel frenzel@os.inf.↩ tu-dresden.de Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file bitops_arch.h.

16.54 bitops_arch.h

```
00001 /****************************************************************************/
00011 /*
00012  * (c) 2000-2009 Author(s)
00013  *     economic rights: Technische Universität Dresden (Germany)
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU Lesser General Public License 2.1.
00016  * Please see the COPYING-LGPL-2.1 file for details.
00017  */
00018
00019 /****************************************************************************/
00020 #ifndef __L4UTIL__INCLUDE__ARCH_AMD64__BITOPS_ARCH_H__
00021 #define __L4UTIL__INCLUDE__ARCH_AMD64__BITOPS_ARCH_H__
00022
00023 EXTERN_C_BEGIN
00024
00025 /****************************************************************************
00026  *** Implementation
00027  ****************************************************************************/
00028
00029 #define __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00030 L4_INLINE void
00031 l4util_set_bit(int b, volatile l4_umword_t * dest)
00032 {
00033   __asm__ __volatile__
00034     (
00035      "lock; bts  %1,%0   \n\t"
00036      :
```

```
      :
      "m"   (*dest),   /* 0 mem, destination operand */
      "Ir"  (b)       /* 1,     bit number */
      :
      "memory", "cc"
      );
}

/* clear bit */
#define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
L4_INLINE void
l4util_clear_bit(int b, volatile l4_umword_t * dest)
{
  __asm__ __volatile__
    (
      "lock; btr  %1,%0   \n\t"
      :
      :
      "m"   (*dest),   /* 0 mem, destination operand */
      "Ir"  (b)        /* 1,     bit number */
      :
      "memory", "cc"
      );
}

/* change bit */
#define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
L4_INLINE void
l4util_complement_bit(int b, volatile l4_umword_t * dest)
{
  __asm__ __volatile__
    (
      "lock; btc  %1,%0   \n\t"
      :
      :
      "m"   (*dest),   /* 0 mem, destination operand */
      "Ir"  (b)        /* 1,     bit number */
      :
      "memory", "cc"
      );
}

/* test bit */
#define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
L4_INLINE int
l4util_test_bit(int b, const volatile l4_umword_t * dest)
{
  l4_int8_t bit;

  __asm__ __volatile__
    (
      "bt    %2,%1   \n\t"
      "setc  %0      \n\t"
      :
      "=r"  (bit)      /* 0,     old bit value */
      :
      "m"   (*dest),   /* 1 mem, destination operand */
      "Ir"  (b)        /* 2,     bit number */
      :
      "memory", "cc"
      );

  return (int)bit;
}


/* bit test and set */
#define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
L4_INLINE int
l4util_bts(int b, volatile l4_umword_t * dest)
{
  l4_int8_t bit;

  __asm__ __volatile__
    (
      "lock; bts  %2,%1   \n\t"
      "setc  %0      \n\t"
      :
      "=r"  (bit)      /* 0,     old bit value */
      :
      "m"   (*dest),   /* 1 mem, destination operand */
      "Ir"  (b)        /* 2,     bit number */
      :
      "memory", "cc"
      );

  return (int)bit;
```

```
}

/* bit test and reset */
#define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
L4_INLINE int
l4util_btr(int b, volatile l4_umword_t * dest)
{
  l4_int8_t bit;

  __asm__ __volatile__
    (
     "lock; btr  %2,%1   \n\t"
     "setc  %0      \n\t"
     :
     "=r"  (bit)      /* 0,     old bit value */
     :
     "m"   (*dest),   /* 1 mem, destination operand */
     "Ir"  (b)        /* 2,     bit number */
     :
     "memory", "cc"
     );

  return (int)bit;
}

/* bit test and complement */
#define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
L4_INLINE int
l4util_btc(int b, volatile l4_umword_t * dest)
{
  l4_int8_t bit;

  __asm__ __volatile__
    (
     "lock; btc   %2,%1   \n\t"
     "setc  %0      \n\t"
     :
     "=r"  (bit)      /* 0,     old bit value */
     :
     "m"   (*dest),   /* 1 mem, destination operand */
     "Ir"  ((l4_umword_t)b)        /* 2,     bit number */
     :
     "memory", "cc"
     );

  return (int)bit;
}

/* bit scan reverse */
#define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
L4_INLINE int
l4util_bsr(l4_umword_t word)
{
  l4_umword_t tmp;

  if (L4_UNLIKELY(word == 0))
    return -1;

  __asm__ __volatile__
    (
     "bsr %1,%0 \n\t"
     :
     "=r" (tmp)       /* 0, index of most significant set bit */
     :
     "r"  (word)      /* 1, argument */
     );

  return tmp;
}

/* bit scan forwad */
#define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
L4_INLINE int
l4util_bsf(l4_umword_t word)
{
  l4_umword_t tmp;

  if (L4_UNLIKELY(word == 0))
    return -1;

  __asm__ __volatile__
    (
     "bsf %1,%0 \n\t"
     :
     "=r" (tmp)       /* 0, index of least significant set bit */
     :
     "r"  (word)      /* 1, argument */
```

```
    );

  return tmp;
}

#define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
L4_INLINE int
l4util_find_first_set_bit(const void * dest, l4_size_t size)
{
  l4_mword_t dummy0, dummy1, res;

  __asm__ __volatile__
    (
     "xor  %%rax,%%rax    \n\t"
     "repe; scasl   \n\t"
     "jz    1f      \n\t"
     "lea  -4(%%rdi),%%rdi  \n\t"
     "bsfq  (%%rdi),%%rax \n"
     "1:      \n\t"
     "sub  %%rbx,%%rdi    \n\t"
     "shl  $3,%%rdi   \n\t"
     "add  %%rdi,%%rax    \n\t"
     :
     "=a" (res), "=&c" (dummy0), "=&D" (dummy1)
     :
     "1" ((size + 31) » 5), "2" (dest), "b" (dest)
     :
     "cc", "memory");

  return res;
}

#define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
L4_INLINE int
l4util_find_first_zero_bit(const void * dest, l4_size_t size)
{
  l4_mword_t dummy0, dummy1, dummy2, res;

  if (!size)
    return 0;

  __asm__ __volatile__
    (
     "mov   $-1,%%rax   \n\t"
     "xor   %%rdx,%%rdx \n\t"
     "repe;  scasl    \n\t"
     "je     1f     \n\t"
     "xor   -4(%%rdi),%%rax \n\t"
     "sub   $4,%%rdi    \n\t"
     "bsf   %%rax,%%rdx \n"
     "1:      \n\t"
     "sub   %[dest],%%rdi \n\t"
     "shl   $3,%%rdi    \n\t"
     "add   %%rdi,%%rdx \n\t"
     :
     "=d" (res), "=&c" (dummy0), "=&D" (dummy1), "=&a" (dummy2)
     :
     "1" ((size + 31) » 5), "2" (dest), [dest] "S" (dest)
     :
     "cc", "memory");

  return res;
}

EXTERN_C_END

#endif /* ! __L4UTIL__INCLUDE__ARCH_AMD64__BITOPS_ARCH_H__ */
```

16.55 x86/l4/util/bitops_arch.h File Reference

x86 bit manipulation functions

16.55.1 Detailed Description

x86 bit manipulation functions

Date

07/03/2001

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file bitops_arch.h.

16.56 bitops_arch.h

```
/*****************************************************************************/
/*
 * (c) 2000-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

/*****************************************************************************/
#ifndef __L4UTIL__INCLUDE__ARCH_X86__BITOPS_ARCH_H__
#define __L4UTIL__INCLUDE__ARCH_X86__BITOPS_ARCH_H__

/*****************************************************************************
 *** Implementation
 *****************************************************************************/

EXTERN_C_BEGIN

/* set bit */
#define __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
L4_INLINE void
l4util_set_bit(int b, volatile l4_umword_t * dest)
{
  __asm__ __volatile__
    (
     "lock; btsl  %1,%0   \n\t"
     :
     :
     "m"   (*dest),   /* 0 mem, destination operand */
     "Ir"  (b)       /* 1,     bit number */
     :
     "memory", "cc"
     );
}

/* clear bit */
#define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
L4_INLINE void
l4util_clear_bit(int b, volatile l4_umword_t * dest)
{
  __asm__ __volatile__
    (
     "lock; btrl  %1,%0   \n\t"
     :
     :
     "m"   (*dest),   /* 0 mem, destination operand */
     "Ir"  (b)        /* 1,     bit number */
     :
     "memory", "cc"
     );
}

/* change bit */
#define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
L4_INLINE void
l4util_complement_bit(int b, volatile l4_umword_t * dest)
{
  __asm__ __volatile__
    (
     "lock; btcl  %1,%0   \n\t"
     :
     :
     "m"   (*dest),   /* 0 mem, destination operand */
     "Ir"  (b)        /* 1,     bit number */
     :
```

```
      "memory", "cc"
      );
}

/* test bit */
#define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
L4_INLINE int
l4util_test_bit(int b, const volatile l4_umword_t * dest)
{
  l4_int8_t bit;

  __asm__ __volatile__
    (
      "btl   %2,%1   \n\t"
      "setc  %0      \n\t"
      :
      "=q"  (bit)      /* 0,     old bit value */
      :
      "m"   (*dest),   /* 1 mem, destination operand */
      "Ir"  (b)        /* 2,     bit number */
      :
      "memory", "cc"
      );

  return (int)bit;
}

/* bit test and set */
#define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
L4_INLINE int
l4util_bts(int b, volatile l4_umword_t * dest)
{
  l4_int8_t bit;

  __asm__ __volatile__
    (
      "lock; btsl  %2,%1   \n\t"
      "setc  %0      \n\t"
      :
      "=q"  (bit)      /* 0,     old bit value */
      :
      "m"   (*dest),   /* 1 mem, destination operand */
      "Ir"  (b)        /* 2,     bit number */
      :
      "memory", "cc"
      );

  return (int)bit;
}

/* bit test and reset */
#define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
L4_INLINE int
l4util_btr(int b, volatile l4_umword_t * dest)
{
  l4_int8_t bit;

  __asm__ __volatile__
    (
      "lock; btrl  %2,%1   \n\t"
      "setc  %0      \n\t"
      :
      "=q"  (bit)      /* 0,     old bit value */
      :
      "m"   (*dest),   /* 1 mem, destination operand */
      "Ir"  (b)        /* 2,     bit number */
      :
      "memory", "cc"
      );

  return (int)bit;
}

/* bit test and complement */
#define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
L4_INLINE int
l4util_btc(int b, volatile l4_umword_t * dest)
{
  l4_int8_t bit;

  __asm__ __volatile__
    (
      "lock; btcl  %2,%1   \n\t"
      "setc  %0      \n\t"
      :
      "=q"  (bit)      /* 0,     old bit value */
      :
```

```
      "m"   (*dest),   /* 1 mem, destination operand */
      "Ir"  (b)        /* 2,     bit number */
      :
      "memory", "cc"
      );

  return (int)bit;
}

/* bit scan reverse */
#define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
L4_INLINE int
l4util_bsr(l4_umword_t word)
{
  int tmp;

  if (L4_UNLIKELY(word == 0))
    return -1;

  __asm__ __volatile__
    (
     "bsrl %1,%0 \n\t"
     :
     "=r" (tmp)       /* 0, index of most significant set bit */
     :
     "r"  (word)      /* 1, argument */
     );

  return tmp;
}

/* bit scan forwad */
#define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
L4_INLINE int
l4util_bsf(l4_umword_t word)
{
  int tmp;

  if (L4_UNLIKELY(word == 0))
    return -1;

  __asm__ __volatile__
    (
     "bsfl %1,%0 \n\t"
     :
     "=r" (tmp)       /* 0, index of least significant set bit */
     :
     "r"  (word)      /* 1, argument */
     );

  return tmp;
}

#define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
L4_INLINE int
l4util_find_first_set_bit(const void * dest, l4_size_t size)
{
  l4_mword_t dummy0, dummy1, res;

  __asm__ __volatile__
    (
     "repe; scasl   \n\t"
     "jz    1f      \n\t"
     "leal  -4(%%edi),%%edi \n\t"
     "bsfl  (%%edi),%%eax \n"
     "1:      \n\t"
     "subl  %%esi,%%edi   \n\t"
     "shll  $3,%%edi    \n\t"
     "addl  %%edi,%%eax   \n\t"
     :
     "=a" (res), "=c" (dummy0), "=D" (dummy1)
     :
     "a"(0), "c" ((size+31) » 5), "D" (dest), "S" (dest)
     :
     "cc", "memory");

  return res;
}

#define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
L4_INLINE int
l4util_find_first_zero_bit(const void * dest, l4_size_t size)
{
  l4_mword_t dummy0, dummy1, dummy2, res;

  if (!size)
    return 0;
```

```

  __asm__ __volatile__
    (
     "repe;  scasl    \n\t"
     "je     1f     \n\t"
     "xorl   -4(%%edi),%%eax  \n\t"
     "subl   $4,%%edi   \n\t"
     "bsfl   %%eax,%%edx  \n"
     "1:      \n\t"
     "subl   %%esi,%%edi  \n\t"
     "shll   $3,%%edi   \n\t"
     "addl   %%edi,%%edx  \n\t"
     :
     "=d" (res), "=c" (dummy0), "=D" (dummy1), "=a" (dummy2)
     :
     "a" (~0), "c" ((size+31) » 5), "d"(0), "D" (dest), "S" (dest)
     :
     "cc", "memory");

  return res;
}

EXTERN_C_END

#endif /* ! __L4UTIL__INCLUDE__ARCH_X86__BITOPS_ARCH_H__ */
```

16.57 arm/l4/util/cpu.h File Reference

CPU related functions.

16.57.1 Detailed Description

CPU related functions.

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file cpu.h.

16.58 cpu.h

```

/*
 * (c) 2004-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

#ifndef __L4_UTIL__ARCH_ARM__CPU_H__
#define __L4_UTIL__ARCH_ARM__CPU_H__

/* Nothing yet */

#endif /* __L4_UTIL__ARCH_ARM__CPU_H__ */
```

16.59 amd64/l4/util/cpu.h File Reference

CPU related functions.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for cpu.h:

Functions

- int l4util_cpu_has_cpuid (void)

  Check whether the CPU supports the "cpuid" instruction.
- unsigned int l4util_cpu_capabilities (void)

  Returns the CPU capabilities if the "cpuid" instruction is available.
- unsigned int l4util_cpu_capabilities_nocheck (void)

  Returns the CPU capabilities.
- void l4util_cpu_cpuid (unsigned long mode, unsigned long ∗eax, unsigned long ∗ebx, unsigned long ∗ecx, unsigned long ∗edx)

  Generic CPUID access function.

16.59.1 Detailed Description

CPU related functions.

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file cpu.h.

16.60 cpu.h

```
00001
00007 /*
00008  * (c) 2004-2009 Author(s)
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU Lesser General Public License 2.1.
00012  * Please see the COPYING-LGPL-2.1 file for details.
00013  */
00014
00015 #ifndef __L4_UTIL_CPU_H
00016 #define __L4_UTIL_CPU_H
00017
00018 #include <l4/sys/compiler.h>
00019
00020 EXTERN_C_BEGIN
00021
00033 L4_INLINE int          l4util_cpu_has_cpuid(void);
00034
00041 L4_INLINE unsigned int l4util_cpu_capabilities(void);
00042
00048 L4_INLINE unsigned int l4util_cpu_capabilities_nocheck(void);
00049
00053 L4_INLINE void
00054 l4util_cpu_cpuid(unsigned long mode,
00055                 unsigned long *eax, unsigned long *ebx,
00056                 unsigned long *ecx, unsigned long *edx);
00057
00059 static inline void
00060 l4util_cpu_pause(void)
00061 {
00062   __asm__ __volatile__ ("rep; nop");
00063 }
00064
00065 L4_INLINE int
00066 l4util_cpu_has_cpuid(void)
00067 {
00068   return 1;
00069 }
00070
00071 L4_INLINE void
00072 l4util_cpu_cpuid(unsigned long mode,
00073                 unsigned long *eax, unsigned long *ebx,
00074                 unsigned long *ecx, unsigned long *edx)
00075 {
00076   asm volatile("cpuid"
00077               : "=a" (*eax),
00078                 "=b" (*ebx),
00079                 "=c" (*ecx),
00080                 "=d" (*edx)
00081               : "a"  (mode)
00082               );
00083 }
00084
00085 L4_INLINE unsigned int
00086 l4util_cpu_capabilities_nocheck(void)
00087 {
00088   unsigned long dummy, capability;
00089
00090   /* get CPU capabilities */
00091   l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);
00092
00093   return capability;
00094 }
00095
00096 L4_INLINE unsigned int
00097 l4util_cpu_capabilities(void)
00098 {
00099   if (!l4util_cpu_has_cpuid())
00100     return 0; /* CPU has not cpuid instruction */
00101
00102   return l4util_cpu_capabilities_nocheck();
00103 }
00104
00105 EXTERN_C_END
00106
00107 #endif
00108
```

16.61 x86/l4/util/cpu.h File Reference

CPU related functions.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for cpu.h:

Functions

- int l4util_cpu_has_cpuid (void)

  Check whether the CPU supports the "cpuid" instruction.
- unsigned int l4util_cpu_capabilities (void)

  Returns the CPU capabilities if the "cpuid" instruction is available.
- unsigned int l4util_cpu_capabilities_nocheck (void)

  Returns the CPU capabilities.
- void l4util_cpu_cpuid (unsigned long mode, unsigned long *eax, unsigned long *ebx, unsigned long *ecx, unsigned long *edx)

  Generic CPUID access function.

16.61.1 Detailed Description

CPU related functions.

Author

Frank Mehnert `fm3@os.inf.tu-dresden.de`

Definition in file cpu.h.

16.62 cpu.h

```
/*
 * (c) 2004-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

#ifndef __L4_UTIL_CPU_H
#define __L4_UTIL_CPU_H

#include <l4/sys/compiler.h>

EXTERN_C_BEGIN

L4_INLINE int          l4util_cpu_has_cpuid(void);

L4_INLINE unsigned int l4util_cpu_capabilities(void);

L4_INLINE unsigned int l4util_cpu_capabilities_nocheck(void);

L4_INLINE void
l4util_cpu_cpuid(unsigned long mode,
                 unsigned long *eax, unsigned long *ebx,
                 unsigned long *ecx, unsigned long *edx);

static inline void
l4util_cpu_pause(void)
{
  __asm__ __volatile__ ("rep; nop");
}

L4_INLINE int
l4util_cpu_has_cpuid(void)
{
  unsigned long eax;

  asm volatile("pushl %%ebx             \t\n"
               "pushfl      \t\n"
               "popl %%eax    \t\n" /* get eflags */
               "movl %%eax, %%ebx \t\n" /* save it */
               "xorl $0x200000, %%eax \t\n" /* toggle ID bit */
               "pushl %%eax   \t\n"
               "popfl     \t\n" /* set again */
               "pushfl      \t\n"
               "popl %%eax    \t\n" /* get it again */
               "xorl %%ebx, %%eax \t\n"
               "pushl %%ebx   \t\n"
               "popfl     \t\n" /* restore saved flags */
               "popl %%ebx              \t\n"
               : "=a" (eax)
               : /* no input */
               );

  return eax & 0x200000;
}

L4_INLINE void
l4util_cpu_cpuid(unsigned long mode,
                 unsigned long *eax, unsigned long *ebx,
                 unsigned long *ecx, unsigned long *edx)
{
  asm volatile("pushl %%ebx      \t\n"
               "cpuid            \t\n"
               "mov %%ebx, %%esi \t\n"
               "popl %%ebx       \t\n"
               : "=a" (*eax),
                 "=S" (*ebx),
                 "=c" (*ecx),
                 "=d" (*edx)
               : "a"  (mode));
}

L4_INLINE unsigned int
l4util_cpu_capabilities_nocheck(void)
{
  unsigned long dummy, capability;

  /* get CPU capabilities */
  l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);

  return capability;
}
```

```
L4_INLINE unsigned int
l4util_cpu_capabilities(void)
{
  if (!l4util_cpu_has_cpuid())
    return 0; /* CPU has not cpuid instruction */

  return l4util_cpu_capabilities_nocheck();
}

EXTERN_C_END

#endif

```

16.63 arm/l4/util/l4_macros.h File Reference

Main function.

16.63.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file l4_macros.h.

16.64 l4_macros.h

```

/*
 * (c) 2006-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

#ifndef _L4UTIL__ARCH_ARM__L4_MACROS_H
#define _L4UTIL__ARCH_ARM__L4_MACROS_H

#include_next <l4/util/l4_macros.h>

#ifndef l4_addr_fmt
#  define l4_addr_fmt   "%08lx"
#endif

#endif /* !_L4UTIL__ARCH_ARM__L4_MACROS_H */
```

16.65 amd64/l4/util/l4_macros.h File Reference

Main function.

16.65.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file l4_macros.h.

16.66 l4_macros.h

```
00001
00008 /*
00009  * (c) 2006-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016 #ifndef _L4UTIL__ARCH_AMD64__L4_MACROS_H
00017 #define _L4UTIL__ARCH_AMD64__L4_MACROS_H
00018
00019 #include_next <l4/util/l4_macros.h>
00020
00021 #ifndef l4_addr_fmt
00022 #  define l4_addr_fmt   "%016lx"
00023 #endif
00024
00025 #endif /* !_L4UTIL__ARCH_AMD64__L4_MACROS_H */
```

16.67 l4/util/l4_macros.h File Reference

Some useful generic macros, L4f version.

16.67.1 Detailed Description

Some useful generic macros, L4f version.

Date

11/12/2002

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file l4_macros.h.

16.68 l4_macros.h

```
/*****************************************************************************/
/*
 * (c) 2000-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

/*****************************************************************************/

#pragma once

/*****************************************************************************
 *** generic macros
 *****************************************************************************/

/* generate L4 thread id printf string */
#ifndef l4util_idstr
#  define l4util_idfmt         "%lx"
#  define l4util_idfmt_adjust  "%04lx"
#  define l4util_idstr(tid)    (tid » L4_CAP_SHIFT)
#endif

```

16.69 x86/l4/util/l4_macros.h File Reference

Main function.

16.69.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file l4_macros.h.

16.70 l4_macros.h

```

/*
 * (c) 2006-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */


#ifndef _L4UTIL__ARCH_X86__L4_MACROS_H
#define _L4UTIL__ARCH_X86__L4_MACROS_H

#include_next <l4/util/l4_macros.h>

#ifndef l4_addr_fmt
#  define l4_addr_fmt   "%08lx"
#endif

#endif /* !_L4UTIL__ARCH_X86__L4_MACROS_H */
```

16.71 arm/l4/util/mbi_argv.h File Reference

Multiboot.

`#include <x86/l4/util/mbi_argv.h>`
Include dependency graph for mbi_argv.h:

16.71.1 Detailed Description

Multiboot.

Definition in file mbi_argv.h.

16.72 mbi_argv.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 /* If this persists, move it to a generic place */
00013 #include <x86/l4/util/mbi_argv.h>
```

16.73 amd64/l4/util/mbi_argv.h File Reference

command line handling

```
#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for mbi_argv.h:

16.73.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file mbi_argv.h.

16.74 mbi_argv.h

```
/*
 * (c) 2003-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

#ifndef L4UTIL_MBI_ARGV
#define L4UTIL_MBI_ARGV

#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>

EXTERN_C_BEGIN

L4_CV void l4util_mbi_to_argv(l4_mword_t flag, l4util_mb_info_t *mbi);

extern int  l4util_argc;
extern char *l4util_argv[];

EXTERN_C_END

#endif
```

16.75 x86/l4/util/mbi_argv.h File Reference

command line handling

```
#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for mbi_argv.h:

This graph shows which files directly or indirectly include this file:

16.75.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file mbi_argv.h.

16.76 mbi_argv.h

```
00001
00008 /*
00009  * (c) 2003-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016 #ifndef L4UTIL_MBI_ARGV
00017 #define L4UTIL_MBI_ARGV
00018
00019 #include <l4/sys/l4int.h>
00020 #include <l4/util/mb_info.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 void l4util_mbi_to_argv(l4_mword_t flag, l4util_mb_info_t *mbi);
00026
00027 extern int  l4util_argc;
00028 extern char *l4util_argv[];
00029
00030 EXTERN_C_END
00031
00032 #endif
00033
```

16.77 arm/l4f/l4/sys/syscall_defs.h File Reference

Syscall entry definitions.

This graph shows which files directly or indirectly include this file:

16.77.1 Detailed Description

Syscall entry definitions.

Definition in file syscall_defs.h.

16.78 syscall_defs.h

```
00001
00005 /*
00006  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #ifndef __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__
00023 #define __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__
00024
00025 #ifndef L4_SYSCALL_MAGIC_OFFSET
00026 #  define L4_SYSCALL_MAGIC_OFFSET 8
00027 #endif
00028 #define L4_SYSCALL_INVOKE   (-0x00000004-L4_SYSCALL_MAGIC_OFFSET)
00029 #define L4_SYSCALL_MEM_OP               (-0x00000008-L4_SYSCALL_MAGIC_OFFSET)
00030
00031 #endif /* __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__ */
```

16.79 contrib/libio-io/l4/io/io.h File Reference

```
#include <l4/sys/types.h>
#include <l4/sys/linkage.h>
```

```
#include <l4/io/types.h>
```

Include dependency graph for io.h:

Functions

- l4_cap_idx_t l4io_request_icu (void)

  Request the ICU object of the client.
- long l4io_request_iomem (l4_addr_t phys, unsigned long size, int flags, l4_addr_t *virt)

  Request an IO memory region.
- long l4io_request_iomem_region (l4_addr_t phys, l4_addr_t virt, unsigned long size, int flags)

  Request an IO memory region and map it to a specified region.
- long l4io_release_iomem (l4_addr_t virt, unsigned long size)

  Release an IO memory region.
- long l4io_request_ioport (unsigned portnum, unsigned len)

  Request an IO port region.
- long l4io_release_ioport (unsigned portnum, unsigned len)

  Release an IO port region.
- l4io_device_handle_t l4io_get_root_device (void)

  Get root device handle of the device bus.
- int l4io_iterate_devices (l4io_device_handle_t *devhandle, l4io_device_t *dev, l4io_resource_handle_↩t *reshandle)

  Iterate over the device bus.
- int l4io_lookup_device (const char *devname, l4io_device_handle_t *dev_handle, l4io_device_t *dev, l4io↩_resource_handle_t *res_handle)

  Find a device by name.
- int l4io_lookup_resource (l4io_device_handle_t devhandle, enum l4io_resource_types_t type, l4io_↩resource_handle_t *reshandle, l4io_resource_t *res)

Request a specific resource from a device description.

- l4_addr_t l4io_request_resource_iomem (l4io_device_handle_t devhandle, l4io_resource_handle_t *reshandle)

  Request IO memory.

- void l4io_request_all_ioports (void(*res_cb)(l4vbus_resource_t const *res))

  Request all available IO-port resources.

- int l4io_has_resource (enum l4io_resource_types_t type, l4vbus_paddr_t start, l4vbus_paddr_t end)

  Check if a resource is available.

16.79.1 Function Documentation

16.79.1.1 l4io_get_root_device()

```
l4io_device_handle_t l4io_get_root_device (
            void ) [inline]
```

Get root device handle of the device bus.

Returns

root device handle

Definition at line 255 of file io.h.

16.79.1.2 l4io_iterate_devices()

```
int l4io_iterate_devices (
            l4io_device_handle_t * devhandle,
            l4io_device_t * dev,
            l4io_resource_handle_t * reshandle )
```

Iterate over the device bus.

Parameters

devhandle	Device handle to start iterating at

Return values

devhandle	Next device handle
dev	Device information, may be NULL
reshandle	Resource handle.

Returns

0 on success, error code otherwise

16.79.1.3 l4io_request_all_ioports()

```
void l4io_request_all_ioports (
            void(*)(l4vbus_resource_t const *res) res_cb )
```

Request all available IO-port resources.

Parameters

res_cb	Callback function called for every port resource found, give NULL for no callback.

16.79.1.4 l4io_request_icu()

```
l4_cap_idx_t l4io_request_icu (
            void  )
```

Request the ICU object of the client.

Returns

Client ICU object, an invalid capability selector is returned if no ICU is available.

16.80 io.h

```
00001
00004 /*
00005  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00006  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012
00013
00014 #pragma once
00015
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/linkage.h>
00018 #include <l4/io/types.h>
00019
00020 EXTERN_C_BEGIN
00021
00038 L4_CV long L4_EXPORT
00039 l4io_request_irq(int irqnum, l4_cap_idx_t irqcap);
00040
00047 L4_CV l4_cap_idx_t L4_EXPORT
00048 l4io_request_icu(void);
00049
00061 L4_CV long L4_EXPORT
00062 l4io_release_irq(int irqnum, l4_cap_idx_t irq_cap);
00063
00088 L4_CV long L4_EXPORT
```

```
l4io_request_iomem(l4_addr_t phys, unsigned long size, int flags,
                   l4_addr_t *virt);

L4_CV long L4_EXPORT
l4io_request_iomem_region(l4_addr_t phys, l4_addr_t virt,
                          unsigned long size, int flags);

L4_CV long L4_EXPORT
l4io_release_iomem(l4_addr_t virt, unsigned long size);

L4_CV long L4_EXPORT
l4io_request_ioport(unsigned portnum, unsigned len);

L4_CV long L4_EXPORT
l4io_release_ioport(unsigned portnum, unsigned len);


/* -------------- Device handling --------------------- */

L4_INLINE
l4io_device_handle_t l4io_get_root_device(void);

L4_CV int L4_EXPORT
l4io_iterate_devices(l4io_device_handle_t *devhandle,
                     l4io_device_t *dev, l4io_resource_handle_t *reshandle);

L4_CV int L4_EXPORT
l4io_lookup_device(const char *devname,
                   l4io_device_handle_t *dev_handle,
                   l4io_device_t *dev, l4io_resource_handle_t *res_handle);

L4_CV int L4_EXPORT
l4io_lookup_resource(l4io_device_handle_t devhandle,
                     enum l4io_resource_types_t type,
                     l4io_resource_handle_t *reshandle,
                     l4io_resource_t *res);


/* --------------- Convenience functions ----------------- */

L4_CV l4_addr_t L4_EXPORT
l4io_request_resource_iomem(l4io_device_handle_t devhandle,
                            l4io_resource_handle_t *reshandle);

L4_CV void L4_EXPORT
l4io_request_all_ioports(void (*res_cb)(l4vbus_resource_t const *res));

L4_CV int L4_EXPORT
l4io_has_resource(enum l4io_resource_types_t type,
                  l4vbus_paddr_t start, l4vbus_paddr_t end);

/* ------------------------------------------------------- */
/* Implementations */

L4_INLINE
l4io_device_handle_t l4io_get_root_device(void)
{ return 0; }

EXTERN_C_END
```

16.81 l4/cxx/alloc.h File Reference

Alloc list.

Data Structures

- class L4::Alloc_list

  A simple list-based allocator.

Namespaces

- L4

  L4 low-level kernel interface.

16.81.1 Detailed Description

Alloc list.

Definition in file alloc.h.

16.82 alloc.h

```
00001
00005 /*
00006  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00007  *              Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4 {
00026
00031   class Alloc_list
00032   {
00033   public:
00034     Alloc_list() : _free(0) {}
00035     Alloc_list(void *blk, unsigned long size) : _free(0)
00036     { free( blk, size); }
00037
00038     void free(void *blk, unsigned long size);
00039     void *alloc(unsigned long size);
00040
00041   private:
00042     struct Elem
00043     {
00044       Elem *next;
00045       unsigned long size;
00046     };
00047
00048     Elem *_free;
00049   };
00050 }
```

16.83 l4/cxx/avl_map File Reference

AVL map.

```
#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/pair>
```

`#include <l4/cxx/avl_set>`

Include dependency graph for avl_map:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct cxx::Bits::Avl_map_get_key< KEY_TYPE >

  Key-getter for Avl_map.
- class cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >

  AVL tree based associative container.

Namespaces

- cxx

  Our C++ library.
- cxx::Bits

  Internal helpers for the cxx package.

16.83.1 Detailed Description

AVL map.

Definition in file avl_map.

16.84 avl_map

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023
00024 #pragma once
00025
00026 #include <l4/cxx/std_alloc>
00027 #include <l4/cxx/std_ops>
00028 #include <l4/cxx/pair>
00029 #include <l4/cxx/avl_set>
00030
00031 namespace cxx {
00032 namespace Bits {
00033
00035 template<typename KEY_TYPE>
00036 struct Avl_map_get_key
00037 {
00038   typedef KEY_TYPE Key_type;
00039   template<typename NODE>
00040   static Key_type const &key_of(NODE const *n)
00041   { return n->item.first; }
00042 };
00043 }
00044
00053 template< typename KEY_TYPE, typename DATA_TYPE,
00054   template<typename A> class COMPARE = Lt_functor,
00055   template<typename B> class ALLOC = New_allocator >
00056 class Avl_map :
00057   public Bits::Base_avl_set<Pair<KEY_TYPE, DATA_TYPE>,
00058                             COMPARE<KEY_TYPE>, ALLOC,
00059                             Bits::Avl_map_get_key<KEY_TYPE> >
00060 {
00061 private:
00062   typedef Pair<KEY_TYPE, DATA_TYPE> Local_item_type;
00063   typedef Bits::Base_avl_set<Local_item_type, COMPARE<KEY_TYPE>, ALLOC,
00064                              Bits::Avl_map_get_key<KEY_TYPE> > Base_type;
00065
00066 public:
00068   typedef COMPARE<KEY_TYPE> Key_compare;
00070   typedef KEY_TYPE Key_type;
00072   typedef DATA_TYPE Data_type;
00074   typedef typename Base_type::Node Node;
00076   typedef typename Base_type::Node_allocator Node_allocator;
00077
```

```
00078   typedef typename Base_type::Iterator Iterator;
00079   typedef typename Base_type::Iterator iterator;
00080   typedef typename Base_type::Const_iterator Const_iterator;
00081   typedef typename Base_type::Const_iterator const_iterator;
00082   typedef typename Base_type::Rev_iterator Rev_iterator;
00083   typedef typename Base_type::Rev_iterator reverse_iterator;
00084   typedef typename Base_type::Const_rev_iterator Const_rev_iterator;
00085   typedef typename Base_type::Const_rev_iterator const_reverse_iterator;
00086
00091   Avl_map(Node_allocator const &alloc = Node_allocator())
00092     : Base_type(alloc)
00093   {}
00094
00110   cxx::Pair<Iterator, int> insert(Key_type const &key, Data_type const &data)
00111   { return Base_type::insert(Pair<Key_type, Data_type>(key, data)); }
00112
00118   Data_type const &operator [] (Key_type const &key) const
00119   { return this->find_node(key)->second; }
00120
00130   Data_type &operator [] (Key_type const &key)
00131   {
00132     Node n = this->find_node(key);
00133     if (n)
00134       return const_cast<Data_type&>(n->second);
00135     else
00136       return insert(key, Data_type()).first->second;
00137   }
00138 };
00139
00140 }
00141
```

16.85 l4/cxx/avl_set File Reference

AVL set.

```
#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/type_traits>
#include <l4/cxx/avl_tree>
```

Include dependency graph for avl_set:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct cxx::Bits::Avl_set_get_key< KEY_TYPE >

  Internal, key-getter for Avl_set nodes.
- class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >

  Internal: AVL set with internally managed nodes.
- class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node

  A smart pointer to a tree item.
- class cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >

  AVL set for simple compareable items.

Namespaces

- cxx

  Our C++ library.
- cxx::Bits

  Internal helpers for the cxx package.

16.85.1 Detailed Description

AVL set.

Definition in file avl_set.

16.86 avl_set

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *               Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/cxx/std_alloc>
00028 #include <l4/cxx/std_ops>
00029 #include <l4/cxx/type_traits>
00030 #include <l4/cxx/avl_tree>
00031
00032 namespace cxx {
00033 namespace Bits {
00042 template< typename Node, typename Key, typename Node_op >
00043 class Avl_set_iter : public __Bst_iter_b<Node, Node_op>
00044 {
00045 private:
00047   typedef __Bst_iter_b<Node, Node_op> Base;
00048
00050   typedef typename Type_traits<Key>::Non_const_type Non_const_key;
00051
00053   typedef Avl_set_iter<Node, Non_const_key, Node_op> Non_const_iter;
00054
00055   using Base::_n;
00056   using Base::_r;
00057   using Base::inc;
00058
00059 public:
00061   Avl_set_iter() = default;
00062
00067   Avl_set_iter(Node const *t) : Base(t) {}
00068
00073   Avl_set_iter(Base const &o) : Base(o) {}
00074
00076   Avl_set_iter(Non_const_iter const &o)
00077   : Base(o) {}
00078
00080   Avl_set_iter &operator = (Non_const_iter const &o)
00081   { Base::operator = (o); return *this; }
00082
00087   Key &operator * () const { return const_cast<Node*>(_n)->item; }
00092   Key *operator -> () const { return &const_cast<Node*>(_n)->item; }
00096   Avl_set_iter &operator ++ () { inc(); return *this; }
00100   Avl_set_iter operator ++ (int)
00101   { Avl_set_iter tmp = *this; inc(); return tmp; }
00102
00103 };
00104
00106 template<typename KEY_TYPE>
00107 struct Avl_set_get_key
00108 {
00109   typedef KEY_TYPE Key_type;
00110   template<typename NODE>
00111   static Key_type const &key_of(NODE const *n)
00112   { return n->item; }
00113 };
00114
00115
00128 template< typename ITEM_TYPE, class COMPARE,
00129           template<typename A> class ALLOC,
00130           typename GET_KEY>
00131 class Base_avl_set
00132 {
00133 public:
00140   enum
00141   {
00142     E_noent =  2,
00143     E_exist = 17,
00144     E_nomem = 12,
```

```
    E_inval = 22
  };
  typedef ITEM_TYPE Item_type;
  typedef GET_KEY Get_key;
  typedef typename GET_KEY::Key_type Key_type;
  typedef typename Type_traits<Item_type>::Const_type Const_item_type;
  typedef COMPARE Item_compare;

private:
  class _Node : public Avl_tree_node
  {
  public:
    Item_type  item;

    _Node() = default;

    _Node(Item_type const &item) : Avl_tree_node(), item(item) {}
  };

public:
  class Node
  {
  private:
    struct No_type;
    friend class Base_avl_set<ITEM_TYPE, COMPARE, ALLOC, GET_KEY>;
    _Node const *_n;
    explicit Node(_Node const *n) : _n(n) {}

  public:
    Node() : _n(0) {}

    Item_type const &operator * () { return _n->item; }
    Item_type const *operator -> () { return &_n->item; }

    bool valid() const { return _n; }

    operator Item_type const * () { if (_n) return &_n->item; else return 0; }
  };

  typedef ALLOC<_Node> Node_allocator;

private:
  typedef Avl_tree<_Node, GET_KEY, COMPARE> Tree;
  Tree _tree;
  Node_allocator _alloc;

  Base_avl_set &operator = (Base_avl_set const &) = delete;

  typedef typename Tree::Fwd_iter_ops Fwd;
  typedef typename Tree::Rev_iter_ops Rev;

public:
  typedef typename Type_traits<Item_type>::Param_type Item_param_type;

  typedef Avl_set_iter<_Node, Item_type, Fwd> Iterator;
  typedef Iterator iterator;
  typedef Avl_set_iter<_Node, Const_item_type, Fwd> Const_iterator;
  typedef Const_iterator const_iterator;
  typedef Avl_set_iter<_Node, Item_type, Rev> Rev_iterator;
  typedef Rev_iterator reverse_iterator;
  typedef Avl_set_iter<_Node, Const_item_type, Rev> Const_rev_iterator;
  typedef Const_rev_iterator const_reverse_iterator;

  explicit Base_avl_set(Node_allocator const &alloc = Node_allocator())
  : _tree(), _alloc(alloc)
  {}

  ~Base_avl_set()
  {
    _tree.remove_all([this](_Node *n)
                     {
                       n->~_Node();
                       _alloc.free(n);
                     });
  }

  inline Base_avl_set(Base_avl_set const &o);

  cxx::Pair<Iterator, int> insert(Item_type const &item);

  int remove(Key_type const &item)
  {
    _Node *n = _tree.remove(item);
    if ((long)n == -E_inval)
      return -E_inval;

    if (n)
```

```
      {
        n->~_Node();
  _alloc.free(n);
  return 0;
      }

    return -E_noent;
  }

  int erase(Key_type const &item)
  { return remove(item); }

  Node find_node(Key_type const &item) const
  { return Node(_tree.find_node(item)); }

  Node lower_bound_node(Key_type const &key) const
  { return Node(_tree.lower_bound_node(key)); }

  Node lower_bound_node(Key_type &&key) const
  { return Node(_tree.lower_bound_node(key)); }

  Const_iterator begin() const { return _tree.begin(); }
  Const_iterator end() const   { return _tree.end(); }

  Iterator begin() { return _tree.begin(); }
  Iterator end()   { return _tree.end(); }

  Const_rev_iterator rbegin() const { return _tree.rbegin(); }
  Const_rev_iterator rend() const   { return _tree.rend(); }

  Rev_iterator rbegin() { return _tree.rbegin(); }
  Rev_iterator rend()   { return _tree.rend(); }

  Const_iterator find(Key_type const &item) const
  { return _tree.find(item); }

#ifdef __DEBUG_L4_AVL
  bool rec_dump(bool print)
  {
    return _tree.rec_dump(print);
  }
#endif
};


//-----------------------------------------------------------------------------
/* Implementation of AVL Tree */

/* Create a copy */
template< typename Item, class Compare, template<typename A> class Alloc, typename KEY_TYPE>
Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::Base_avl_set(Base_avl_set const &o)
  : _tree(), _alloc(o._alloc)
{
  for (Const_iterator i = o.begin(); i != o.end(); ++i)
    insert(*i);
}

/* Insert new _Node. */
template< typename Item, class Compare, template< typename A > class Alloc, typename KEY_TYPE>
Pair<typename Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::Iterator, int>
Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::insert(Item const &item)
{
  _Node *n = _alloc.alloc();
  if (!n)
    return cxx::pair(end(), -E_nomem);

  new (n, Nothrow()) _Node(item);
  Pair<_Node *, bool> err = _tree.insert(n);
  if (!err.second)
    {
      n->~_Node();
      _alloc.free(n);
    }

  return cxx::pair(Iterator(typename Tree::Iterator(err.first, err.first)), err.second ? 0 :
 -E_exist);
}

} // namespace Bits

template< typename ITEM_TYPE, class COMPARE = Lt_functor<ITEM_TYPE>,
          template<typename A> class ALLOC = New_allocator>
class Avl_set :
  public Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
                            Bits::Avl_set_get_key<ITEM_TYPE> >
{
private:
```

```
  typedef Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
                             Bits::Avl_set_get_key<ITEM_TYPE> > Base;

public:
  typedef typename Base::Node_allocator Node_allocator;
  Avl_set() = default;
  Avl_set(Node_allocator const &alloc)
  : Base(alloc)
  {}
};

} // namespace cxx
```

16.87 l4/cxx/avl_tree File Reference

AVL tree.

```
#include "std_ops"
#include "pair"
#include "bits/bst.h"
#include "bits/bst_iter.h"
```

Include dependency graph for avl_tree:

This graph shows which files directly or indirectly include this file:

Data Structures

- class cxx::Avl_tree_node

  Node of an AVL tree.
- class cxx::Avl_tree< Node, Get_key, Compare >

  A generic AVL tree.

Namespaces

- cxx

  Our C++ library.

16.87.1 Detailed Description

AVL tree.

Definition in file avl_tree.

16.88 avl_tree

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *               Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include "std_ops"
00028 #include "pair"
00029
00030 #include "bits/bst.h"
00031 #include "bits/bst_iter.h"
00032
00033 namespace cxx {
00034
00038 class Avl_tree_node : public Bits::Bst_node
00039 {
00040 private:
00041   template< typename Node, typename Get_key, typename Compare >
00042   friend class Avl_tree;
00043
00045   typedef Bits::Direction Bal;
00047   typedef Bits::Direction Dir;
00048
00049   // We are a final BST node, hide interior.
00051   using Bits::Bst_node::next;
00052   using Bits::Bst_node::next_p;
00053   using Bits::Bst_node::rotate;
00057   Bal _balance;
00058
00059 protected:
00061   Avl_tree_node() = default;
00062
00063 private:
00064   Avl_tree_node(Avl_tree_node const &o) = delete;
00065   Avl_tree_node(Avl_tree_node &&o) = delete;
00066
00068   Avl_tree_node &operator = (Avl_tree_node const &o) = default;
00070   Avl_tree_node &operator = (Avl_tree_node &&o) = default;
00071
00073   explicit Avl_tree_node(bool) : Bits::Bst_node(true), _balance(Dir::N) {}
00074
00076   static Bits::Bst_node *rotate2(Bst_node **t, Bal idir, Bal pre);
00077
00079   bool balanced() const { return _balance == Bal::N; }
00080
00082   Bal balance() const { return _balance; }
00083
00085   void balance(Bal b) { _balance = b; }
00086 };
00087
00088
00105 template< typename Node, typename Get_key,
00106           typename Compare = Lt_functor<typename Get_key::Key_type> >
00107 class Avl_tree : public Bits::Bst<Node, Get_key, Compare>
00108 {
00109 private:
00110   typedef Bits::Bst<Node, Get_key, Compare> Bst;
00111
00113   using Bst::_head;
00114
00116   using Bst::k;
00117
00119   typedef typename Avl_tree_node::Bal Bal;
00121   typedef typename Avl_tree_node::Bal Dir;
00122
00123   Avl_tree(Avl_tree const &o) = delete;
00124   Avl_tree &operator = (Avl_tree const &o) = delete;
00125   Avl_tree(Avl_tree &&o) = delete;
00126   Avl_tree &operator = (Avl_tree &&o) = delete;
```

```
00127
00128 public:
00130   typedef typename Bst::Key_type Key_type;
00131   typedef typename Bst::Key_param_type Key_param_type;
00133
00134   // Grab iterator types from Bst
00137   typedef typename Bst::Iterator Iterator;
00139   typedef typename Bst::Const_iterator Const_iterator;
00141   typedef typename Bst::Rev_iterator Rev_iterator;
00143   typedef typename Bst::Const_rev_iterator Const_rev_iterator;
00145
00153   Pair<Node *, bool> insert(Node *new_node);
00154
00161   Node *remove(Key_param_type key);
00165   Node *erase(Key_param_type key) { return remove(key); }
00166
00168   Avl_tree() = default;
00169
00171   ~Avl_tree() noexcept
00172   {
00173     this->remove_all([](Node *){});
00174   }
00175
00176 #ifdef __DEBUG_L4_AVL
00177   bool rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char pfx);
00178   bool rec_dump(bool print)
00179   {
00180     int dp=0;
00181     return rec_dump(static_cast<Avl_tree_node *>(_head), 0, &dp, print, '+');
00182   }
00183 #endif
00184 };
00185
00186
00187 //----------------------------------------------------------------------------
00188 /* IMPLEMENTATION: Bits::__Bst_iter_b */
00189
00190
00191 inline
00192 Bits::Bst_node *
00193 Avl_tree_node::rotate2(Bst_node **t, Bal idir, Bal pre)
00194 {
00195   typedef Bits::Bst_node N;
00196   typedef Avl_tree_node A;
00197   N *tmp[2] = { *t, N::next(*t, idir) };
00198   *t = N::next(tmp[1], !idir);
00199   A *n = static_cast<A*>(*t);
00200
00201   N::next(tmp[0], idir, N::next(n, !idir));
00202   N::next(tmp[1], !idir, N::next(n, idir));
00203   N::next(n, !idir, tmp[0]);
00204   N::next(n, idir, tmp[1]);
00205
00206   n->balance(Bal::N);
00207
00208   if (pre == Bal::N)
00209     {
00210       static_cast<A*>(tmp[0])->balance(Bal::N);
00211       static_cast<A*>(tmp[1])->balance(Bal::N);
00212       return 0;
00213     }
00214
00215   static_cast<A*>(tmp[pre != idir])->balance(!pre);
00216   static_cast<A*>(tmp[pre == idir])->balance(Bal::N);
00217
00218   return N::next(tmp[pre == idir], !pre);
00219 }
00220
00221 //----------------------------------------------------------------------------
00222 /* Implementation of AVL Tree */
00223
00224 /* Insert new _Node. */
00225 template< typename Node, typename Get_key, class Compare>
00226 Pair<Node *, bool>
00227 Avl_tree<Node, Get_key, Compare>::insert(Node *new_node)
00228 {
00229   typedef Avl_tree_node A;
00230   typedef Bits::Bst_node N;
00231   N **t = &_head; /* search variable */
00232   N **s = &_head; /* node where rebalancing may occur */
00233   Key_param_type new_key = Get_key::key_of(new_node);
00234
00235   // search insertion point
00236   for (N *p; (p = *t);)
00237     {
00238       Dir b = this->dir(new_key, p);
00239       if (b == Dir::N)
```

```
  return pair(static_cast<Node*>(p), false);

      if (!static_cast<A const *>(p)->balanced())
  s = t;

      t = A::next_p(p, b);
    }

  *static_cast<A*>(new_node) = A(true);
  *t = new_node;

  N *n = *s;
  A *a = static_cast<A*>(n);
  if (!a->balanced())
    {
      A::Bal b(this->greater(new_key, n));
      if (a->balance() != b)
  {
    // ok we got in balance the shorter subtree go higher
    a->balance(Bal::N);
    // propagate the new balance down to the new node
    n = A::next(n, b);
  }
      else if (b == Bal(this->greater(new_key, A::next(n, b))))
  {
    // left-left or right-right case -> single rotation
    A::rotate(s, b);
    a->balance(Bal::N);
    static_cast<A*>(*s)->balance(Bal::N);
    n = A::next(*s, b);
  }
      else
  {
    // need a double rotation
    n = A::next(A::next(n, b), !b);
    n = A::rotate2(s, b, n == new_node ? Bal::N : Bal(this->greater(new_key, n)));
  }
    }

  for (A::Bal b; n && n != new_node; static_cast<A*>(n)->balance(b), n = A::next(n, b))
    b = Bal(this->greater(new_key, n));

  return pair(new_node, true);
}


/* remove an element */
template< typename Node, typename Get_key, class Compare>
inline
Node *Avl_tree<Node, Get_key, Compare>::remove(Key_param_type key)
{
  typedef Avl_tree_node A;
  typedef Bits::Bst_node N;
  N **q = &_head; /* search variable */
  N **s = &_head; /* last ('deepest') node on the search path to q
                   * with balance 0, at this place the rebalancing
                   * stops in any case */
  N **t = 0;
  Dir dir;

  // find target node and rebalancing entry
  for (N *n; (n = *q); q = A::next_p(n, dir))
    {
      dir = Dir(this->greater(key, n));
      if (dir == Dir::L && !this->greater(k(n), key))
  /* found node */
  t = q;

      if (!A::next(n, dir))
  break;

      A const *a = static_cast<A const *>(n);
      if (a->balanced() || (a->balance() == !dir && A::next<A>(n, !dir)->balanced()))
  s = q;
    }

  // nothing found
  if (!t)
    return 0;

  A *i = static_cast<A*>(*t);

  for (N *n; (n = *s); s = A::next_p(n, dir))
    {
      dir = Dir(this->greater(key, n));

      if (!A::next(n, dir))
```

```
  break;

      A *a = static_cast<A*>(n);
      // got one out of balance
      if (a->balanced())
  a->balance(!dir);
      else if (a->balance() == dir)
  a->balance(Bal::N);
      else
  {
    // we need rotations to get in balance
    Bal b = A::next<A>(n, !dir)->balance();
    if (b == dir)
      A::rotate2(s, !dir, A::next<A>(A::next(n, !dir), dir)->balance());
    else
      {
        A::rotate(s, !dir);
        if (b != Bal::N)
    {
      a->balance(Bal::N);
      static_cast<A*>(*s)->balance(Bal::N);
    }
        else
    {
      a->balance(!dir);
      static_cast<A*>(*s)->balance(dir);
    }
      }
    if (n == i)
      t = A::next_p(*s, dir);
  }
    }

  A *n = static_cast<A*>(*q);
  *t = n;
  *q = A::next(n, !dir);
  *n = *i;

  return static_cast<Node*>(i);
}

#ifdef __DEBUG_L4_AVL
template< typename Node, typename Get_key, class Compare>
bool Avl_tree<Node, Get_key, Compare>::rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char pfx)
{
  typedef Avl_tree_node A;

  if (!n)
    return true;

  int dpx[2] = {depth,depth};
  bool res = true;

  res = rec_dump(A::next<A>(n, Dir::R), depth + 1, dpx + 1, print, '/');

  if (print)
    {
      fprintf(stderr, "%2d: [%8p] b=%1d: ", depth, n, (int)n->balance().d);

      for (int i = 0; i < depth; ++i)
  std::cerr « "    ";

      std::cerr « pfx « (static_cast<Node*>(n)->item) « std::endl;
    }

  res = res & rec_dump(A::next<A>(n, Dir::L), depth + 1, dpx, print, '\\');

  int b = dpx[1] - dpx[0];

  if (b < 0)
    *dp = dpx[0];
  else
    *dp = dpx[1];

  Bal x = n->balance();
  if ((b < -1 || b > 1) ||
      (b == 0 && x != Bal::N) ||
      (b == -1 && x != Bal::L) ||
      (b == 1 && x != Bal::R))
    {
      if (print)
  fprintf(stderr, "%2d: [%8p] b=%1d: balance error %d\n", depth, n, (int)n->balance().d, b);
      return false;
    }
  return res;
}
```

```
#endif

}
```

16.89 l4/cxx/basic_ostream File Reference

Basic IO stream.

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::IOModifier

  Modifier class for the IO stream.

Namespaces

- L4

  L4 low-level kernel interface.

Variables

- IOModifier const L4::hex

  Modifies the stream to print numbers as hexadecimal values.
- IOModifier const L4::dec

  Modifies the stream to print numbers as decimal values.

16.89.1 Detailed Description

Basic IO stream.

Definition in file basic_ostream.

16.90 basic_ostream

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4 {
00026
00033   class IOModifier
00034   {
00035   public:
00036     IOModifier(int x) : mod(x) {}
00037     bool operator == (IOModifier o) { return mod == o.mod; }
00038     bool operator != (IOModifier o) { return mod != o.mod; }
00039     int mod;
00040   };
00041
00046   class IOBackend
00047   {
00048   public:
00049     typedef int Mode;
00050
00051   protected:
00052     friend class BasicOStream;
00053
00054     IOBackend()
00055     : int_mode(10)
00056     {}
00057
00058     virtual ~IOBackend() {}
00059
00060     virtual void write(char const *str, unsigned len) = 0;
00061
00062   private:
00063     void write(IOModifier m);
00064     void write(long long int c, int len);
00065     void write(long long unsigned c, int len);
00066     void write(long long unsigned c, unsigned char base = 10,
00067                unsigned char len = 0, char pad = ' ');
00068
00069     Mode mode() const
00070     { return int_mode; }
00071
00072     void mode(Mode m)
00073     { int_mode = m; }
00074
00075     int int_mode;
00076   };
00077
00082   class BasicOStream
00083   {
00084   public:
00085     BasicOStream(IOBackend *b)
00086     : iob(b)
00087     {}
00088
00089     void write(char const *str, unsigned len)
00090     {
00091       if (iob)
00092         iob->write(str, len);
00093     }
00094
00095     void write(long long int c, int len)
00096     {
00097       if (iob)
00098         iob->write(c, len);
00099     }
00100
00101     void write(long long unsigned c, unsigned char base = 10,
00102                unsigned char len = 0, char pad = ' ')
00103     {
```

```
      if (iob)
        iob->write(c, base, len, pad);
    }

    void write(long long unsigned c, int len)
    {
      if (iob)
        iob->write(c, len);
    }

    void write(IOModifier m)
    {
      if (iob)
        iob->write(m);
    }

    IOBackend::Mode be_mode() const
    {
      if (iob)
        return iob->mode();
      return 0;
    }

    void be_mode(IOBackend::Mode m)
    {
      if (iob)
        iob->mode(m);
    }

  private:
    IOBackend *iob;
  };

  class IONumFmt
  {
  public:
    IONumFmt(unsigned long long n, unsigned char base = 10,
             unsigned char len = 0, char pad = ' ')
      : n(n), base(base), len(len), pad(pad)
    {}

  BasicOStream &print(BasicOStream &o) const;

  private:
    unsigned long long n;
    unsigned char base, len;
    char pad;
  };

  inline IONumFmt n_hex(unsigned long long n) { return IONumFmt(n, 16); }

  extern IOModifier const hex;

  extern IOModifier const dec;

  inline
  BasicOStream &IONumFmt::print(BasicOStream &o) const
  {
    o.write(n, base, len, pad);
    return o;
  }
}


// Implementation

inline
L4::BasicOStream &
operator « (L4::BasicOStream &s, char const * const str)
{
  if (!str)
    {
      s.write("(NULL)", 6);
      return s;
    }

  unsigned l = 0;
  for (; str[l] != 0; l++)
    ;
  s.write(str, l);
  return s;
}

inline
L4::BasicOStream &
operator « (L4::BasicOStream &s, signed short u)
{
```

```
  s.write((long long signed)u, -1);
  return s;
}

inline
L4::BasicOStream &
operator « (L4::BasicOStream &s, signed u)
{
  s.write((long long signed)u, -1);
  return s;
}

inline
L4::BasicOStream &
operator « (L4::BasicOStream &s, signed long u)
{
  s.write((long long signed)u, -1);
  return s;
}

inline
L4::BasicOStream &
operator « (L4::BasicOStream &s, signed long long u)
{
  s.write(u, -1);
  return s;
}

inline
L4::BasicOStream &
operator « (L4::BasicOStream &s, unsigned short u)
{
  s.write((long long unsigned)u, -1);
  return s;
}

inline
L4::BasicOStream &
operator « (L4::BasicOStream &s, unsigned u)
{
  s.write((long long unsigned)u, -1);
  return s;
}

inline
L4::BasicOStream &
operator « (L4::BasicOStream &s, unsigned long u)
{
  s.write((long long unsigned)u, -1);
  return s;
}

inline
L4::BasicOStream &
operator « (L4::BasicOStream &s, unsigned long long u)
{
  s.write(u, -1);
  return s;
}

inline
L4::BasicOStream &
operator « (L4::BasicOStream &s, void const *u)
{
  long unsigned x = (long unsigned)u;
  L4::IOBackend::Mode mode = s.be_mode();
  s.write(L4::hex);
  s.write((long long unsigned)x, -1);
  s.be_mode(mode);
  return s;
}

inline
L4::BasicOStream &
operator « (L4::BasicOStream &s, L4::IOModifier m)
{
  s.write(m);
  return s;
}

inline
L4::BasicOStream &
operator « (L4::BasicOStream &s, char c)
{
  s.write(&c, 1);
  return s;
}
```

```
00288
00289 inline
00290 L4::BasicOStream &
00291 operator « (L4::BasicOStream &o, L4::IONumFmt const &n)
00292 { return n.print(o); }
```

16.91 l4/cxx/basic_vector.h File Reference

Basic vector.

`#include <l4/cxx/std_alloc>`
Include dependency graph for basic_vector.h:

Namespaces

- cxx

  Our C++ library.

16.91.1 Detailed Description

Basic vector.

Definition in file basic_vector.h.

16.92 basic_vector.h

```
00001
00005 /*
00006  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <l4/cxx/std_alloc>
00025
00026 namespace cxx {
00027
00028 template< typename T >
00029 class Basic_vector
00030 {
00031 public:
00032   Basic_vector(T *array, unsigned long capacity)
00033   : _array(array), _capacity(capacity)
00034   {
00035     for (unsigned long i = 0; i < capacity; ++i)
00036       new (&_array[i]) T();
00037   }
00038
00039 private:
00040   T *_array;
00041   unsigned long _capacity;
00042 };
00043
00044 };
```

16.93 l4/cxx/bits/bst.h File Reference

AVL tree.

```
#include "../type_traits"
#include "bst_base.h"
#include "bst_iter.h"
```

Include dependency graph for bst.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- class cxx::Bits::Bst< Node, Get_key, Compare >

  Basic binary search tree (BST).

Namespaces

- cxx

  Our C++ library.
- cxx::Bits

  Internal helpers for the cxx package.

16.93.1 Detailed Description

AVL tree.

Definition in file bst.h.

16.94 bst.h

```
00001 // vi:ft=cpp
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *               Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include "../type_traits"
00027 #include "bst_base.h"
00028 #include "bst_iter.h"
00029
00030 namespace cxx { namespace Bits {
00031
00039 template< typename Node, typename Get_key, typename Compare >
00040 class Bst
00041 {
00042 private:
00043   typedef Direction Dir;
00045   struct Fwd
00046   {
00047     static Node *child(Node const *n, Direction d)
00048     { return Bst_node::next<Node>(n, d); }
00049
00050     static bool cmp(Node const *l, Node const *r)
00051     { return Compare()(Get_key::key_of(l), Get_key::key_of(r)); }
00052   };
00053
00055   struct Rev
00056   {
00057     static Node *child(Node const *n, Direction d)
00058     { return Bst_node::next<Node>(n, !d); }
00059
00060     static bool cmp(Node const *l, Node const *r)
00061     { return Compare()(Get_key::key_of(r), Get_key::key_of(l)); }
00062   };
00063
00064 public:
00066   typedef typename Get_key::Key_type Key_type;
00068   typedef typename Type_traits<Key_type>::Param_type Key_param_type;
00069
00071   typedef Fwd Fwd_iter_ops;
00073   typedef Rev Rev_iter_ops;
00074
00078   typedef __Bst_iter<Node, Node, Fwd> Iterator;
00080   typedef __Bst_iter<Node, Node const, Fwd> Const_iterator;
00082   typedef __Bst_iter<Node, Node, Rev> Rev_iterator;
00084   typedef __Bst_iter<Node, Node const, Rev> Const_rev_iterator;
00086
00087 protected:
00098   Bst_node *_head;
00099
00101   Bst() : _head(0) {}
00102
00104   Node *head() const { return static_cast<Node*>(_head); }
00105
00107   static Key_type k(Bst_node const *n)
00108   { return Get_key::key_of(static_cast<Node const *>(n)); }
00109
00118   static Dir dir(Key_param_type l, Key_param_type r)
00119   {
00120     Compare cmp;
00121     Dir d(cmp(r, l));
00122     if (d == Direction::L && !cmp(l, r))
00123       return Direction::N;
00124     return d;
00125   }
00126
00135   static Dir dir(Key_param_type l, Bst_node const *r)
00136   { return dir(l, k(r)); }
00137
00139   static bool greater(Key_param_type l, Key_param_type r)
```

```
00140   { return Compare()(r, l); }
00141
00143   static bool greater(Key_param_type l, Bst_node const *r)
00144   { return greater(l, k(r)); }
00145
00147   static bool greater(Bst_node const *l, Bst_node const *r)
00148   { return greater(k(l), k(r)); }
00157   template<typename FUNC>
00158   static void remove_tree(Bst_node *head, FUNC &&callback)
00159   {
00160     if (Bst_node *n = Bst_node::next(head, Dir::L))
00161       remove_tree(n, callback);
00162
00163     if (Bst_node *n = Bst_node::next(head, Dir::R))
00164       remove_tree(n, callback);
00165
00166     callback(static_cast<Node *>(head));
00167   }
00168
00169 public:
00170
00179   Const_iterator begin() const { return Const_iterator(head()); }
00184   Const_iterator end() const { return Const_iterator(); }
00185
00190   Iterator begin() { return Iterator(head()); }
00195   Iterator end() { return Iterator(); }
00196
00201   Const_rev_iterator rbegin() const { return Const_rev_iterator(head()); }
00206   Const_rev_iterator rend() const { return Const_rev_iterator(); }
00207
00212   Rev_iterator rbegin() { return Rev_iterator(head()); }
00217   Rev_iterator rend() { return Rev_iterator(); }
00225
00231   Node *find_node(Key_param_type key) const;
00232
00239   Node *lower_bound_node(Key_param_type key) const;
00240
00247   Const_iterator find(Key_param_type key) const;
00248
00257   template<typename FUNC>
00258   void remove_all(FUNC &&callback)
00259   {
00260     if (!_head)
00261       return;
00262
00263     Bst_node *head = _head;
00264     _head = 0;
00265     remove_tree(head, cxx::forward<FUNC>(callback));
00266   }
00267
00268
00270 };
00271
00272 /* find an element */
00273 template< typename Node, typename Get_key, class Compare>
00274 inline
00275 Node *
00276 Bst<Node, Get_key, Compare>::find_node(Key_param_type key) const
00277 {
00278   Dir d;
00279
00280   for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00281     {
00282       d = dir(key, q);
00283       if (d == Dir::N)
00284   return static_cast<Node*>(q);
00285     }
00286   return 0;
00287 }
00288
00289 template< typename Node, typename Get_key, class Compare>
00290 inline
00291 Node *
00292 Bst<Node, Get_key, Compare>::lower_bound_node(Key_param_type key) const
00293 {
00294   Dir d;
00295   Bst_node *r = 0;
00296
00297   for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00298     {
00299       d = dir(key, q);
00300       if (d == Dir::L)
00301   r = q; // found a node greater than key
00302       else if (d == Dir::N)
00303   return static_cast<Node*>(q);
00304     }
00305   return static_cast<Node*>(r);
```

```
}

/* find an element */
template< typename Node, typename Get_key, class Compare>
inline
typename Bst<Node, Get_key, Compare>::Const_iterator
Bst<Node, Get_key, Compare>::find(Key_param_type key) const
{
  Bst_node *q = _head;
  Bst_node *r = q;

  for (Dir d; q; q = Bst_node::next(q, d))
    {
      d = dir(key, q);
      if (d == Dir::N)
  return Iterator(static_cast<Node*>(q), static_cast<Node *>(r));

      if (d != Dir::L && q == r)
  r = Bst_node::next(q, d);
    }
  return Iterator();
}

}}
```

16.95 l4/cxx/bits/bst_base.h File Reference

AVL tree.

This graph shows which files directly or indirectly include this file:

Data Structures

- struct cxx::Bits::Direction

  The direction to go in a binary search tree.
- class cxx::Bits::Bst_node

  Basic type of a node in a binary search tree (BST).

Namespaces

- cxx

Our C++ library.

- cxx::Bits

  Internal helpers for the cxx package.

16.95.1 Detailed Description

AVL tree.

Definition in file bst_base.h.

16.96 bst_base.h

```
00001 // vi:ft=cpp
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *               Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 /*
00028  * This file contains very basic bits for implementing binary serach trees
00029  */
00030 namespace cxx {
00034 namespace Bits {
00035
00039 struct Direction
00040 {
00042   enum Direction_e
00043   {
00044     L = 0,
00045     R = 1,
00046     N = 2
00047   };
00048   unsigned char d;
00049
00051   Direction() = default;
00052
00054   Direction(Direction_e d) : d(d) {}
00055
00057   explicit Direction(bool b) : d(Direction_e(b)) /*d(b ? R : L)*/ {}
00058
00063   Direction operator ! () const { return Direction(!d); }
00064
00067   bool operator == (Direction_e o) const { return d == o; }
00068   bool operator != (Direction_e o) const { return d != o; }
00069   bool operator == (Direction o) const { return d == o.d; }
00070   bool operator != (Direction o) const { return d != o.d; }
00072 };
00073
00077 class Bst_node
00078 {
00079   // all BSTs are friends
00080   template< typename Node, typename Get_key, typename Compare >
00081   friend class Bst;
00082
00083 protected:
00094   static Bst_node *next(Bst_node const *p, Direction d)
00095   { return p->_c[d.d]; }
00096
00098   static void next(Bst_node *p, Direction d, Bst_node *n)
00099   { p->_c[d.d] = n; }
```

```
00100
00102   static Bst_node **next_p(Bst_node *p, Direction d)
00103   { return &p->_c[d.d]; }
00104
00106   template< typename Node > static
00107   Node *next(Bst_node const *p, Direction d)
00108   { return static_cast<Node *>(p->_c[d.d]); }
00109
00111   static void rotate(Bst_node **t, Direction idir);
00114 private:
00115   Bst_node *_c[2];
00116
00117 protected:
00119   Bst_node() {}
00120
00122   explicit Bst_node(bool) { _c[0] = _c[1] = 0; }
00123 };
00124
00125 inline
00126 void
00127 Bst_node::rotate(Bst_node **t, Direction idir)
00128 {
00129   Bst_node *tmp = *t;
00130   *t = next(tmp, idir);
00131   next(tmp, idir, next(*t, !idir));
00132   next(*t, !idir, tmp);
00133 }
00134
00135 }}
```

16.97 l4/cxx/bits/bst_iter.h File Reference

AVL tree.

`#include "bst_base.h"`

Include dependency graph for bst_iter.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- cxx

  Our C++ library.
- cxx::Bits

  Internal helpers for the cxx package.

16.97.1 Detailed Description

AVL tree.

Definition in file bst_iter.h.

16.98 bst_iter.h

```
00001 // vi:ft=cpp
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *               Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include "bst_base.h"
00028
00029 namespace cxx { namespace Bits {
00030
00039 template< typename Node, typename Node_op >
00040 class __Bst_iter_b
00041 {
00042 protected:
00043   typedef Direction Dir;
00044   Node const *_n;
00045   Node const *_r;
00046
00048   __Bst_iter_b() : _n(0), _r(0) {}
00049
00055   __Bst_iter_b(Node const *t)
00056     : _n(t), _r(_n)
00057   { _downmost(); }
00058
00059   __Bst_iter_b(Node const *t, Node const *r)
00060     : _n(t), _r(r)
00061   {}
00062
00064   inline void _downmost();
00065
00067   inline void inc();
00068
00069 public:
00071   bool operator == (__Bst_iter_b const &o) const { return _n == o._n; }
00073   bool operator != (__Bst_iter_b const &o) const { return _n != o._n; }
00074 };
00075
00084 template< typename Node, typename Node_type, typename Node_op >
00085 class __Bst_iter : public __Bst_iter_b<Node, Node_op>
00086 {
00087 private:
00089   typedef __Bst_iter_b<Node, Node_op> Base;
00090
00091   using Base::_n;
00092   using Base::_r;
00093   using Base::inc;
00094
00095 public:
00097   __Bst_iter() {}
00098
00104   __Bst_iter(Node const *t) : Base(t) {}
00105   __Bst_iter(Node const *t, Node const *r) : Base(t, r) {}
00106
00107 //  template<typename Key2>
00108   __Bst_iter(Base const &o) : Base(o) {}
00109
00114   Node_type &operator * () const { return *const_cast<Node *>(_n); }
00119   Node_type *operator -> () const { return const_cast<Node *>(_n); }
00123   __Bst_iter &operator ++ () { inc(); return *this; }
00127   __Bst_iter &operator ++ (int)
00128   { __Bst_iter tmp = *this; inc(); return tmp; }
00129 };
00130
00131
00132 //-----------------------------------------------------------------------------
00133 /* IMPLEMENTATION: __Bst_iter_b */
00134
00135 template< typename Node, typename Node_op>
00136 inline
```

```
void __Bst_iter_b<Node, Node_op>::_downmost()
{
  while (_n)
    {
      Node *n = Node_op::child(_n, Dir::L);
      if (n)
  _n = n;
      else
  return;
    }
}

template< typename Node, typename Node_op>
void __Bst_iter_b<Node, Node_op>::inc()
{
  if (!_n)
    return;

  if (_n == _r)
    {
      _r = _n = Node_op::child(_r, Dir::R);
      _downmost();
      return;
    }

  if (Node_op::child(_n, Dir::R))
    {
      _n = Node_op::child(_n, Dir::R);
      _downmost();
      return;
    }

  Node const *q = _r;
  Node const *p = _r;
  while (1)
    {
      if (Node_op::cmp(_n, q))
  {
    p = q;
    q = Node_op::child(q, Dir::L);
  }
      else if (_n == q || Node_op::child(q, Dir::R) == _n)
  {
    _n = p;
    return;
  }
      else
  q = Node_op::child(q, Dir::R);
    }
}

}}
```

16.99 l4/cxx/bits/smart_ptr_list.h File Reference

Implementation of a list of smart-pointer-managed objects.

```
#include "../type_traits"
```

Include dependency graph for smart_ptr_list.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- class cxx::Bits::Smart_ptr_list_item< T, STORE_T >

  List item for an arbitrary item in a Smart_ptr_list.
- class cxx::Bits::Smart_ptr_list< ITEM >

  List of smart-pointer-managed objects.

Namespaces

- cxx

  Our C++ library.
- cxx::Bits

  Internal helpers for the cxx package.

16.99.1 Detailed Description

Implementation of a list of smart-pointer-managed objects.

Definition in file smart_ptr_list.h.

16.100 smart_ptr_list.h

```
00001
00005 /*
00006  * Copyright (C) 2018 Kernkonzept GmbH.
00007  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00008  *
00009  * This file is distributed under the terms of the GNU General Public
00010  * License, version 2.  Please see the COPYING-GPL-2 file for details.
00011  */
00012 #pragma once
00013
00014 #include "../type_traits"
00015
00016 namespace cxx { namespace Bits {
00017
00027 template <typename T, typename STORE_T>
00028 class Smart_ptr_list_item
00029 {
00030   using Value_type = T;
00031   using Storage_type = STORE_T;
00032
00033   template<typename U> friend class Smart_ptr_list;
00034   Storage_type _n;
00035 };
00036
00046 template <typename ITEM>
00047 class Smart_ptr_list
00048 {
00049   using Value_type = typename ITEM::Value_type;
00050   using Next_type = typename ITEM::Storage_type;
00051
00052 public:
00053   class Iterator
00054   {
00055   public:
00056     Iterator() : _c(nullptr) {}
00057
00058     Value_type *operator * () const { return _c; }
00059     Value_type *operator -> () const { return _c; }
00060
00061     Iterator operator ++ ()
00062     {
00063       _c = _c->_n.get();
00064       return *this;
00065     }
00066
00067     bool operator == (Iterator const &o) const { return _c == o._c; }
00068     bool operator != (Iterator const &o) const { return !operator == (o); }
00069
00070   private:
00071     friend class Smart_ptr_list;
00072
00073     explicit Iterator(Value_type *i) : _c(i) {}
00074
00075     Value_type *_c;
00076   };
00077
00078   class Const_iterator
00079   {
00080   public:
00081     Const_iterator() : _c(nullptr) {}
00082
00083     Value_type const *operator * () const { return _c; }
00084     Value_type const *operator -> () const { return _c; }
00085
00086     Const_iterator operator ++ ()
00087     {
00088       _c = _c->_n.get();
00089       return *this;
00090     }
00091
00092     bool operator == (Const_iterator const &o) const { return _c == o._c; }
00093     bool operator != (Const_iterator const &o) const { return !operator == (o); }
```

```
  private:
    friend class Smart_ptr_list;

    explicit Const_iterator(Value_type const *i) : _c(i) {}

    Value_type const *_c;
  };

  Smart_ptr_list() : _b(&_f) {}

  void push_front(Next_type &&e)
  {
    e->_n = cxx::move(this->_f);
    this->_f = cxx::move(e);

    if (_b == &_f)
      _b = &(_f->_n);
  }

  void push_front(Next_type const &e)
  {
    e->_n = cxx::move(this->_f);
    this->_f = e;

    if (_b == &_f)
      _b = &(_f->_n);
  }

  void push_back(Next_type &&e)
  {
    *_b = cxx::move(e);
    _b = &((*_b)->_n);
  }

  void push_back(Next_type const &e)
  {
    *_b = e;
    _b = &((*_b)->_n);
  }

  Value_type *front() const
  { return _f.get(); }

  Next_type pop_front()
  {
    Next_type ret = cxx::move(_f);

    if (ret)
      _f = cxx::move(ret->_n);

    if (!_f)
      _b = &_f;

    return ret;
  }

  bool empty() const
  { return !_f; }

  Iterator begin() { return Iterator(_f.get()); }
  Iterator end() { return Iterator(); }

  Const_iterator begin() const { return Const_iterator(_f.get()); }
  Const_iterator end() const { return Const_iterator(); }

  Const_iterator cbegin() const { return const_iterator(_f.get()); }
  Const_iterator cend() const { return Const_iterator(); }

private:
  Next_type _f;
  Next_type *_b;
};

} }
```

16.101 l4/cxx/exceptions File Reference

Base exceptions.

```
#include <l4/cxx/l4types.h>
#include <l4/cxx/basic_ostream>
```

```
#include <l4/sys/err.h>
#include <l4/sys/capability>
#include <l4/util/backtrace.h>
```

Include dependency graph for exceptions:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Exception_tracer

  Back-trace support for exceptions.
- class L4::Base_exception

  Base class for all exceptions, thrown by the L4Re framework.
- class L4::Runtime_error

  Exception for an abstract runtime error.
- class L4::Out_of_memory

  Exception signalling insufficient memory.

- class L4::Element_already_exists

  Exception for duplicate element insertions.
- class L4::Unknown_error

  Exception for an unknown condition.
- class L4::Element_not_found

  Exception for a failed lookup (element not found).
- class L4::Invalid_capability

  Indicates that an invalid object was invoked.
- class L4::Com_error

  Error conditions during IPC.
- class L4::Bounds_error

  Access out of bounds.

Namespaces

- L4

  L4 low-level kernel interface.

Macros

- #define L4_CXX_EXCEPTION_BACKTRACE 20

  Number of instruction pointers in backtrace.

16.101.1 Detailed Description

Base exceptions.

Definition in file exceptions.

16.102 exceptions

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/cxx/l4types.h>
00028 #include <l4/cxx/basic_ostream>
00029 #include <l4/sys/err.h>
00030 #include <l4/sys/capability>
00031
00032
00039 #ifndef L4_CXX_NO_EXCEPTION_BACKTRACE
```

```
00040 # define L4_CXX_EXCEPTION_BACKTRACE 20
00041 #endif
00042
00043 #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00044 #include <l4/util/backtrace.h>
00045 #endif
00046
00048 namespace L4
00049 {
00062   class Exception_tracer
00063   {
00064 #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00065   private:
00066     void *_pc_array[L4_CXX_EXCEPTION_BACKTRACE];
00067     int   _frame_cnt;
00068
00069   protected:
00073 #if defined(__PIC__)
00074     Exception_tracer() throw() : _frame_cnt(0) {}
00075 #else
00076     Exception_tracer() throw()
00077     : _frame_cnt(l4util_backtrace(_pc_array, L4_CXX_EXCEPTION_BACKTRACE)) {}
00078 #endif
00079
00080   public:
00084     void const *const *pc_array() const throw() { return _pc_array; }
00088     int frame_count() const throw() { return _frame_cnt; }
00089 #else
00090   protected:
00094     Exception_tracer() throw() {}
00095
00096   public:
00100     void const *const *pc_array() const throw() { return 0; }
00104     int frame_count() const throw() { return 0; }
00105 #endif
00106   };
00107
00116   class Base_exception : public Exception_tracer
00117   {
00118   protected:
00120     Base_exception() throw() {}
00121
00122   public:
00126     virtual char const *str() const throw () = 0;
00127
00129     virtual ~Base_exception() throw () {}
00130   };
00131
00139   class Runtime_error : public Base_exception
00140   {
00141   private:
00142     long _errno;
00143     char _extra[80];
00144
00145   public:
00152     explicit Runtime_error(long err_no, char const *extra = 0) throw ()
00153       : _errno(err_no)
00154     {
00155       if (!extra)
00156         _extra[0] = 0;
00157       else
00158         {
00159           unsigned i = 0;
00160           for (; i < sizeof(_extra) && extra[i]; ++i)
00161             _extra[i] = extra[i];
00162           _extra[i < sizeof(_extra) ? i : sizeof(_extra) - 1] = 0;
00163         }
00164     }
00165     char const *str() const throw ()
00166     { return l4sys_errtostr(_errno); }
00167
00173     char const *extra_str() const { return _extra; }
00174     ~Runtime_error() throw () {}
00175
00181     long err_no() const throw() { return _errno; }
00182   };
00183
00188   class Out_of_memory : public Runtime_error
00189   {
00190   public:
00192     explicit Out_of_memory(char const *extra = "") throw()
00193     : Runtime_error(-L4_ENOMEM, extra) {}
00195     ~Out_of_memory() throw() {}
00196   };
00197
00198
00203   class Element_already_exists : public Runtime_error
```

```
  {
  public:
    explicit Element_already_exists(char const *e = "") throw()
    : Runtime_error(-L4_EEXIST, e) {}
    ~Element_already_exists() throw() {}
  };

  class Unknown_error : public Base_exception
  {
  public:
    Unknown_error() throw() {}
    char const *str() const throw() { return "unknown error"; }
    ~Unknown_error() throw() {}
  };

  class Element_not_found : public Runtime_error
  {
  public:
    explicit Element_not_found(char const *e = "") throw()
    : Runtime_error(-L4_ENOENT, e) {}
  };

  class Invalid_capability : public Base_exception
  {
  private:
    Cap<void> const _o;

  public:
    explicit Invalid_capability(Cap<void> const &o) throw() : _o(o) {}
    template< typename T>
    explicit Invalid_capability(Cap<T> const &o) throw() : _o(o.cap()) {}
    char const *str() const throw() { return "invalid object"; }

    Cap<void> const &cap() const throw() { return _o; }
    ~Invalid_capability() throw() {}
  };

  class Com_error : public Runtime_error
  {
  public:
    explicit Com_error(long err) throw() : Runtime_error(err) {}

    ~Com_error() throw() {}
  };

  class Bounds_error : public Runtime_error
  {
  public:
    explicit Bounds_error(char const *e = "") throw()
    : Runtime_error(-L4_ERANGE, e) {}
    ~Bounds_error() throw() {}
  };
};

inline
L4::BasicOStream &
operator « (L4::BasicOStream &o, L4::Base_exception const &e)
{
  o « "Exception: " « e.str() « ", backtrace ...\n";
  for (int i = 0; i < e.frame_count(); ++i)
    o « L4::n_hex(l4_addr_t(e.pc_array()[i])) « '\n';

  return o;
}

inline
L4::BasicOStream &
operator « (L4::BasicOStream &o, L4::Runtime_error const &e)
{
  o « "Exception: " « e.str() « ": ";
  if (e.extra_str())
    o « e.extra_str() « ": ";
  o « "backtrace ...\n";
  for (int i = 0; i < e.frame_count(); ++i)
    o « L4::n_hex(l4_addr_t(e.pc_array()[i])) « '\n';

  return o;
}
```

16.103 l4/cxx/iostream File Reference

IO Stream.

```
#include <l4/cxx/basic_ostream>
```

Include dependency graph for iostream:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4

  L4 low-level kernel interface.

Variables

- BasicOStream L4::cout

  Standard output stream.
- BasicOStream L4::cerr

  Standard error stream.

16.103.1 Detailed Description

IO Stream.

Definition in file iostream.

16.104 iostream

```
// -*- Mode: C++ -*-
// vim:ft=cpp
/*
 * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/cxx/basic_ostream>

namespace L4 {

  extern BasicOStream cout;

  extern BasicOStream cerr;

  extern void iostream_init();

  static void  __attribute__((used, constructor)) __iostream_init()
  { iostream_init(); }
};
```

16.105 l4/cxx/ipc_helper File Reference

IPC helper.

```
#include <l4/cxx/exceptions>
#include <l4/sys/utcb.h>
```

Include dependency graph for ipc_helper:

Namespaces

- L4

  L4 low-level kernel interface.

Functions

- void L4::throw_ipc_exception (L4::Cap< void > const &o, l4_msgtag_t const &err, l4_utcb_t *utcb)

  Throw an L4 IPC error as exception.
- void L4::throw_ipc_exception (void const *o, l4_msgtag_t const &err, l4_utcb_t *utcb)

  Throw an L4 IPC error as exception.

16.105.1 Detailed Description

IPC helper.

Definition in file ipc_helper.

16.106 ipc_helper

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/cxx/exceptions>
00027 #include <l4/sys/utcb.h>
00028
00029 namespace L4
00030 {
00031 #ifdef __EXCEPTIONS
00040   inline void
00041   throw_ipc_exception(L4::Cap<void> const &o, l4_msgtag_t const &err,
00042                       l4_utcb_t *utcb)
00043   {
00044     (void)o;
00045     if (err.has_error())
00046       throw (L4::Com_error(l4_error_u(err, utcb)));
00047   }
00048
00057   inline void
00058   throw_ipc_exception(void const *o, l4_msgtag_t const &err,
00059                       l4_utcb_t *utcb)
00060   { throw_ipc_exception(L4::Cap<void>(o), err, utcb); }
00061 #endif
00062
00063 }
```

16.107 l4/cxx/ipc_stream File Reference

IPC stream.

```
#include <l4/sys/ipc.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_varg>
#include <l4/cxx/type_traits>
#include <l4/cxx/minmax>
```

Include dependency graph for ipc_stream:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Ipc::Str_cp_in< T >

Abstraction for extracting a zero-terminated string from an lpc::Istream.

- class L4::Ipc::Msg_ptr< T >

  Pointer to an element of type T in an lpc::Istream.
- class L4::Ipc::Istream

  Input stream for IPC unmarshalling.
- class L4::Ipc::Ostream

  Output stream for IPC marshalling.
- class L4::Ipc::Iostream

  Input/Output stream for IPC [un]marshalling.

Namespaces

- L4

  L4 low-level kernel interface.
- L4::Ipc

  IPC related functionality.

Functions

- template<typename T >
  Internal::Buf_cp_out< T > L4::Ipc::buf_cp_out (T const *v, unsigned long size)

  Insert an array into an lpc::Ostream.
- template<typename T >
  Internal::Buf_cp_in< T > L4::Ipc::buf_cp_in (T *v, unsigned long &size)

  Extract an array from an lpc::Istream.
- template<typename T >
  Str_cp_in< T > L4::Ipc::str_cp_in (T *v, unsigned long &size)

  Create a Str_cp_in for the given values.
- template<typename T >
  Msg_ptr< T > L4::Ipc::msg_ptr (T *&p)

  Create an Msg_ptr to adjust the given pointer.
- template<typename T >
  Internal::Buf_in< T > L4::Ipc::buf_in (T *&v, unsigned long &size)

  Return a pointer to stream array data.
- L4::Ipc::Istream & operator>> (L4::Ipc::Istream &s, bool &v)

  Extract one element of type `T` from the stream `s`.
- L4::Ipc::Istream & operator>> (L4::Ipc::Istream &s, l4_msgtag_t &v)

  Extract the L4 message tag from the stream `s`.
- template<typename T >
  L4::Ipc::Istream & operator>> (L4::Ipc::Istream &s, L4::Ipc::Internal::Buf_in< T > const &v)

  Extract an array of `T` elements from the stream `s`.
- template<typename T >
  L4::Ipc::Istream & operator>> (L4::Ipc::Istream &s, L4::Ipc::Msg_ptr< T > const &v)

  Extract an element of type `T` from the stream `s`.
- template<typename T >
  L4::Ipc::Istream & operator>> (L4::Ipc::Istream &s, L4::Ipc::Internal::Buf_cp_in< T > const &v)

  Extract an array of `T` elements from the stream `s`.
- template<typename T >
  L4::Ipc::Istream & operator>> (L4::Ipc::Istream &s, L4::Ipc::Str_cp_in< T > const &v)

  Extract a zero-terminated string from the stream.

- L4::Ipc::Ostream & operator<< (L4::Ipc::Ostream &s, bool v)

  Insert an element to type `T` into the stream `s`.
- L4::Ipc::Ostream & operator<< (L4::Ipc::Ostream &s, l4_msgtag_t const &v)

  Insert the L4 message tag into the stream `s`.
- template<typename T >
  L4::Ipc::Ostream & operator<< (L4::Ipc::Ostream &s, L4::Ipc::Internal::Buf_cp_out< T > const &v)

  Insert an array with elements of type `T` into the stream `s`.
- L4::Ipc::Ostream & operator<< (L4::Ipc::Ostream &s, char const *v)

  Insert a zero terminated character string into the stream `s`.
- template<typename T >
  T L4::Ipc::read (Istream &s)

  Read a value out of a stream.

16.107.1 Detailed Description

IPC stream.

Definition in file ipc_stream.

16.107.2 Function Documentation

16.107.2.1 operator<<() [1/4]

```
L4::Ipc::Ostream& operator<< (
            L4::Ipc::Ostream & s,
            bool v )  [inline]
```

Insert an element to type `T` into the stream `s`.

Parameters

s	The stream to insert the element `v`.
v	The element to insert.

Returns

The stream `s`.

Definition at line 1202 of file ipc_stream.

References L4::Ipc::Ostream::put().

Here is the call graph for this function:

16.107.2.2 operator<<() [2/4]

```
L4::Ipc::Ostream& operator<< (
            L4::Ipc::Ostream & s,
            char const * v )  [inline]
```

Insert a zero terminated character string into the stream `s`.

Parameters

s	The stream to insert the string `v`.
v	The string to insert.

Returns

The stream `s`.

This operator produces basically the same content as the array insertion, however the length of the array is calculated using `strlen(v) + 1` The string is copied into the message including the trailing zero.

Definition at line 1274 of file ipc_stream.

References L4::Ipc::Ostream::put().

Here is the call graph for this function:

16.107.2.3 operator<<() [3/4]

```
template<typename T >
L4::Ipc::Ostream& operator<< (
            L4::Ipc::Ostream & s,
            L4::Ipc::Internal::Buf_cp_out< T > const & v )  [inline]
```

Insert an array with elements of type `T` into the stream `s`.

Parameters

s	The stream to insert the array `v`.
v	The array to insert (see Ipc::Buf_cp_out()).

Returns

The stream `s`.

Definition at line 1253 of file ipc_stream.

References L4::Ipc::Ostream::put().

Here is the call graph for this function:

16.107.2.4 operator<<() [4/4]

```
L4::Ipc::Ostream& operator<< (
            L4::Ipc::Ostream & s,
            l4_msgtag_t const & v )  [inline]
```

Insert the L4 message tag into the stream `s`.

Parameters

s	The stream to insert the tag `v`.
v	The L4 message tag to insert.

Returns

The stream `s`.

Note

Only one message tag can be inserted into a stream. Multiple insertions simply overwrite previous insertions.

Definition at line 1237 of file ipc_stream.

References L4::Ipc::Ostream::tag().

Here is the call graph for this function:

16.107.2.5 operator>>() [1/6]

```
L4::Ipc::Istream& operator>> (
            L4::Ipc::Istream & s,
            bool & v )  [inline]
```

Extract one element of type `T` from the stream `s`.

Parameters

	s	The stream to extract from.
out	*v*	Extracted value.

Returns

The stream `s`.

Definition at line 1049 of file ipc_stream.

References L4::Ipc::Istream::get().

Here is the call graph for this function:

16.107.2.6 operator>>() [2/6]

```
template<typename T >
L4::Ipc::Istream& operator>> (
            L4::Ipc::Istream & s,
            L4::Ipc::Internal::Buf_cp_in< T > const & v )  [inline]
```

Extract an array of `T` elements from the stream `s`.

Parameters

	s	The stream to extract from.
out	*v*	Buffer description to copy the array to (Ipc::Buf_cp_out()).

Returns

The stream `s`.

This operator does a copy out of the data into the given buffer.

See Ipc::Buf_in, Ipc::Buf_cp_in, and Ipc::Buf_cp_out.

Definition at line 1156 of file ipc_stream.

References L4::Ipc::Istream::get(), and cxx::min().

Here is the call graph for this function:

16.107.2.7 operator>>() [3/6]

```
template<typename T >
L4::Ipc::Istream& operator>> (
            L4::Ipc::Istream & s,
            L4::Ipc::Internal::Buf_in< T > const & v )  [inline]
```

Extract an array of `T` elements from the stream `s`.

Parameters

	s	The stream to extract from.
out	*v*	Pointer to the extracted array (ipc_buf_in()).

Returns

The stream `s`.

This operator actually does not copy out the data in the array, but returns a pointer into the message buffer itself. This means that the data is only valid as long as there is no new data inserted into the stream.

Note

If array does not fit into transmitted words size will be set to zero. Client has to implement check against zero.

See Ipc::Buf_in, Ipc::Buf_cp_in, and Ipc::Buf_cp_out.

Definition at line 1108 of file ipc_stream.

References L4::Ipc::Istream::get(), and L4::Ipc::Istream::has_more().

Here is the call graph for this function:

16.107.2.8 operator>>() [4/6]

```
template<typename T >
L4::Ipc::Istream& operator>> (
            L4::Ipc::Istream & s,
            L4::Ipc::Msg_ptr< T > const & v )  [inline]
```

Extract an element of type `T` from the stream `s`.

Parameters

	s	The stream to extract from.
out	*v*	Pointer to the extracted element.

Returns

The stream `s`.

This operator actually does not copy out the data, but returns a pointer into the message buffer itself. This means that the data is only valid as long as there is no new data inserted into the stream.

See Msg_ptr.

Definition at line 1135 of file ipc_stream.

References L4::Ipc::Istream::get().

Here is the call graph for this function:

16.107.2.9 operator>>() [5/6]

```
template<typename T >
L4::Ipc::Istream& operator>> (
            L4::Ipc::Istream & s,
            L4::Ipc::Str_cp_in< T > const & v )  [inline]
```

Extract a zero-terminated string from the stream.

Parameters

	s	The stream to extract from.
out	*v*	Buffer description to copy the array to (Ipc::Str_cp_out()).

Returns

The stream `s`.

This operator does a copy out of the data into the given buffer.

Definition at line 1177 of file ipc_stream.

References L4::Ipc::Istream::get(), and cxx::min().

Here is the call graph for this function:

16.107.2.10 operator>>() [6/6]

```
L4::Ipc::Istream& operator>> (
            L4::Ipc::Istream & s,
            l4_msgtag_t & v ) [inline]
```

Extract the L4 message tag from the stream s.

Parameters

	s	The stream to extract from.
out	*v*	The extracted tag.

Returns

The stream s.

Definition at line 1083 of file ipc_stream.

References L4::Ipc::Istream::tag().

Here is the call graph for this function:

16.108 ipc_stream

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/ipc.h>
00028 #include <l4/sys/capability>
00029 #include <l4/sys/cxx/ipc_types>
00030 #include <l4/sys/cxx/ipc_varg>
00031 #include <l4/cxx/type_traits>
00032 #include <l4/cxx/minmax>
00033
00034 namespace L4 {
00035 namespace Ipc {
00036
00037 class Ostream;
00038 class Istream;
00039
00040 namespace Internal {
00058 template< typename T >
00059 class Buf_cp_out
00060 {
00061 public:
00068   Buf_cp_out(T const *v, unsigned long size) : _v(v), _s(size) {}
00069
00077   unsigned long size() const { return _s; }
00078
00086   T const *buf() const { return _v; }
00087
00088 private:
00089   friend class Ostream;
00090   T const *_v;
00091   unsigned long _s;
00092 };
00093 }
00094
00110 template< typename T >
00111 Internal::Buf_cp_out<T> buf_cp_out(T const *v, unsigned long size)
00112 { return Internal::Buf_cp_out<T>(v, size); }
00113
00114
00115 namespace Internal {
00128 template< typename T >
00129 class Buf_cp_in
00130 {
00131 public:
00140   Buf_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00141
00142   unsigned long &size() const { return *_s; }
00143   T *buf() const { return _v; }
00144
00145 private:
00146   friend class Istream;
00147   T *_v;
00148   unsigned long *_s;
00149 };
00150 }
00151
00169 template< typename T >
00170 Internal::Buf_cp_in<T> buf_cp_in(T *v, unsigned long &size)
00171 { return Internal::Buf_cp_in<T>(v, size); }
00172
00188 template< typename T >
00189 class Str_cp_in
00190 {
00191 public:
00200   Str_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00201
```

```
00202   unsigned long &size() const { return *_s; }
00203   T *buf() const { return _v; }
00204
00205 private:
00206   friend class Istream;
00207   T *_v;
00208   unsigned long *_s;
00209 };
00210
00223 template< typename T >
00224 Str_cp_in<T> str_cp_in(T *v, unsigned long &size)
00225 { return Str_cp_in<T>(v, size); }
00226
00239 template< typename T >
00240 class Msg_ptr
00241 {
00242 private:
00243   T **_p;
00244 public:
00251   explicit Msg_ptr(T *&p) : _p(&p) {}
00252   void set(T *p) const { *_p = p; }
00253 };
00254
00262 template< typename T >
00263 Msg_ptr<T> msg_ptr(T *&p)
00264 { return Msg_ptr<T>(p); }
00265
00266
00267 namespace Internal {
00281 template< typename T >
00282 class Buf_in
00283 {
00284 public:
00291   Buf_in(T *&v, unsigned long &size) : _v(&v), _s(&size) {}
00292
00293   void set_size(unsigned long s) const { *_s = s; }
00294   T *&buf() const { return *_v; }
00295
00296 private:
00297   friend class Istream;
00298   T **_v;
00299   unsigned long *_s;
00300 };
00301 }
00302
00320 template< typename T >
00321 Internal::Buf_in<T> buf_in(T *&v, unsigned long &size)
00322 { return Internal::Buf_in<T>(v, size); }
00323
00324 namespace Utcb_stream_check
00325 {
00326   static bool check_utcb_data_offset(unsigned sz)
00327   { return sz > sizeof(l4_umword_t) * L4_UTCB_GENERIC_DATA_SIZE; }
00328 }
00329
00330
00345 class Istream
00346 {
00347 public:
00359   Istream(l4_utcb_t *utcb)
00360   : _tag(), _utcb(utcb),
00361     _current_msg(reinterpret_cast<char*>(l4_utcb_mr_u(utcb)->mr)),
00362     _pos(0), _current_buf(0)
00363   {}
00364
00369   void reset()
00370   {
00371     _pos = 0;
00372     _current_buf = 0;
00373     _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00374   }
00375
00379   template< typename T >
00380   bool has_more(unsigned long count = 1)
00381   {
00382     auto const max_bytes = L4_UTCB_GENERIC_DATA_SIZE * sizeof(l4_umword_t);
00383     unsigned apos = cxx::Type_traits<T>::align(_pos);
00384     return (count <= max_bytes / sizeof(T))
00385            && (apos + (sizeof(T) * count)
00386                <= _tag.words() * sizeof(l4_umword_t));
00387   }
00388
00393
00404   template< typename T >
00405   unsigned long get(T *buf, unsigned long elems)
00406   {
00407     if (L4_UNLIKELY(!has_more<T>(elems)))
```

```
      return 0;

    unsigned long size = elems * sizeof(T);
    _pos = cxx::Type_traits<T>::align(_pos);

    __builtin_memcpy(buf, _current_msg + _pos, size);
    _pos += size;
    return elems;
  }


  template< typename T >
  void skip(unsigned long elems)
  {
    if (L4_UNLIKELY(!has_more<T>(elems)))
      return;

    unsigned long size = elems * sizeof(T);
    _pos = cxx::Type_traits<T>::align(_pos);
    _pos += size;
  }

  template< typename T >
  unsigned long get(Msg_ptr<T> const &buf, unsigned long elems = 1)
  {
    if (L4_UNLIKELY(!has_more<T>(elems)))
      return 0;

    unsigned long size = elems * sizeof(T);
    _pos = cxx::Type_traits<T>::align(_pos);

    buf.set(reinterpret_cast<T*>(_current_msg + _pos));
    _pos += size;
    return elems;
  }


  template< typename T >
  bool get(T &v)
  {
    if (L4_UNLIKELY(!has_more<T>()))
      {
        v = T();
        return false;
      }

    _pos = cxx::Type_traits<T>::align(_pos);
    v = *(reinterpret_cast<T*>(_current_msg + _pos));
    _pos += sizeof(T);
    return true;
  }


  bool get(Ipc::Varg *va)
  {
    Ipc::Varg::Tag t;
    if (!has_more<Ipc::Varg::Tag>())
      {
        va->tag(0);
  return 0;
      }
    get(t);
    va->tag(t);
    char const *d;
    get(msg_ptr(d), va->length());
    va->data(d);

    return 1;
  }

  l4_msgtag_t tag() const { return _tag; }


  l4_msgtag_t &tag() { return _tag; }


  inline bool put(Buf_item const &);

  inline bool put(Small_buf const &);



  inline l4_msgtag_t wait(l4_umword_t *src)
  { return wait(src, L4_IPC_NEVER); }

  inline l4_msgtag_t wait(l4_umword_t *src, l4_timeout_t timeout);

```

```
00583   inline l4_msgtag_t receive(l4_cap_idx_t src)
00584   { return receive(src, L4_IPC_NEVER); }
00585   inline l4_msgtag_t receive(l4_cap_idx_t src, l4_timeout_t timeout);
00586
00588
00592   inline l4_utcb_t *utcb() const { return _utcb; }
00593
00594 protected:
00595   l4_msgtag_t _tag;
00596   l4_utcb_t *_utcb;
00597   char *_current_msg;
00598   unsigned _pos;
00599   unsigned char _current_buf;
00600 };
00601
00602 class Istream_copy : public Istream
00603 {
00604 private:
00605   l4_msg_regs_t _mrs;
00606
00607 public:
00608   Istream_copy(Istream const &o) : Istream(o), _mrs(*l4_utcb_mr_u(o.utcb()))
00609   {
00610     // do some reverse mr to utcb trickery
00611     _utcb = (l4_utcb_t *)((l4_addr_t)&_mrs - (l4_addr_t)l4_utcb_mr_u((l4_utcb_t *)0));
00612     _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00613   }
00614
00615 };
00616
00632 class Ostream
00633 {
00634 public:
00638   Ostream(l4_utcb_t *utcb)
00639   : _tag(), _utcb(utcb),
00640     _current_msg(reinterpret_cast<char *>(l4_utcb_mr_u(_utcb)->mr)),
00641     _pos(0), _current_item(0)
00642   {}
00643
00647   void reset()
00648   {
00649     _pos = 0;
00650     _current_item = 0;
00651     _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00652   }
00653
00661
00668   template< typename T >
00669   bool put(T *buf, unsigned long size)
00670   {
00671     size *= sizeof(T);
00672     _pos = cxx::Type_traits<T>::align(_pos);
00673     if (Utcb_stream_check::check_utcb_data_offset(_pos + size))
00674       return false;
00675
00676     __builtin_memcpy(_current_msg + _pos, buf, size);
00677     _pos += size;
00678     return true;
00679   }
00680
00686   template< typename T >
00687   bool put(T const &v)
00688   {
00689     _pos = cxx::Type_traits<T>::align(_pos);
00690     if (Utcb_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
00691       return false;
00692
00693     *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00694     _pos += sizeof(T);
00695     return true;
00696   }
00697
00698   int put(Varg const &va)
00699   {
00700     put(va.tag());
00701     put(va.data(), va.length());
00702
00703     return 0;
00704   }
00705
00706   template< typename T >
00707   int put(Varg_t<T> const &va)
00708   { return put(static_cast<Varg const &>(va)); }
00709
00715   l4_msgtag_t tag() const { return _tag; }
00716
00722   l4_msgtag_t &tag() { return _tag; }
```

```
00723
00725
00730   inline bool put_snd_item(Snd_item const &);
00731
00732
00737
00747   inline l4_msgtag_t send(l4_cap_idx_t dst, long proto = 0, unsigned flags = 0);
00748
00750
00754   inline l4_utcb_t *utcb() const { return _utcb; }
00755 #if 0
00759   unsigned long tell() const
00760   {
00761     unsigned w = (_pos + sizeof(l4_umword_t)-1) / sizeof(l4_umword_t);
00762     w -= _current_item * 2;
00763     _tag = l4_msgtag(0, w, _current_item, 0);
00764   }
00765 #endif
00766 public:
00767   l4_msgtag_t prepare_ipc(long proto = 0, unsigned flags = 0)
00768   {
00769     unsigned w = (_pos + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
00770     w -= _current_item * 2;
00771     return l4_msgtag(proto, w, _current_item, flags);
00772   }
00773
00774   // XXX: this is a hack for <l4/sys/cxx/ipc_server> adaption
00775   void set_ipc_params(l4_msgtag_t tag)
00776   {
00777     _pos = (tag.words() + tag.items() * 2) * sizeof(l4_umword_t);
00778     _current_item = tag.items();
00779   }
00780 protected:
00781   l4_msgtag_t _tag;
00782   l4_utcb_t *_utcb;
00783   char *_current_msg;
00784   unsigned _pos;
00785   unsigned char _current_item;
00786 };
00787
00788
00800 class Iostream : public Istream, public Ostream
00801 {
00802 public:
00803
00812   explicit Iostream(l4_utcb_t *utcb)
00813   : Istream(utcb), Ostream(utcb)
00814   {}
00815
00816   // disambiguate those functions
00817   l4_msgtag_t tag() const { return Istream::tag(); }
00818   l4_msgtag_t &tag() { return Istream::tag(); }
00819   l4_utcb_t *utcb() const { return Istream::utcb(); }
00820
00826   void reset()
00827   {
00828     Istream::reset();
00829     Ostream::reset();
00830   }
00831
00832
00840
00841   using Istream::get;
00842   using Istream::put;
00843   using Ostream::put;
00844
00846
00851
00867   inline l4_msgtag_t call(l4_cap_idx_t dst, l4_timeout_t timeout, long proto = 0);
00868   inline l4_msgtag_t call(l4_cap_idx_t dst, long proto = 0);
00869
00885   inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst, long proto = 0)
00886   { return reply_and_wait(src_dst, L4_IPC_SEND_TIMEOUT_0, proto); }
00887
00888   inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00889                                    long proto = 0)
00890   { return send_and_wait(dest, src, L4_IPC_SEND_TIMEOUT_0, proto); }
00891
00908   inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst,
00909                                     l4_timeout_t timeout, long proto = 0);
00910   inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00911                                    l4_timeout_t timeout, long proto = 0);
00912   inline l4_msgtag_t reply(l4_timeout_t timeout, long proto = 0);
00913   inline l4_msgtag_t reply(long proto = 0)
00914   { return reply(L4_IPC_SEND_TIMEOUT_0, proto); }
00915
00917 };
```

```
00918
00919
00920 inline bool
00921 Ostream::put_snd_item(Snd_item const &v)
00922 {
00923   typedef Snd_item T;
00924   _pos = cxx::Type_traits<Snd_item>::align(_pos);
00925   if (Utcb_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
00926     return false;
00927
00928   *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00929   _pos += sizeof(T);
00930   ++_current_item;
00931   return true;
00932 }
00933
00934
00935 inline bool
00936 Istream::put(Buf_item const &item)
00937 {
00938   if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 3)
00939     return false;
00940
00941   l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00942
00943   reinterpret_cast<Buf_item&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00944   _current_buf += 2;
00945   return true;
00946 }
00947
00948
00949 inline bool
00950 Istream::put(Small_buf const &item)
00951 {
00952   if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 2)
00953     return false;
00954
00955   l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00956
00957   reinterpret_cast<Small_buf&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00958   _current_buf += 1;
00959   return true;
00960 }
00961
00962
00963 inline l4_msgtag_t
00964 Ostream::send(l4_cap_idx_t dst, long proto, unsigned flags)
00965 {
00966   l4_msgtag_t tag = prepare_ipc(proto, L4_MSGTAG_FLAGS & flags);
00967   return l4_ipc_send(dst, _utcb, tag, L4_IPC_NEVER);
00968 }
00969
00970 inline l4_msgtag_t
00971 Iostream::call(l4_cap_idx_t dst, l4_timeout_t timeout, long label)
00972 {
00973   l4_msgtag_t tag = prepare_ipc(label);
00974   tag = l4_ipc_call(dst, Ostream::_utcb, tag, timeout);
00975   Istream::tag() = tag;
00976   Istream::_pos = 0;
00977   return tag;
00978 }
00979
00980 inline l4_msgtag_t
00981 Iostream::call(l4_cap_idx_t dst, long label)
00982 { return call(dst, L4_IPC_NEVER, label); }
00983
00984
00985 inline l4_msgtag_t
00986 Iostream::reply_and_wait(l4_umword_t *src_dst, l4_timeout_t timeout, long proto)
00987 {
00988   l4_msgtag_t tag = prepare_ipc(proto);
00989   tag = l4_ipc_reply_and_wait(Ostream::_utcb, tag, src_dst, timeout);
00990   Istream::tag() = tag;
00991   Istream::_pos = 0;
00992   return tag;
00993 }
00994
00995
00996 inline l4_msgtag_t
00997 Iostream::send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00998                         l4_timeout_t timeout, long proto)
00999 {
01000   l4_msgtag_t tag = prepare_ipc(proto);
01001   tag = l4_ipc_send_and_wait(dest, Ostream::_utcb, tag, src, timeout);
01002   Istream::tag() = tag;
01003   Istream::_pos = 0;
01004   return tag;
```

```
}

inline l4_msgtag_t
Iostream::reply(l4_timeout_t timeout, long proto)
{
  l4_msgtag_t tag = prepare_ipc(proto);
  tag = l4_ipc_send(L4_INVALID_CAP | L4_SYSF_REPLY, Ostream::_utcb, tag, timeout);
  Istream::tag() = tag;
  Istream::_pos = 0;
  return tag;
}

inline l4_msgtag_t
Istream::wait(l4_umword_t *src, l4_timeout_t timeout)
{
  l4_msgtag_t res;
  res = l4_ipc_wait(_utcb, src, timeout);
  tag() = res;
  _pos = 0;
  return res;
}


inline l4_msgtag_t
Istream::receive(l4_cap_idx_t src, l4_timeout_t timeout)
{
  l4_msgtag_t res;
  res = l4_ipc_receive(src, _utcb, timeout);
  tag() = res;
  _pos = 0;
  return res;
}

} // namespace Ipc
} // namespace L4

inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, bool &v) { s.get(v); return s; }
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, int &v) { s.get(v); return s; }
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, long int &v) { s.get(v); return s; }
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, long long int &v) { s.get(v); return s; }
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned int &v) { s.get(v); return s; }
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned long int &v) { s.get(v); return s; }
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned long long int &v) { s.get(v); return s; }
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, short int &v) { s.get(v); return s; }
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned short int &v) { s.get(v); return s; }
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, char &v) { s.get(v); return s; }
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned char &v) { s.get(v); return s; }
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, signed char &v) { s.get(v); return s; }
inline L4::Ipc::Istream &operator « (L4::Ipc::Istream &s, L4::Ipc::Buf_item const &v) { s.put(v); return s; }
inline L4::Ipc::Istream &operator « (L4::Ipc::Istream &s, L4::Ipc::Small_buf const &v) { s.put(v); return s; }
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Snd_item &v)
{
  l4_umword_t b, d;
  s » b » d;
  v = L4::Ipc::Snd_item(b, d);
  return s;
}
inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Varg &v)
{ s.get(&v); return s; }


inline
L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, l4_msgtag_t &v)
{
  v = s.tag();
  return s;
}

template< typename T >
inline
L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
                              L4::Ipc::Internal::Buf_in<T> const &v)
{
  unsigned long si;
  if (s.get(si) && s.has_more<T>(si))
    v.set_size(s.get(L4::Ipc::Msg_ptr<T>(v.buf()), si));
  else
    v.set_size(0);
  return s;
}

template< typename T >
```

```
inline
L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
                              L4::Ipc::Msg_ptr<T> const &v)
{
  s.get(v);
  return s;
}

template< typename T >
inline
L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
                              L4::Ipc::Internal::Buf_cp_in<T> const &v)
{
  unsigned long sz;
  s.get(sz);
  v.size() = s.get(v.buf(), cxx::min(v.size(), sz));
  return s;
}

template< typename T >
inline
L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
                              L4::Ipc::Str_cp_in<T> const &v)
{
  unsigned long sz;
  s.get(sz);
  unsigned long rsz = s.get(v.buf(), cxx::min(v.size(), sz));
  if (rsz < v.size() && v.buf()[rsz - 1])
    ++rsz; // add the zero termination behind the received data

  if (rsz != 0)
    v.buf()[rsz - 1] = 0;

  v.size() = rsz;
  return s;
}


inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, bool v) { s.put(v); return s; }
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, int v) { s.put(v); return s; }
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, long int v) { s.put(v); return s; }
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, long long int v) { s.put(v); return s; }
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned int v) { s.put(v); return s; }
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned long int v) { s.put(v); return s; }
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned long long int v) { s.put(v); return s; }
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, short int v) { s.put(v); return s; }
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned short int v) { s.put(v); return s; }
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, char v) { s.put(v); return s; }
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned char v) { s.put(v); return s; }
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, signed char v) { s.put(v); return s; }
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Snd_item const &v) { s.put_snd_item(v); return s; }
template< typename T >
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Cap<T> const &v)
{ s « L4::Ipc::Snd_fpage(v.fpage()); return s; }

inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Varg const &v)
{ s.put(v); return s; }
template< typename T >
inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Varg_t<T> const &v)
{ s.put(v); return s; }

inline
L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, l4_msgtag_t const &v)
{
  s.tag() = v;
  return s;
}

template< typename T >
inline
L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s,
                              L4::Ipc::Internal::Buf_cp_out<T> const &v)
{
  s.put(v.size());
  s.put(v.buf(), v.size());
  return s;
}

inline
L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, char const *v)
{
  unsigned long l = __builtin_strlen(v) + 1;
  s.put(l);
  s.put(v, l);
```

```
  return s;
}

namespace L4 { namespace Ipc {
template< typename T >
inline
T read(Istream &s) { T t; s » t; return t; }

} // namespace Ipc
} // namespace L4
```

16.109 l4/cxx/l4iostream File Reference

L4 IO stream.

```
#include <l4/cxx/basic_ostream>
#include <l4/sys/types.h>
#include <l4/sys/capability>
```

Include dependency graph for l4iostream:

16.109.1 Detailed Description

L4 IO stream.

Definition in file l4iostream.

16.110 l4iostream

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/cxx/basic_ostream>
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/capability>
00029
00030 inline
00031 L4::BasicOStream &operator « (L4::BasicOStream &o, l4_msgtag_t const &tag)
00032 {
00033   L4::IOBackend::Mode m = o.be_mode();
00034   o « "[l=" « L4::dec « tag.label() « "; w=" « tag.words() « "; i="
00035     « tag.items() « "]";
00036   o.be_mode(m);
00037   return o;
00038 }
00039
00040 template<typename T>
00041 inline
00042 L4::BasicOStream &operator « (L4::BasicOStream &o, L4::Cap<T> const &cap)
00043 {
00044   o « "[C:" « L4::n_hex(cap.cap()) « "]";
00045   return o;
00046 }
```

16.111 l4/cxx/l4types.h File Reference

L4 Types.

```
#include <l4/sys/types.h>
#include <l4/cxx/basic_ostream>
```

Include dependency graph for l4types.h:

This graph shows which files directly or indirectly include this file:

16.111.1 Detailed Description

L4 Types.

Definition in file l4types.h.

16.112 l4types.h

```
00001
00005 /*
00006  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
```

```
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/sys/types.h>
#include <l4/cxx/basic_ostream>
```

16.113 l4/cxx/main_thread File Reference

Main thread.

`#include <l4/cxx/thread>`

Include dependency graph for main_thread:

Namespaces

- cxx

  Our C++ library.

16.113.1 Detailed Description

Main thread.

Definition in file main_thread.

16.114 main_thread

```
00001
00005 /*
00006  * (c) 2004-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *              Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023
00024 #ifndef L4_CXX_MAIN_THREAD_H__
00025 #define L4_CXX_MAIN_THREAD_H__
00026
00027 #include <l4/cxx/thread>
00028
00029 namespace cxx {
00030   class MainThread : public Thread
00031   {
00032   public:
00033     MainThread() : Thread(true)
00034     {}
00035   };
00036 };
00037
00038 #endif /* L4_CXX_MAIN_THREAD_H__ */
```

16.115 l4/cxx/pair File Reference

Pair implementation.

This graph shows which files directly or indirectly include this file:

Data Structures

- struct cxx::Pair< First, Second >

  Pair of two values.
- class cxx::Pair_first_compare< Cmp, Typ >

  Comparison functor for Pair.

Namespaces

- cxx

  Our C++ library.

16.115.1 Detailed Description

Pair implementation.

Definition in file pair.

16.116 pair

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace cxx {
00026
00035 template< typename First, typename Second >
00036 struct Pair
00037 {
00039   typedef First First_type;
00041   typedef Second Second_type;
00042
00044   First first;
00046   Second second;
00047
00053   template<typename A1, typename A2>
00054   Pair(A1 &&first, A2 &&second)
00055   : first(first), second(second) {}
00056
00058   Pair() = default;
00059 };
00060
00061 template< typename F, typename S >
00062 Pair<F,S> pair(F const &f, S const &s)
00063 { return cxx::Pair<F,S>(f,s); }
00064
00065
00074 template< typename Cmp, typename Typ >
00075 class Pair_first_compare
00076 {
00077 private:
00078   Cmp const &_cmp;
00079
```

```
public:
  Pair_first_compare(Cmp const &cmp = Cmp()) : _cmp(cmp) {}

  bool operator () (Typ const &l, Typ const &r) const
  { return _cmp(l.first,r.first); }
};

}

template< typename OS, typename A, typename B >
inline
OS &operator « (OS &os, cxx::Pair<A,B> const &p)
{
  os « p.first « ';' « p.second;
  return os;
}

```

16.117 l4/cxx/ref_ptr_list File Reference

Implementation of a list of ref-ptr-managed objects.

```
#include <l4/cxx/ref_ptr>
#include "bits/smart_ptr_list.h"
```

Include dependency graph for ref_ptr_list:

Data Structures

- struct cxx::Ref_obj_list_item< T >

  Item for list linked via cxx::Ref_ptr with default refence counting.

Namespaces

- cxx

  Our C++ library.

Typedefs

- template<typename T >
  using cxx::Ref_ptr_list_item = Bits::Smart_ptr_list_item< T, cxx::Ref_ptr< T > >

  Item for list linked with cxx::Ref_ptr.
- template<typename T >
  using cxx::Ref_ptr_list = Bits::Smart_ptr_list< Ref_ptr_list_item< T > >

  Single-linked list where elements are connected via a cxx::Ref_ptr.

16.117.1 Detailed Description

Implementation of a list of ref-ptr-managed objects.

Definition in file ref_ptr_list.

16.118 ref_ptr_list

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * Copyright (C) 2018 Kernkonzept GmbH.
00008  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00009  *
00010  * This file is distributed under the terms of the GNU General Public
00011  * License, version 2.  Please see the COPYING-GPL-2 file for details.
00012  */
00013 #pragma once
00014
00015 #include <l4/cxx/ref_ptr>
00016
00017 #include "bits/smart_ptr_list.h"
00018
00019 namespace cxx {
00020
00022 template <typename T>
00023 using Ref_ptr_list_item = Bits::Smart_ptr_list_item<T, cxx::Ref_ptr<T> >;
00024
00026 template <typename T>
00027 struct Ref_obj_list_item : public Ref_ptr_list_item<T>, public cxx::Ref_obj {};
00028
00032 template <typename T>
00033 using Ref_ptr_list = Bits::Smart_ptr_list<Ref_ptr_list_item<T> >;
00034
00035 }
```

16.119 l4/cxx/std_exc_io File Reference

Base exceptions std stream operator.

```
#include <l4/cxx/exceptions>
#include <iostream>
```

Include dependency graph for std_exc_io:

16.119.1 Detailed Description

Base exceptions std stream operator.

Definition in file std_exc_io.

16.120 std_exc_io

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *          Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/cxx/exceptions>
00028 #include <iostream>
00029
00030 inline
00031 std::ostream &
00032 operator « (std::ostream &o, L4::Base_exception const &e)
```

```
{
  o « "Exception: " « e.str() « ", backtrace ...\n";
  for (int i = 0; i < e.frame_count(); ++i)
    o « (void *)(e.pc_array()[i]) « '\n';

  return o;
}

inline
std::ostream &
operator « (std::ostream &o, L4::Runtime_error const &e)
{
  o « "Exception: " « e.str() « ": ";
  if (e.extra_str())
    o « e.extra_str() « ": ";
  o « "backtrace ...\n";
  for (int i = 0; i < e.frame_count(); ++i)
    o « (void *)(e.pc_array()[i]) « '\n';

  return o;
}
```

16.121 l4/cxx/string.h File Reference

String.

`#include <l4/cxx/basic_ostream>`

Include dependency graph for string.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::String

  A null-terminated string container class.

Namespaces

- L4

  L4 low-level kernel interface.

16.121.1 Detailed Description

String.

Definition in file string.h.

16.122 string.h

```
00001
00005 /*
00006  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00007  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/cxx/basic_ostream>
00026
00027 namespace L4 {
00028
00033   class String
00034   {
00035   public:
00036     String(char const *str = "") : _str(str)
00037     {}
00038
00039     unsigned length() const
00040     {
00041       unsigned l;
00042       for (l = 0; _str[l]; l++)
00043         ;
00044       return l;
00045     }
00046
00047     char const *p_str() const { return _str; }
00048
00049   private:
00050     char const *_str;
00051   };
00052 }
00053
00054 inline
00055 L4::BasicOStream &operator « (L4::BasicOStream &o, L4::String const &s)
00056 {
00057   o « s.p_str();
00058   return o;
00059 }
```

16.123 l4/cxx/unique_ptr_list File Reference

Implementation of a list of unique-ptr-managed objects.

```
#include <l4/cxx/unique_ptr>
#include "bits/smart_ptr_list.h"
```

Include dependency graph for unique_ptr_list:

Namespaces

- cxx

  Our C++ library.

Typedefs

- template<typename T >
  using cxx::Unique_ptr_list_item = Bits::Smart_ptr_list_item< T, cxx::unique_ptr< T > >

  Item for list linked with cxx::unique_ptr.
- template<typename T >
  using cxx::Unique_ptr_list = Bits::Smart_ptr_list< Unique_ptr_list_item< T > >

  Single-linked list where elements are connected with a cxx::unique_ptr.

16.123.1 Detailed Description

Implementation of a list of unique-ptr-managed objects.

Definition in file unique_ptr_list.

16.124 unique_ptr_list

```
// vi:set ft=cpp: -*- Mode: C++ -*-
/*
 * Copyright (C) 2018 Kernkonzept GmbH.
 * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
 *
 * This file is distributed under the terms of the GNU General Public
 * License, version 2.  Please see the COPYING-GPL-2 file for details.
 */
#pragma once

#include <l4/cxx/unique_ptr>

#include "bits/smart_ptr_list.h"

namespace cxx {

template <typename T>
using Unique_ptr_list_item = Bits::Smart_ptr_list_item<T, cxx::unique_ptr<T> >;

template <typename T>
using Unique_ptr_list = Bits::Smart_ptr_list<Unique_ptr_list_item<T> >;

}
```

16.125 l4/irq/irq.h File Reference

IRQ handling routines.

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
```

Include dependency graph for irq.h:

Functions

- l4irq_t * l4irq_attach (int irqnum)

  Attach/connect to IRQ.
- l4irq_t * l4irq_attach_ft (int irqnum, unsigned mode)

  Attach/connect to IRQ using given type.

- l4irq_t * l4irq_attach_thread (int irqnum, l4_cap_idx_t to_thread)

  Attach/connect to IRQ.
- l4irq_t * l4irq_attach_thread_ft (int irqnum, l4_cap_idx_t to_thread, unsigned mode)

  Attach/connect to IRQ using given type.
- long l4irq_wait (l4irq_t *irq)

  Wait for specified IRQ.
- long l4irq_unmask_and_wait_any (l4irq_t *unmask_irq, l4irq_t **ret_irq)

  Unmask a specific IRQ and wait for any attached IRQ.
- long l4irq_wait_any (l4irq_t **irq)

  Wait for any attached IRQ.
- long l4irq_unmask (l4irq_t *irq)

  Unmask a specific IRQ.
- long l4irq_detach (l4irq_t *irq)

  Detach from IRQ.
- l4irq_t * l4irq_request (int irqnum, void(*isr_handler)(void *), void *isr_data, int irq_thread_prio, unsigned mode)

  Attach asychronous ISR handler to IRQ.
- long l4irq_release (l4irq_t *irq)

  Release asynchronous ISR handler and free resources.
- l4irq_t * l4irq_attach_cap (l4_cap_idx_t irqcap)

  Attach/connect to IRQ.
- l4irq_t * l4irq_attach_cap_ft (l4_cap_idx_t irqcap, unsigned mode)

  Attach/connect to IRQ using given type.
- l4irq_t * l4irq_attach_thread_cap (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread)

  Attach/connect to IRQ.
- l4irq_t * l4irq_attach_thread_cap_ft (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread, unsigned mode)

  Attach/connect to IRQ using given type.
- l4irq_t * l4irq_request_cap (l4_cap_idx_t irqcap, void(*isr_handler)(void *), void *isr_data, int irq_thread_↩prio, unsigned mode)

  Attach asychronous ISR handler to IRQ.

16.125.1 Detailed Description

IRQ handling routines.

Definition in file irq.h.

16.126 irq.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *              Henning Schild <hschild@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/compiler.h>
00017 #include <l4/sys/types.h>
00018
00019 __BEGIN_DECLS
00020
```

```
struct l4irq_t;
typedef struct l4irq_t l4irq_t;

L4_CV l4irq_t *
l4irq_attach(int irqnum);

L4_CV l4irq_t *
l4irq_attach_ft(int irqnum, unsigned mode);

L4_CV l4irq_t *
l4irq_attach_thread(int irqnum, l4_cap_idx_t to_thread);

L4_CV l4irq_t *
l4irq_attach_thread_ft(int irqnum, l4_cap_idx_t to_thread,
                       unsigned mode);

L4_CV long
l4irq_wait(l4irq_t *irq);

L4_CV long
l4irq_unmask_and_wait_any(l4irq_t *unmask_irq, l4irq_t **ret_irq);

L4_CV long
l4irq_wait_any(l4irq_t **irq);

L4_CV long
l4irq_unmask(l4irq_t *irq);

L4_CV long
l4irq_detach(l4irq_t *irq);



/**********************************************************************/
L4_CV l4irq_t *
l4irq_request(int irqnum, void (*isr_handler)(void *), void *isr_data,
              int irq_thread_prio, unsigned mode);

L4_CV long
l4irq_release(l4irq_t *irq);



/**********************************************************************/
/**********************************************************************/

L4_CV l4irq_t *
l4irq_attach_cap(l4_cap_idx_t irqcap);

L4_CV l4irq_t *
l4irq_attach_cap_ft(l4_cap_idx_t irqcap, unsigned mode);

L4_CV l4irq_t *
l4irq_attach_thread_cap(l4_cap_idx_t irqcap, l4_cap_idx_t to_thread);

L4_CV l4irq_t *
l4irq_attach_thread_cap_ft(l4_cap_idx_t irqcap, l4_cap_idx_t to_thread,
                           unsigned mode);

/**********************************************************************/
L4_CV l4irq_t *
l4irq_request_cap(l4_cap_idx_t irqcap,
                  void (*isr_handler)(void *), void *isr_data,
                  int irq_thread_prio, unsigned mode);

__END_DECLS
```

16.127 arm/l4/util/irq.h File Reference

ARM specific implementation of irq functions.

`#include <l4/sys/compiler.h>`
Include dependency graph for irq.h:

16.127.1 Detailed Description

ARM specific implementation of irq functions.

Do not use.

Definition in file irq.h.

16.128 irq.h

```
00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016 #ifndef __L4UTIL__ARCH_ARCH__IRQ_H__
00017 #define __L4UTIL__ARCH_ARCH__IRQ_H__
00018
00019 #ifdef __GNUC__
00020
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 L4_INLINE void l4util_cli (void);
00026 L4_INLINE void l4util_sti (void);
00027 L4_INLINE void l4util_flags_save(l4_umword_t *flags);
00028 L4_INLINE void l4util_flags_restore(l4_umword_t *flags);
00029
00030 L4_INLINE
00031 void
00032 l4util_cli(void)
00033 {
00034   extern void __do_not_use_l4util_cli(void);
00035   __do_not_use_l4util_cli();
00036 }
00037
00038
```

```
L4_INLINE
void
l4util_sti(void)
{
  extern void __do_not_use_l4util_sti(void);
  __do_not_use_l4util_sti();
}


L4_INLINE
void
l4util_flags_save(l4_umword_t *flags)
{
  (void)flags;
  extern void __do_not_use_l4util_flags_save(void);
  __do_not_use_l4util_flags_save();
}

L4_INLINE
void
l4util_flags_restore(l4_umword_t *flags)
{
  (void)flags;
  extern void __do_not_use_l4util_flags_restore(void);
  __do_not_use_l4util_flags_restore();
}

EXTERN_C_END

#endif //__GNUC__

#endif /* ! __L4UTIL__ARCH_ARCH__IRQ_H__ */
```

16.129 amd64/l4/util/irq.h File Reference

some PIC and hardware interrupt related functions

```
#include <l4/sys/compiler.h>
#include <l4/util/port_io.h>
```

Include dependency graph for irq.h:

Functions

- void l4util_irq_acknowledge (unsigned int irq)

  Acknowledge IRQ at PIC in fully special nested mode.

16.129.1 Detailed Description

some PIC and hardware interrupt related functions

Date

2003

Author

Jork Loeser jork.loeser@inf.tu-dresden.de Frank Mehnert fm3@os.inf.tu-dresden.↩de

Definition in file irq.h.

16.129.2 Function Documentation

16.129.2.1 l4util_irq_acknowledge()

```
void l4util_irq_acknowledge (
            unsigned int irq ) [inline]
```

Acknowledge IRQ at PIC in fully special nested mode.

Parameters

irq	number of interrupt to acknowledge

Definition at line 66 of file irq.h.

References l4util_in8(), and l4util_out8().

Here is the call graph for this function:

16.130 irq.h

```
00001
00010 /*
00011  * (c) 2003-2009 Author(s)
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU Lesser General Public License 2.1.
00015  * Please see the COPYING-LGPL-2.1 file for details.
00016  */
00017
00018 #ifndef __L4_IRQ_H__
00019 #define __L4_IRQ_H__
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/util/port_io.h>
00023
00024 EXTERN_C_BEGIN
00025
00029 L4_INLINE void
00030 l4util_irq_acknowledge(unsigned int irq);
00031
00034 static inline void
00035 l4util_cli (void)
00036 {
00037   __asm__ __volatile__ ("cli" : : : "memory");
00038 }
00039
00042 static inline void
00043 l4util_sti (void)
00044 {
00045   __asm__ __volatile__ ("sti" : : : "memory");
00046 }
00047
00051 static inline void
00052 l4util_flags_save (l4_umword_t *flags)
00053 {
00054   __asm__ __volatile__ ("pushf ; popq %0 " :"=g" (*flags) : :"memory");
00055 }
00056
00059 static inline void
00060 l4util_flags_restore (l4_umword_t *flags)
00061 {
00062   __asm__ __volatile__ ("pushq %0 ; popf" : :"g" (*flags) : "memory");
00063 }
00064
00065 L4_INLINE void
00066 l4util_irq_acknowledge(unsigned int irq)
00067 {
00068   if (irq > 7)
00069     {
00070       l4util_out8(0x60+(irq & 7), 0xA0);
00071       l4util_out8(0x0B,0xA0);
00072       if (l4util_in8(0xA0) == 0)
00073   l4util_out8(0x60 + 2, 0x20);
00074     }
00075   else
00076     l4util_out8(0x60+irq, 0x20);     /* acknowledge the irq */
00077 };
00078
00079 EXTERN_C_END
00080
00081 #endif
```

16.131 x86/l4/util/irq.h File Reference

some PIC and hardware interrupt related functions

```
#include <l4/sys/compiler.h>
#include <l4/util/port_io.h>
```

Include dependency graph for irq.h:

Functions

- void l4util_irq_acknowledge (unsigned int irq)

  Acknowledge IRQ at PIC in fully special nested mode.

16.131.1 Detailed Description

some PIC and hardware interrupt related functions

Date

2003

Author

Jork Loeser jork.loeser@inf.tu-dresden.de Frank Mehnert fm3@os.inf.tu-dresden.↩de

Definition in file irq.h.

16.131.2 Function Documentation

16.131.2.1 l4util_irq_acknowledge()

```
void l4util_irq_acknowledge (
            unsigned int irq )  [inline]
```

Acknowledge IRQ at PIC in fully special nested mode.

Parameters

irq	number of interrupt to acknowledge

Definition at line 66 of file irq.h.

References l4util_in8(), and l4util_out8().

Here is the call graph for this function:

16.132 irq.h

```
00001
00010 /*
00011  * (c) 2003-2009 Author(s)
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU Lesser General Public License 2.1.
00015  * Please see the COPYING-LGPL-2.1 file for details.
00016  */
00017
00018 #ifndef __L4_IRQ_H__
00019 #define __L4_IRQ_H__
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/util/port_io.h>
00023
00024 EXTERN_C_BEGIN
00025
00029 L4_INLINE void
00030 l4util_irq_acknowledge(unsigned int irq);
00031
00034 static inline void
00035 l4util_cli (void)
00036 {
00037   __asm__ __volatile__ ("cli" : : : "memory");
00038 }
00039
00042 static inline void
00043 l4util_sti (void)
00044 {
00045   __asm__ __volatile__ ("sti" : : : "memory");
00046 }
00047
00051 static inline void
00052 l4util_flags_save (l4_umword_t *flags)
00053 {
00054   __asm__ __volatile__ ("pushfl ; popl %0 " :"=g" (*flags) : :"memory");
00055 }
00056
00059 static inline void
00060 l4util_flags_restore (l4_umword_t *flags)
00061 {
00062   __asm__ __volatile__ ("pushl %0 ; popfl" : :"g" (*flags) : "memory");
00063 }
00064
00065 L4_INLINE void
```

```
l4util_irq_acknowledge(unsigned int irq)
{
  if (irq > 7)
    {
      l4util_out8(0x60+(irq & 7), 0xA0);
      l4util_out8(0x0B,0xA0);
      if (l4util_in8(0xA0) == 0)
  l4util_out8(0x60 + 2, 0x20);
    }
  else
    l4util_out8(0x60+irq, 0x20);     /* acknowledge the irq */
};

EXTERN_C_END

#endif
```

16.133 l4/sys/irq.h File Reference

C Irq interface.

```
#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>
#include <l4/sys/rcv_endpoint.h>
```

Include dependency graph for irq.h:

This graph shows which files directly or indirectly include this file:

Functions

- l4_msgtag_t l4_irq_mux_chain (l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW

  Chain an IRQ to another master IRQ source.
- l4_msgtag_t l4_irq_mux_chain_u (l4_cap_idx_t irq, l4_cap_idx_t slave, l4_utcb_t *utcb) L4_NOTHROW

  Attach an IRQ to this multiplexer.
- l4_msgtag_t l4_irq_detach (l4_cap_idx_t irq) L4_NOTHROW

  Detach from an interrupt source.
- l4_msgtag_t l4_irq_detach_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW

  Detach from this interrupt.
- l4_msgtag_t l4_irq_trigger (l4_cap_idx_t irq) L4_NOTHROW

  Trigger an IRQ.
- l4_msgtag_t l4_irq_trigger_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW

  Trigger the object.
- l4_msgtag_t l4_irq_receive (l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW

  Unmask and wait for specified IRQ.
- l4_msgtag_t l4_irq_receive_u (l4_cap_idx_t irq, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW

  Unmask and wait for this IRQ.
- l4_msgtag_t l4_irq_wait (l4_cap_idx_t irq, l4_umword_t *label, l4_timeout_t to) L4_NOTHROW

  Unmask IRQ and wait for any message.
- l4_msgtag_t l4_irq_wait_u (l4_cap_idx_t irq, l4_umword_t *label, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW

  Unmask IRQ and (open) wait for any message.
- l4_msgtag_t l4_irq_unmask (l4_cap_idx_t irq) L4_NOTHROW

  Unmask IRQ.
- l4_msgtag_t l4_irq_unmask_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW

  Unmask IRQ.

16.133.1 Detailed Description

C Irq interface.

Definition in file irq.h.

16.134 irq.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/kernel_object.h>
00028 #include <l4/sys/ipc.h>
00029 #include <l4/sys/rcv_endpoint.h>
00030
00078 L4_INLINE l4_msgtag_t
00079 l4_irq_mux_chain(l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW;
00080
00087 L4_INLINE l4_msgtag_t
00088 l4_irq_mux_chain_u(l4_cap_idx_t irq, l4_cap_idx_t slave,
00089                    l4_utcb_t *utcb) L4_NOTHROW;
00090
00099 L4_INLINE l4_msgtag_t
00100 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW;
00101
00108 L4_INLINE l4_msgtag_t
00109 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00110
00111
00127 L4_INLINE l4_msgtag_t
00128 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW;
00129
00136 L4_INLINE l4_msgtag_t
00137 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00138
00148 L4_INLINE l4_msgtag_t
00149 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW;
00150
00157 L4_INLINE l4_msgtag_t
00158 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW;
00159
00170 L4_INLINE l4_msgtag_t
00171 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00172            l4_timeout_t to) L4_NOTHROW;
00173
00180 L4_INLINE l4_msgtag_t
00181 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
00182              l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW;
00183
00194 L4_INLINE l4_msgtag_t
00195 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW;
00196
00203 L4_INLINE l4_msgtag_t
00204 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00205
00209 enum L4_irq_sender_op
00210 {
00211   L4_IRQ_SENDER_OP_RESERVED1 = 0, // Ex ATTACH
00212   L4_IRQ_SENDER_OP_DETACH    = 1
00213 };
00214
00218 enum L4_irq_mux_op
00219 {
00220   L4_IRQ_MUX_OP_CHAIN = 0
00221 };
00222
00226 enum L4_irq_op
00227 {
00228   L4_IRQ_OP_TRIGGER   = 2,
00229   L4_IRQ_OP_EOI       = 4
00230 };
00231
00232 /*****************************************************************************
00233  * Implementations
```

```
 */

L4_INLINE l4_msgtag_t
l4_irq_mux_chain_u(l4_cap_idx_t irq, l4_cap_idx_t slave,
                   l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
  m->mr[0] = L4_IRQ_MUX_OP_CHAIN;
  m->mr[1] = l4_map_obj_control(0, 0);
  m->mr[2] = l4_obj_fpage(slave, 0, L4_CAP_FPAGE_RWS).raw;
  return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_MUX, 1, 1, 0),
                     L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_SENDER_OP_DETACH;
  return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_SENDER, 1, 0, 0),
                     L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
{
  return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 0, 0, 0),
                     L4_IPC_BOTH_TIMEOUT_0);
}

L4_INLINE l4_msgtag_t
l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
  return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), to);
}

L4_INLINE l4_msgtag_t
l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
            l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
  return l4_ipc_send_and_wait(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0),
                              label, to);
}

L4_INLINE l4_msgtag_t
l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
  return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), L4_IPC_NEVER);
}


L4_INLINE l4_msgtag_t
l4_irq_mux_chain(l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW
{
  return l4_irq_mux_chain_u(irq, slave, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW
{
  return l4_irq_detach_u(irq, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW
{
  return l4_irq_trigger_u(irq, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW
{
  return l4_irq_receive_u(irq, to, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
            l4_timeout_t to) L4_NOTHROW
{
  return l4_irq_wait_u(irq, label, to, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW
{
```

```
00321   return l4_irq_unmask_u(irq, l4_utcb());
00322 }
00323
```

16.135 l4/libedid/edid.h File Reference

`#include <l4/sys/compiler.h>`
Include dependency graph for edid.h:

Enumerations

- enum Libedid_consts { Libedid_block_size = 128 }

  EDID constants.

Functions

- int libedid_check_header (const unsigned char ∗edid)

  Check for valid EDID header.
- int libedid_checksum (const unsigned char ∗edid)

  Calculates the EDID checksum.
- unsigned libedid_version (const unsigned char ∗edid)

  Returns the EDID version number.
- unsigned libedid_revision (const unsigned char ∗edid)

  Returns the EDID revision number.
- void libedid_pnp_id (const unsigned char ∗edid, unsigned char ∗id)

  Extracts the display's PnP ID.
- void libedid_prefered_resolution (const unsigned char ∗edid, unsigned ∗w, unsigned ∗h)

  Extract the display's prefered mode.
- unsigned libedid_num_ext_blocks (const unsigned char ∗edid)

  Get the number of EDID extension blocks.
- unsigned libedid_dump_standard_timings (const unsigned char ∗edid)

  Dump the standard timings to stdout.
- void libedid_dump (const unsigned char ∗edid)

  Dump raw EDID data to stdout.

16.136 edid.h

```

/*
 * (c) 2014 Matthias Lange <matthias.lange@kernkonzept.com>
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */
#pragma once

#include <l4/sys/compiler.h>

enum Libedid_consts
{
  Libedid_block_size = 128,
};

__BEGIN_DECLS

int libedid_check_header(const unsigned char *edid);

int libedid_checksum(const unsigned char *edid);

unsigned libedid_version(const unsigned char *edid);

unsigned libedid_revision(const unsigned char *edid);

void libedid_pnp_id(const unsigned char *edid, unsigned char *id);

void libedid_prefered_resolution(const unsigned char *edid,
                                 unsigned *w, unsigned *h);

unsigned libedid_num_ext_blocks(const unsigned char *edid);

unsigned libedid_dump_standard_timings(const unsigned char *edid);

void libedid_dump(const unsigned char *edid);

__END_DECLS
```

16.137 l4/libgfxbitmap/bitmap.h File Reference

Bitmap renderer header file.

```
#include <l4/sys/types.h>
#include <l4/re/c/video/view.h>
```

Include dependency graph for bitmap.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct gfxbitmap_offset

  offsets in pmap[] and bmap[]

Param macros for bmap_*

Bitmap type - start least or start most significant bit

- #define pSLIM_BMAP_START_MSB 0x02

'pbm'-style: "The bits are stored eight per byte, high bit first low bit last."

- #define pSLIM_BMAP_START_LSB 0x01
- typedef unsigned int gfxbitmap_color_t

  Standard color type.
- typedef unsigned int gfxbitmap_color_pix_t

  Specific color type.
- gfxbitmap_color_pix_t gfxbitmap_convert_color (l4re_video_view_info_t *vi, gfxbitmap_color_t rgb)

  Convert a color.
- void gfxbitmap_fill (l4_uint8_t *vfb, l4re_video_view_info_t *vi, int x, int y, int w, int h, gfxbitmap_color_pix_t color)

  Fill a rectangular area with a color.
- void gfxbitmap_bmap (l4_uint8_t *vfb, l4re_video_view_info_t *vi, l4_int16_t x, l4_int16_t y, l4_uint32_t w, l4_uint32_t h, l4_uint8_t *bmap, gfxbitmap_color_pix_t fgc, gfxbitmap_color_pix_t bgc, struct gfxbitmap_offset *offset, l4_uint8_t mode)

  Fill a rectangular area with a bicolor bitmap pattern.
- void gfxbitmap_set (l4_uint8_t *vfb, l4re_video_view_info_t *vi, l4_int16_t x, l4_int16_t y, l4_uint32_t w, l4_uint32_t h, l4_uint32_t xoffs, l4_uint32_t yoffs, l4_uint8_t *pmap, struct gfxbitmap_offset *offset, l4_uint32_t pwidth)

  Set area from source area.
- void gfxbitmap_copy (l4_uint8_t *dest, l4_uint8_t *src, l4re_video_view_info_t *vi, int x, int y, int w, int h, int dx, int dy)

  Copy a rectangular area.

16.137.1 Detailed Description

Bitmap renderer header file.

Definition in file bitmap.h.

16.137.2 Macro Definition Documentation

16.137.2.1 pSLIM_BMAP_START_LSB

```
#define pSLIM_BMAP_START_LSB 0x01
```

the other way round

Definition at line 41 of file bitmap.h.

16.137.3 Typedef Documentation

16.137.3.1 gfxbitmap_color_pix_t

```
typedef unsigned int gfxbitmap_color_pix_t
```

Specific color type.

This color type is specific for a particular framebuffer, it can be use to write pixel on a framebuffer. Use gfxbitmap↩_convert_color to convert from gfxbitmap_color_t to gfxbitmap_color_pix_t.

Definition at line 64 of file bitmap.h.

16.137.3.2 gfxbitmap_color_t

```
typedef unsigned int gfxbitmap_color_t
```

Standard color type.

It's a RGB type with 8bits for each channel, regardless of the framebuffer used.

Definition at line 55 of file bitmap.h.

16.137.4 Function Documentation

16.137.4.1 gfxbitmap_bmap()

```
void gfxbitmap_bmap (
            l4_uint8_t * vfb,
            l4re_video_view_info_t * vi,
            l4_int16_t x,
            l4_int16_t y,
            l4_uint32_t w,
            l4_uint32_t h,
            l4_uint8_t * bmap,
            gfxbitmap_color_pix_t fgc,
            gfxbitmap_color_pix_t bgc,
            struct gfxbitmap_offset * offset,
            l4_uint8_t mode )
```

Fill a rectangular area with a bicolor bitmap pattern.

Parameters

vfb	Frame buffer.
vi	Frame buffer information structure.
x	X position of area.
y	Y position of area.
w	Width of area.
h	Height of area.
bmap	Bitmap pattern.
fgc	Foreground color.
bgc	Background color.
offset	Offsets.

See also

pSLIM_BMAP_START_MSB and pSLIM_BMAP_START_LSB.

16.137.4.2 gfxbitmap_convert_color()

```
gfxbitmap_color_pix_t gfxbitmap_convert_color (
            l4re_video_view_info_t * vi,
            gfxbitmap_color_t rgb )
```

Convert a color.

Converts a given color in standard format to the format used in the framebuffer.

16.137.4.3 gfxbitmap_copy()

```
void gfxbitmap_copy (
            l4_uint8_t * dest,
            l4_uint8_t * src,
            l4re_video_view_info_t * vi,
            int x,
            int y,
            int w,
            int h,
            int dx,
            int dy )
```

Copy a rectangular area.

Parameters

dest	Destination frame buffer.
src	Source frame buffer.
vi	Frame buffer information structure.
x	Source X position of area.
y	Source Y position of area.
w	Width of area.
h	Height of area.
dx	Source X position of area.
dy	Source Y position of area.

16.137.4.4 gfxbitmap_fill()

```
void gfxbitmap_fill (
            l4_uint8_t * vfb,
```

```
            l4re_video_view_info_t * vi,
            int x,
            int y,
            int w,
            int h,
            gfxbitmap_color_pix_t color )
```

Fill a rectangular area with a color.

Parameters

vfb	Frame buffer.
vi	Frame buffer information structure.
x	X position of area.
y	Y position of area.
w	Width of area.
h	Height of area.
color	Color of area.

16.137.4.5 gfxbitmap_set()

```
void gfxbitmap_set (
            l4_uint8_t * vfb,
            l4re_video_view_info_t * vi,
            l4_int16_t x,
            l4_int16_t y,
            l4_uint32_t w,
            l4_uint32_t h,
            l4_uint32_t xoffs,
            l4_uint32_t yoffs,
            l4_uint8_t * pmap,
            struct gfxbitmap_offset * offset,
            l4_uint32_t pwidth )
```

Set area from source area.

Parameters

vfb	Frame buffer.
vi	Frame buffer information structure.
x	X position of area.
y	Y position of area.
w	Width of area.
h	Height of area.
pmap	Source.
xoffs	X offset.
yoffs	Y offset.
offset	Offsets.
pwidth	Width of source in bytes.

16.138 bitmap.h

```
00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015 #include <l4/re/c/video/view.h>
00016
00032 EXTERN_C_BEGIN
00038 #define pSLIM_BMAP_START_MSB    0x02
00041 #define pSLIM_BMAP_START_LSB    0x01
00043
00048
00055 typedef unsigned int gfxbitmap_color_t;
00056
00064 typedef unsigned int gfxbitmap_color_pix_t;
00065
00067 struct gfxbitmap_offset
00068 {
00069   l4_uint32_t preskip_x;
00070   l4_uint32_t preskip_y;
00071   l4_uint32_t endskip_x;
00072 };
00073
00080 gfxbitmap_color_pix_t
00081 gfxbitmap_convert_color(l4re_video_view_info_t *vi, gfxbitmap_color_t rgb);
00082
00094 void
00095 gfxbitmap_fill(l4_uint8_t *vfb, l4re_video_view_info_t *vi,
00096                int x, int y, int w, int h, gfxbitmap_color_pix_t color);
00097
00115 void
00116 gfxbitmap_bmap(l4_uint8_t *vfb, l4re_video_view_info_t *vi,
00117                l4_int16_t x, l4_int16_t y, l4_uint32_t w,
00118                l4_uint32_t h, l4_uint8_t *bmap,
00119                gfxbitmap_color_pix_t fgc, gfxbitmap_color_pix_t bgc,
00120                struct gfxbitmap_offset *offset, l4_uint8_t mode);
00121
00137 void
00138 gfxbitmap_set(l4_uint8_t *vfb, l4re_video_view_info_t *vi,
00139               l4_int16_t x, l4_int16_t y, l4_uint32_t w,
00140               l4_uint32_t h, l4_uint32_t xoffs, l4_uint32_t yoffs,
00141               l4_uint8_t *pmap, struct gfxbitmap_offset *offset,
00142               l4_uint32_t pwidth);
00143
00157 void
00158 gfxbitmap_copy(l4_uint8_t *dest, l4_uint8_t *src, l4re_video_view_info_t *vi,
00159                int x, int y, int w, int h, int dx, int dy);
00162 EXTERN_C_END
```

16.139 l4/libgfxbitmap/font.h File Reference

Bitmap font renderer header file.

```
#include <l4/sys/compiler.h>
#include <l4/re/c/video/view.h>
#include <l4/libgfxbitmap/bitmap.h>
```

Include dependency graph for font.h:

Macros

- #define GFXBITMAP_DEFAULT_FONT (void ∗)0

  Constant to use for the default font.

Typedefs

- typedef void ∗ gfxbitmap_font_t

  Font.

Enumerations

- enum

  Constant for length field.

Functions

- int gfxbitmap_font_init (void)

  Initialize the library.
- gfxbitmap_font_t gfxbitmap_font_get (const char ∗name)

  Get a font descriptor.
- unsigned gfxbitmap_font_width (gfxbitmap_font_t font)

Get the font width.

- unsigned gfxbitmap_font_height (gfxbitmap_font_t font)

  Get the font height.

- void * gfxbitmap_font_data (gfxbitmap_font_t font, unsigned c)

  Get bitmap font data for a specific character.

- void gfxbitmap_font_text (void *fb, l4re_video_view_info_t *vi, gfxbitmap_font_t font, const char *text, unsigned len, unsigned x, unsigned y, gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg)

  Render a string to a framebuffer.

- void gfxbitmap_font_text_scale (void *fb, l4re_video_view_info_t *vi, gfxbitmap_font_t font, const char *text, unsigned len, unsigned x, unsigned y, gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg, int scale_x, int scale_y)

  Render a string to a framebuffer, including scaling.

16.139.1 Detailed Description

Bitmap font renderer header file.

Definition in file font.h.

16.139.2 Enumeration Type Documentation

16.139.2.1 anonymous enum

```
anonymous enum
```

Constant for length field.

Use this if the function should call strlen on the text argument itself.

Definition at line 38 of file font.h.

16.139.3 Function Documentation

16.139.3.1 gfxbitmap_font_data()

```
void* gfxbitmap_font_data (
            gfxbitmap_font_t font,
            unsigned c )
```

Get bitmap font data for a specific character.

Parameters

font	Font.
c	Character.

Returns

Pointer to bmap data, NULL on error.

16.139.3.2 gfxbitmap_font_get()

```
gfxbitmap_font_t gfxbitmap_font_get (
            const char * name )
```

Get a font descriptor.

Parameters

name	Name of the font.

Returns

A (opaque) font descriptor, or NULL if font could not be found.

16.139.3.3 gfxbitmap_font_height()

```
unsigned gfxbitmap_font_height (
            gfxbitmap_font_t font )
```

Get the font height.

Parameters

font	Font.

Returns

Font height, 0 if font height could not be retrieved.

16.139.3.4 gfxbitmap_font_init()

```
int gfxbitmap_font_init (
            void )
```

Initialize the library.

This function must be called before any other font function of this library.

Returns

0 on success, other on error

16.139.3.5 gfxbitmap_font_text()

```
void gfxbitmap_font_text (
            void * fb,
            l4re_video_view_info_t * vi,
            gfxbitmap_font_t font,
            const char * text,
            unsigned len,
            unsigned x,
            unsigned y,
            gfxbitmap_color_pix_t fg,
            gfxbitmap_color_pix_t bg )
```

Render a string to a framebuffer.

Parameters

fb	Pointer to frame buffer.
vi	Frame buffer info structure.
font	Font.
text	Text string.
len	Length of the text string.
x	Horizontal position in the frame buffer.
y	Vertical position in the frame buffer.
fg	Foreground color.
bg	Background color.

16.139.3.6 gfxbitmap_font_text_scale()

```
void gfxbitmap_font_text_scale (
            void * fb,
            l4re_video_view_info_t * vi,
            gfxbitmap_font_t font,
            const char * text,
            unsigned len,
            unsigned x,
            unsigned y,
            gfxbitmap_color_pix_t fg,
            gfxbitmap_color_pix_t bg,
```

```
int scale_x,
int scale_y )
```

Render a string to a framebuffer, including scaling.

Parameters

fb	Pointer to frame buffer.
vi	Frame buffer info structure.
font	Font.
text	Text string.
len	Length of the text string.
x	Horizontal position in the frame buffer.
y	Vertical position in the frame buffer.
fg	Foreground color.
bg	Background color.
scale↩_x	Horizonal scale factor.
scale↩_y	Vertical scale factor.

16.139.3.7 gfxbitmap_font_width()

```
unsigned gfxbitmap_font_width (
            gfxbitmap_font_t font )
```

Get the font width.

Parameters

font	Font.

Returns

Font width, 0 if font width could not be retrieved.

16.140 font.h

```
00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015 #include <l4/re/c/video/view.h>
00016 #include <l4/libgfxbitmap/bitmap.h>
00017
00027
```

```
#define GFXBITMAP_DEFAULT_FONT (void *)0

enum { GFXBITMAP_USE_STRLEN = ~0U };

EXTERN_C_BEGIN

typedef void *gfxbitmap_font_t;

L4_CV int gfxbitmap_font_init(void);

L4_CV gfxbitmap_font_t gfxbitmap_font_get(const char *name);

L4_CV unsigned
gfxbitmap_font_width(gfxbitmap_font_t font);

L4_CV unsigned
gfxbitmap_font_height(gfxbitmap_font_t font);

L4_CV void *
gfxbitmap_font_data(gfxbitmap_font_t font, unsigned c);

L4_CV void
gfxbitmap_font_text(void *fb, l4re_video_view_info_t *vi,
                    gfxbitmap_font_t font, const char *text, unsigned len,
                    unsigned x, unsigned y,
                    gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg);

L4_CV void
gfxbitmap_font_text_scale(void *fb, l4re_video_view_info_t *vi,
                          gfxbitmap_font_t font, const char *text, unsigned len,
                          unsigned x, unsigned y,
                          gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg,
                          int scale_x, int scale_y);
EXTERN_C_END
```

16.141 l4/libgfxbitmap/support File Reference

Terminal support functionality.

`#include <l4/re/video/view>`

Include dependency graph for support:

16.141.1 Detailed Description

Terminal support functionality.

Date

2009

Author

Adam Lackorzynski `adam@os.inf.tu-dresden.de`

Definition in file support.

16.142 support

```
/* vim:set ft=cpp: */
/*
 * (c) 2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */
#ifndef __LIBTERM__SUPPORT_H__
#define __LIBTERM__SUPPORT_H__

#include <l4/re/video/view>

void
libterm_init_colors(L4Re::Video::View::Info *fbi);

int
libterm_get_color(int mode, int color);

#endif
```

16.143 l4/re/c/dataspace.h File Reference

Data space C interface.

`#include <l4/sys/types.h>`

Include dependency graph for dataspace.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct l4re_ds_stats_t

  Information about the data space.

Typedefs

- typedef l4_cap_idx_t l4re_ds_t

  Dataspace type.

Enumerations

- enum l4re_ds_map_flags { }

  Flags to specify the memory mapping type of a request.

Functions

- long l4re_ds_clear (l4re_ds_t ds, l4re_ds_offset_t offset, l4re_ds_size_t size) L4_NOTHROW
- long l4re_ds_allocate (l4re_ds_t ds, l4re_ds_offset_t offset, l4re_ds_size_t size) L4_NOTHROW
- int l4re_ds_copy_in (l4re_ds_t ds, l4re_ds_offset_t dst_offs, l4re_ds_t src, l4re_ds_offset_t src_offs, l4re_ds_size_t size) L4_NOTHROW
- l4re_ds_size_t l4re_ds_size (l4re_ds_t ds) L4_NOTHROW
- l4re_ds_flags_t l4re_ds_flags (l4re_ds_t ds) L4_NOTHROW
- int l4re_ds_info (l4re_ds_t ds, l4re_ds_stats_t *stats) L4_NOTHROW

16.143.1 Detailed Description

Data space C interface.

Definition in file dataspace.h.

16.144 dataspace.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00031 #include <l4/sys/types.h>
00032
00033 EXTERN_C_BEGIN
00034
```

```
typedef l4_cap_idx_t l4re_ds_t;
typedef l4_uint64_t l4re_ds_size_t;
typedef l4_uint64_t l4re_ds_offset_t;
typedef l4_uint64_t l4re_ds_map_addr_t;
typedef unsigned long l4re_ds_flags_t;

typedef struct {
  l4re_ds_size_t size;
  l4re_ds_flags_t flags;
} l4re_ds_stats_t;

enum l4re_ds_map_flags {
  L4RE_DS_F_R   = L4_FPAGE_RO,
  L4RE_DS_F_W   = L4_FPAGE_W,
  L4RE_DS_F_X   = L4_FPAGE_X,
  L4RE_DS_F_RW  = L4_FPAGE_RW,
  L4RE_DS_F_RX  = L4_FPAGE_RX,
  L4RE_DS_F_RWX = L4_FPAGE_RWX,

  L4RE_DS_F_RIGHTS_MASK = 0x0f,

  L4RE_DS_F_NORMAL        = 0x00,
  L4RE_DS_F_CACHEABLE     = L4RE_DS_F_NORMAL,
  L4RE_DS_F_BUFFERABLE    = 0x10,
  L4RE_DS_F_UNCACHEABLE   = 0x20,
  L4RE_DS_F_CACHING_MASK  = 0x30,
  L4RE_DS_F_CACHING_SHIFT = 4,
};

L4_CV int
l4re_ds_map(l4re_ds_t ds,
            l4re_ds_offset_t offset,
            l4re_ds_flags_t flags,
            l4re_ds_map_addr_t local_addr,
            l4re_ds_map_addr_t min_addr,
            l4re_ds_map_addr_t max_addr) L4_NOTHROW;

L4_CV int
l4re_ds_map_region(l4re_ds_t ds,
                   l4re_ds_offset_t offset,
                   l4re_ds_flags_t flags,
                   l4re_ds_map_addr_t min_addr,
                   l4re_ds_map_addr_t max_addr) L4_NOTHROW;

L4_CV long
l4re_ds_clear(l4re_ds_t ds, l4re_ds_offset_t offset,
              l4re_ds_size_t size) L4_NOTHROW;

L4_CV long
l4re_ds_allocate(l4re_ds_t ds,
                 l4re_ds_offset_t offset,
                 l4re_ds_size_t size) L4_NOTHROW;

L4_CV int
l4re_ds_copy_in(l4re_ds_t ds, l4re_ds_offset_t dst_offs,
                l4re_ds_t src, l4re_ds_offset_t src_offs,
                l4re_ds_size_t size) L4_NOTHROW;

L4_CV l4re_ds_size_t
l4re_ds_size(l4re_ds_t ds) L4_NOTHROW;

L4_CV l4re_ds_flags_t
l4re_ds_flags(l4re_ds_t ds) L4_NOTHROW;

L4_CV int
l4re_ds_info(l4re_ds_t ds, l4re_ds_stats_t *stats) L4_NOTHROW;

EXTERN_C_END
```

16.145 l4/re/c/debug.h File Reference

Debug C interface.

`#include <l4/sys/types.h>`

Include dependency graph for debug.h:

Functions

- long l4re_debug_obj_debug (l4_cap_idx_t srv, unsigned long function) L4_NOTHROW

  Call debug function of L4Re service.

16.145.1 Detailed Description

Debug C interface.

Definition in file debug.h.

16.146 debug.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00029 #include <l4/sys/types.h>
00030
00031 EXTERN_C_BEGIN
00032
00040 L4_CV long
00041 l4re_debug_obj_debug(l4_cap_idx_t srv, unsigned long function) L4_NOTHROW;
00042
00043 EXTERN_C_END
```

16.147 l4/re/c/dma_space.h File Reference

DMA space C interface.

```
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
```

Include dependency graph for dma_space.h:

Typedefs

- typedef l4_cap_idx_t l4re_dma_space_t

  DMA space capability type.
- typedef l4_uint64_t l4re_dma_space_dma_addr_t

  Data type for DMA addresses.

Enumerations

- enum l4re_dma_space_direction { L4RE_DMA_SPACE_BIDIRECTIONAL , L4RE_DMA_SPACE_TO_DEVICE , L4RE_DMA_SPACE_FROM_DEVICE , L4RE_DMA_SPACE_NONE }

  Direction of the DMA transfers.
- enum l4re_dma_space_space_attribs { L4RE_DMA_SPACE_COHERENT = 1 << 0 , L4RE_DMA_SPACE_PHYS_SPACE = 1 << 1 }

  Attributes assigned to the DMA space when associated with a specific device.

Functions

- long l4re_dma_space_map (l4re_dma_space_t dma, l4re_ds_t src, l4re_ds_offset_t offset, l4_size_t *size, unsigned long attrs, enum l4re_dma_space_direction dir, l4re_dma_space_dma_addr_t *dma_addr) L4_NOTHROW

  Map the given part of this data space into the DMA address space.
- long l4re_dma_space_unmap (l4re_dma_space_t dma, l4re_dma_space_dma_addr_t dma_addr, l4_size_t size, unsigned long attrs, enum l4re_dma_space_direction dir) L4_NOTHROW

  Unmap the given part of this data space from the DMA address space.
- long l4re_dma_space_associate (l4re_dma_space_t dma, l4_cap_idx_t dma_task, unsigned long attr) L4_NOTHROW

  Associate a DMA task for a device to this Dma_space.
- long l4re_dma_space_disassociate (l4re_dma_space_t dma)

  Disassociate the DMA task from this Dma_space.

16.147.1 Detailed Description

DMA space C interface.

Definition in file dma_space.h.

16.147.2 Enumeration Type Documentation

16.147.2.1 l4re_dma_space_direction

`enum l4re_dma_space_direction`

Direction of the DMA transfers.

Enumerator

L4RE_DMA_SPACE_BIDIRECTIONAL	device reads and writes to the memory
L4RE_DMA_SPACE_TO_DEVICE	device reads the memory
L4RE_DMA_SPACE_FROM_DEVICE	device writes to the memory
L4RE_DMA_SPACE_NONE	device is coherently connected

Definition at line 37 of file dma_space.h.

16.147.2.2 l4re_dma_space_space_attribs

`enum l4re_dma_space_space_attribs`

Attributes assigned to the DMA space when associated with a specific device.

See also

Space_attribs

Enumerator

L4RE_DMA_SPACE_COHERENT	The device is connected coherently with the cache. This means that the map() and unmap() do not need to sync CPU caches before and after DMA.
L4RE_DMA_SPACE_PHYS_SPACE	The DMA space has no DMA task assigned and uses the CPUs physical memory.

Definition at line 48 of file dma_space.h.

16.148 dma_space.h

```
00001
00005 /*
00006  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction.  Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License.  This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021 #pragma once
00022
00029 #include <l4/sys/types.h>
00030 #include <l4/re/c/dataspace.h>
00031
00032 EXTERN_C_BEGIN
00033
00037 enum l4re_dma_space_direction
00038 {
00039   L4RE_DMA_SPACE_BIDIRECTIONAL,
00040   L4RE_DMA_SPACE_TO_DEVICE,
00041   L4RE_DMA_SPACE_FROM_DEVICE,
00042   L4RE_DMA_SPACE_NONE
00043 };
00044
00048 enum l4re_dma_space_space_attribs
00049 {
00050   L4RE_DMA_SPACE_COHERENT   = 1 « 0,
00051   L4RE_DMA_SPACE_PHYS_SPACE = 1 « 1,
00052 };
00053
00059 typedef l4_cap_idx_t l4re_dma_space_t;
00060
00062 typedef l4_uint64_t l4re_dma_space_dma_addr_t;
00063
00069 L4_CV long
00070 l4re_dma_space_map(l4re_dma_space_t dma, l4re_ds_t src,
00071                    l4re_ds_offset_t offset,
00072                    l4_size_t * size, unsigned long attrs,
00073                    enum l4re_dma_space_direction dir,
00074                    l4re_dma_space_dma_addr_t *dma_addr) L4_NOTHROW;
00075
00076
00082 L4_CV long
00083 l4re_dma_space_unmap(l4re_dma_space_t dma, l4re_dma_space_dma_addr_t dma_addr,
00084                      l4_size_t size, unsigned long attrs,
00085                      enum l4re_dma_space_direction dir) L4_NOTHROW;
00086
00092 L4_CV long
00093 l4re_dma_space_associate(l4re_dma_space_t dma, l4_cap_idx_t dma_task,
00094                          unsigned long attr) L4_NOTHROW;
00095
```

```
L4_CV long
l4re_dma_space_disassociate(l4re_dma_space_t dma);


EXTERN_C_END
```

16.149 l4/re/c/event.h File Reference

Event C interface.

```
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/event.h>
```

Include dependency graph for event.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct l4re_event_t

  Event structure used in buffer.

Functions

- long l4re_event_get_buffer (const l4_cap_idx_t server, const l4re_ds_t ds) L4_NOTHROW

  Get an event signal buffer.
- long l4re_event_get_num_streams (const l4_cap_idx_t server) L4_NOTHROW

  Get number of streams.
- long l4re_event_get_stream_info (const l4_cap_idx_t server, int idx, l4re_event_stream_info_t *info) L4_NOTHROW

  Get information on a stream.
- long l4re_event_get_stream_info_for_id (const l4_cap_idx_t server, l4_umword_t stream_id, l4re_event_stream_info_t *info) L4_NOTHROW

  Get info for a stream given a stream id.
- long l4re_event_get_axis_info (const l4_cap_idx_t server, l4_umword_t id, unsigned naxes, unsigned const *axis, l4re_event_absinfo_t *info) L4_NOTHROW

  Get Axis information for a stream.

16.149.1 Detailed Description

Event C interface.

Definition in file event.h.

16.150 event.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00031 #include <l4/sys/types.h>
00032 #include <l4/re/c/dataspace.h>
00033 #include <l4/re/event.h>
00034
00035 EXTERN_C_BEGIN
00036
00040 typedef struct
00041 {
00042   long long time;
00043   unsigned short type;
00044   unsigned short code;
00045   int value;
00046   l4_umword_t stream_id;
00047 } l4re_event_t;
00048
00060 L4_CV long
00061 l4re_event_get_buffer(const l4_cap_idx_t server,
00062                       const l4re_ds_t ds) L4_NOTHROW;
00063
00074 L4_CV long
00075 l4re_event_get_num_streams(const l4_cap_idx_t server) L4_NOTHROW;
00076
00089 L4_CV long
```

```
l4re_event_get_stream_info(const l4_cap_idx_t server,
                           int idx, l4re_event_stream_info_t *info) L4_NOTHROW;

L4_CV long
l4re_event_get_stream_info_for_id(const l4_cap_idx_t server,
                                  l4_umword_t stream_id,
                                  l4re_event_stream_info_t *info) L4_NOTHROW;

L4_CV long
l4re_event_get_axis_info(const l4_cap_idx_t server, l4_umword_t id,
                         unsigned naxes, unsigned const *axis,
                         l4re_event_absinfo_t *info) L4_NOTHROW;

EXTERN_C_END
```

16.151 l4/re/event.h File Reference

Events.

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
```

Include dependency graph for event.h:

This graph shows which files directly or indirectly include this file:

16.151.1 Detailed Description

Events.

Definition in file event.h.

16.152 event.h

```
00001
00005 /*
00006  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <l4/sys/compiler.h>
00025 #include <l4/sys/l4int.h>
00026
00027 typedef struct L4_EXPORT_TYPE l4re_event_stream_id_t
00028 {
00029   l4_uint16_t bustype;
00030   l4_uint16_t vendor;
00031   l4_uint16_t product;
00032   l4_uint16_t version;
00033 } l4re_event_stream_id_t;
00034
00035 typedef struct L4_EXPORT_TYPE l4re_event_absinfo_t
00036 {
00037   l4_int32_t value;
00038   l4_int32_t min;
00039   l4_int32_t max;
00040   l4_int32_t fuzz;
00041   l4_int32_t flat;
00042   l4_int32_t resolution;
00043 } l4re_event_absinfo_t;
00044
00045 enum l4re_event_stream_max_values_t
00046 {
00047   L4RE_EVENT_EV_MAX  = 0x1f,
00048   L4RE_EVENT_KEY_MAX = 0x1ff,
00049   L4RE_EVENT_REL_MAX = 0xf,
00050   L4RE_EVENT_ABS_MAX = 0x3f,
00051   L4RE_EVENT_PROP_MAX = 0x1f,
00052   L4RE_EVENT_SW_MAX   = 0xf, // should be >= L4RE_SW_MAX
00053 };
00054
00055 enum l4re_event_stream_props_t
00056 {
00057   L4RE_EVENT_STREAM_CALIBRATE = 0x001,
00058 };
00059
00060
00061 #define __UNUM_B(x) ((x+1) + sizeof(unsigned long)*8 - 1) / (sizeof(unsigned long)*8)
00062
00063 typedef struct L4_EXPORT_TYPE l4re_event_stream_info_t
00064 {
00065   l4_umword_t stream_id;
00066   char name[32];
00067   char phys[32];
00068   l4re_event_stream_id_t id;
00069
00070   unsigned long propbits[__UNUM_B(L4RE_EVENT_PROP_MAX)];
00071
00072   unsigned long evbits[__UNUM_B(L4RE_EVENT_EV_MAX)];
00073   unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00074   unsigned long relbits[__UNUM_B(L4RE_EVENT_REL_MAX)];
00075   unsigned long absbits[__UNUM_B(L4RE_EVENT_ABS_MAX)];
```

```
  unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];

} l4re_event_stream_info_t;

typedef struct L4_EXPORT_TYPE l4re_event_stream_state_t
{
  unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
  unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
} l4re_event_stream_state_t;

#undef __UNUM_B

```

16.153 l4/re/c/log.h File Reference

Log C interface.

`#include <l4/re/env.h>`

Include dependency graph for log.h:

Functions

- void l4re_log_print (char const ∗string) L4_NOTHROW

  Write a null terminated string to the default log.
- void l4re_log_printn (char const ∗string, int len) L4_NOTHROW

  Write a string of a given length to the default log.
- void l4re_log_print_srv (const l4_cap_idx_t logcap, char const ∗string) L4_NOTHROW

  Write a null terminated string to a log.
- void l4re_log_printn_srv (const l4_cap_idx_t logcap, char const ∗string, int len) L4_NOTHROW

  Write a string of a given length to a log.

16.153.1 Detailed Description

Log C interface.

Definition in file log.h.

16.154 log.h

```
00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00030 #include <l4/re/env.h>
00031
00032 EXTERN_C_BEGIN
00033
00044 L4_CV L4_INLINE void
00045 l4re_log_print(char const *string) L4_NOTHROW;
00046
00058 L4_CV L4_INLINE void
00059 l4re_log_printn(char const *string, int len) L4_NOTHROW;
00060
00061
00062
00063
00075 L4_CV void
00076 l4re_log_print_srv(const l4_cap_idx_t logcap,
00077                    char const *string) L4_NOTHROW;
00078
00091 L4_CV void
00092 l4re_log_printn_srv(const l4_cap_idx_t logcap,
00093                     char const *string, int len) L4_NOTHROW;
00094
00095
00096 /********** Implementations **************************/
00097
00098 L4_CV L4_INLINE void
00099 l4re_log_print(char const *string) L4_NOTHROW
00100 {
00101   l4re_log_print_srv(l4re_global_env->log, string);
00102 }
00103
00104 L4_CV L4_INLINE void
00105 l4re_log_printn(char const *string, int len) L4_NOTHROW
00106 {
00107   l4re_log_printn_srv(l4re_global_env->log, string, len);
00108 }
00109
00110 EXTERN_C_END
```

16.155 l4/re/c/mem_alloc.h File Reference

Memory allocator C interface.

```
#include <l4/re/env.h>
#include <l4/sys/consts.h>
#include <l4/re/c/dataspace.h>
```

Include dependency graph for mem_alloc.h:

Enumerations

- enum l4re_ma_flags

  Flags for requesting memory at the memory allocator.

Functions

- long l4re_ma_alloc (long size, l4re_ds_t const mem, unsigned long flags) L4_NOTHROW

  Allocate memory.
- long l4re_ma_alloc_align (long size, l4re_ds_t const mem, unsigned long flags, unsigned long align) L4_NOTHROW

  Allocate memory.
- long l4re_ma_alloc_align_srv (l4_cap_idx_t srv, long size, l4re_ds_t const mem, unsigned long flags, unsigned long align) L4_NOTHROW

  Allocate memory.

16.155.1 Detailed Description

Memory allocator C interface.

Definition in file mem_alloc.h.

16.156 mem_alloc.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <l4/re/env.h>
00025 #include <l4/sys/consts.h>
00026
00027 #include <l4/re/c/dataspace.h>
00028
00035 EXTERN_C_BEGIN
00036
00042 enum l4re_ma_flags {
00043   L4RE_MA_CONTINUOUS  = 0x01,
00044   L4RE_MA_PINNED      = 0x02,
00045   L4RE_MA_SUPER_PAGES = 0x04,
00046 };
00047
00048
00079 L4_CV L4_INLINE long
00080 l4re_ma_alloc(long size, l4re_ds_t const mem,
00081               unsigned long flags) L4_NOTHROW;
00082
00116 L4_CV L4_INLINE long
00117 l4re_ma_alloc_align(long size, l4re_ds_t const mem,
00118                     unsigned long flags, unsigned long align) L4_NOTHROW;
00119
00138 L4_CV long
00139 l4re_ma_alloc_align_srv(l4_cap_idx_t srv, long size,
00140                         l4re_ds_t const mem, unsigned long flags,
00141                         unsigned long align) L4_NOTHROW;
00142
00143 /***************** Implementation *****************/
00144
00145 L4_CV L4_INLINE long
00146 l4re_ma_alloc(long size, l4re_ds_t const mem,
00147               unsigned long flags) L4_NOTHROW
00148 {
00149   return l4re_ma_alloc_align_srv(l4re_global_env->mem_alloc, size, mem,
00150                                  flags, 0);
00151 }
00152
00153 L4_CV L4_INLINE long
00154 l4re_ma_alloc_align(long size, l4re_ds_t const mem,
00155                     unsigned long flags, unsigned long align) L4_NOTHROW
00156 {
00157   return l4re_ma_alloc_align_srv(l4re_global_env->mem_alloc, size, mem,
00158                                  flags, align);
00159 }
00160
00161 EXTERN_C_END
```

16.157 l4/re/c/namespace.h File Reference

Namespace functions, C interface.

`#include <l4/re/env.h>`
Include dependency graph for namespace.h:

Typedefs

- typedef l4_cap_idx_t l4re_namespace_t

  Namespace type.

Enumerations

- enum l4re_ns_register_flags

  Namespace register flags.

Functions

- long l4re_ns_query_to_srv (l4re_namespace_t srv, char const ∗name, l4_cap_idx_t const cap, int timeout) L4_NOTHROW

  Query the name space for the object named by `name`.
- long l4re_ns_query_srv (l4re_namespace_t srv, char const ∗name, l4_cap_idx_t const cap) L4_NOTHROW

  Query the name space for the object named by `name`.
- long l4re_ns_register_obj_srv (l4re_namespace_t srv, char const ∗name, l4_cap_idx_t const obj, unsigned flags) L4_NOTHROW

  Register an object with a name.

16.157.1 Detailed Description

Namespace functions, C interface.

Definition in file namespace.h.

16.158 namespace.h

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/re/env.h>

enum l4re_ns_register_flags {
  L4RE_NS_REGISTER_RO  = L4_FPAGE_RO,
  L4RE_NS_REGISTER_DIR = 0x10,
  L4RE_NS_REGISTER_RW  = L4_FPAGE_RX,
  L4RE_NS_REGISTER_RWS = L4_FPAGE_RWX,
  L4RE_NS_REGISTER_S   = L4_FPAGE_W,
};

EXTERN_C_BEGIN

typedef l4_cap_idx_t l4re_namespace_t;



L4_CV long
l4re_ns_query_to_srv(l4re_namespace_t srv, char const *name,
                     l4_cap_idx_t const cap, int timeout) L4_NOTHROW;

L4_CV L4_INLINE long
l4re_ns_query_srv(l4re_namespace_t srv, char const *name,
                  l4_cap_idx_t const cap) L4_NOTHROW;

L4_CV long
l4re_ns_register_obj_srv(l4re_namespace_t srv, char const *name,
                         l4_cap_idx_t const obj, unsigned flags) L4_NOTHROW;



/****** Implementation ***********/

L4_CV L4_INLINE long
l4re_ns_query_srv(l4re_namespace_t srv, char const *name,
                  l4_cap_idx_t const cap) L4_NOTHROW
{
  return l4re_ns_query_to_srv(srv, name, cap, 40000);
}


EXTERN_C_END
```

16.159 l4/re/c/rm.h File Reference

Region map interface, C interface.

```
#include <l4/re/env.h>
#include <l4/re/c/dataspace.h>
```

Include dependency graph for rm.h:

Enumerations

- enum l4re_rm_flags_values {
  L4RE_RM_F_R = L4RE_DS_F_R , **L4RE_RM_F_W** = L4RE_DS_F_W , **L4RE_RM_F_X** = L4RE_DS_F_X , **L4RE_RM_F_RX** = L4RE_DS_F_RX ,
  L4RE_RM_F_RW = L4RE_DS_F_RW , **L4RE_RM_F_RWX** = L4RE_DS_F_RWX , L4RE_RM_F_NO_ALIAS = 0x200 , L4RE_RM_F_PAGER = 0x400 ,
  L4RE_RM_F_RESERVED = 0x800 , L4RE_RM_CACHING_SHIFT = 4 , L4RE_RM_F_CACHING = L4RE↩_DS_F_CACHING_MASK , L4RE_RM_REGION_FLAGS = 0xffff ,
  L4RE_RM_F_CACHE_NORMAL = L4RE_DS_F_NORMAL , L4RE_RM_F_CACHE_BUFFERED = L4↩RE_DS_F_BUFFERABLE , L4RE_RM_F_CACHE_UNCACHED = L4RE_DS_F_UNCACHEABLE ,
  L4RE_RM_F_SEARCH_ADDR = 0x20000 ,
  L4RE_RM_F_IN_AREA = 0x40000 , L4RE_RM_F_EAGER_MAP = 0x80000 , L4RE_RM_F_ATTACH_FLAGS = 0xf0000 }

  Flags for region operations.

Functions

- int l4re_rm_reserve_area (l4_addr_t ∗start, unsigned long size, l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW

  Reserve the given area in the region map.
- int l4re_rm_free_area (l4_addr_t addr) L4_NOTHROW

  Free an area from the region map.
- int l4re_rm_attach (void ∗∗start, unsigned long size, l4re_rm_flags_t flags, l4re_ds_t mem, l4re_rm_offset_t offs, unsigned char align) L4_NOTHROW

  Attach a data space to a region.
- int l4re_rm_detach (void ∗addr) L4_NOTHROW

Detach and unmap a region from the address space in the current task.

- int l4re_rm_detach_ds (void *addr, l4re_ds_t *ds) L4_NOTHROW

  Detach and unmap a region and return affected dataspace in the current task.

- int l4re_rm_detach_unmap (l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW

  Detach and unmap in specified task.

- int l4re_rm_detach_ds_unmap (void *addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW

  Detach and unmap in specified task.

- int l4re_rm_find (l4_addr_t *addr, unsigned long *size, l4re_rm_offset_t *offset, l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW

  Find a region given an address and size.

- void l4re_rm_show_lists (void) L4_NOTHROW

  Dump region map internal data structures.

- int l4re_rm_reserve_area_srv (l4_cap_idx_t rm, l4_addr_t *start, unsigned long size, l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW
- int l4re_rm_free_area_srv (l4_cap_idx_t rm, l4_addr_t addr) L4_NOTHROW
- int l4re_rm_attach_srv (l4_cap_idx_t rm, void **start, unsigned long size, l4re_rm_flags_t flags, l4re_ds_t mem, l4re_rm_offset_t offs, unsigned char align) L4_NOTHROW
- int l4re_rm_detach_srv (l4_cap_idx_t rm, l4_addr_t addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW
- int l4re_rm_find_srv (l4_cap_idx_t rm, l4_addr_t *addr, unsigned long *size, l4re_rm_offset_t *offset, l4re↩_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW
- void l4re_rm_show_lists_srv (l4_cap_idx_t rm) L4_NOTHROW

  Dump region map internal data structures.

16.159.1 Detailed Description

Region map interface, C interface.

Definition in file rm.h.

16.160 rm.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00031 #include <l4/re/env.h>
00032 #include <l4/re/c/dataspace.h>
00033
00034 EXTERN_C_BEGIN
00035
00040 enum l4re_rm_flags_values {
00041   L4RE_RM_F_R     = L4RE_DS_F_R,
00042   L4RE_RM_F_W     = L4RE_DS_F_W,
00043   L4RE_RM_F_X     = L4RE_DS_F_X,
00044   L4RE_RM_F_RX    = L4RE_DS_F_RX,
00045   L4RE_RM_F_RW    = L4RE_DS_F_RW,
```

```
  L4RE_RM_F_RWX  = L4RE_DS_F_RWX,

  L4RE_RM_F_NO_ALIAS     = 0x200,
  L4RE_RM_F_PAGER        = 0x400,
  L4RE_RM_F_RESERVED     = 0x800,
  L4RE_RM_CACHING_SHIFT    = 4,
  L4RE_RM_F_CACHING      = L4RE_DS_F_CACHING_MASK,

  L4RE_RM_REGION_FLAGS = 0xffff,
  L4RE_RM_F_CACHE_NORMAL   = L4RE_DS_F_NORMAL,

  L4RE_RM_F_CACHE_BUFFERED = L4RE_DS_F_BUFFERABLE,

  L4RE_RM_F_CACHE_UNCACHED = L4RE_DS_F_UNCACHEABLE,

  L4RE_RM_F_SEARCH_ADDR  = 0x20000,
  L4RE_RM_F_IN_AREA      = 0x40000,
  L4RE_RM_F_EAGER_MAP    = 0x80000,
  L4RE_RM_F_ATTACH_FLAGS = 0xf0000,
};

typedef l4_uint32_t l4re_rm_flags_t;
typedef l4_uint64_t l4re_rm_offset_t;

L4_CV L4_INLINE int
l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
                     l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW;

L4_CV L4_INLINE int
l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW;

L4_CV L4_INLINE int
l4re_rm_attach(void **start, unsigned long size, l4re_rm_flags_t flags,
               l4re_ds_t mem,
               l4re_rm_offset_t offs,
               unsigned char align) L4_NOTHROW;


L4_CV L4_INLINE int
l4re_rm_detach(void *addr) L4_NOTHROW;

L4_CV L4_INLINE int
l4re_rm_detach_ds(void *addr, l4re_ds_t *ds) L4_NOTHROW;

L4_CV L4_INLINE int
l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW;

L4_CV L4_INLINE int
l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds,
                        l4_cap_idx_t task) L4_NOTHROW;


L4_CV L4_INLINE int
l4re_rm_find(l4_addr_t *addr, unsigned long *size,
             l4re_rm_offset_t *offset,
             l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW;

L4_CV L4_INLINE void
l4re_rm_show_lists(void) L4_NOTHROW;


/*
 * Variants of functions that also take a capability of the region map
 * service.
 */


L4_CV int
l4re_rm_reserve_area_srv(l4_cap_idx_t rm, l4_addr_t *start, unsigned long size,
                         l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW;

L4_CV int
l4re_rm_free_area_srv(l4_cap_idx_t rm, l4_addr_t addr) L4_NOTHROW;

L4_CV int
l4re_rm_attach_srv(l4_cap_idx_t rm, void **start, unsigned long size,
                   l4re_rm_flags_t flags, l4re_ds_t mem,
                   l4re_rm_offset_t offs,
                   unsigned char align) L4_NOTHROW;


L4_CV int
l4re_rm_detach_srv(l4_cap_idx_t rm, l4_addr_t addr,
                   l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW;


L4_CV int
```

```
l4re_rm_find_srv(l4_cap_idx_t rm, l4_addr_t *addr,
                 unsigned long *size, l4re_rm_offset_t *offset,
                 l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW;

L4_CV void
l4re_rm_show_lists_srv(l4_cap_idx_t rm) L4_NOTHROW;


/********** Implementations **************************/

L4_CV L4_INLINE int
l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
                     l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW
{
  return l4re_rm_reserve_area_srv(l4re_global_env->rm, start, size,
                                  flags, align);
}

L4_CV L4_INLINE int
l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW
{
  return l4re_rm_free_area_srv(l4re_global_env->rm, addr);
}

L4_CV L4_INLINE int
l4re_rm_attach(void **start, unsigned long size, l4re_rm_flags_t flags,
               l4re_ds_t mem, l4re_rm_offset_t offs,
               unsigned char align) L4_NOTHROW
{
  return l4re_rm_attach_srv(l4re_global_env->rm, start, size,
                            flags, mem, offs, align);
}


L4_CV L4_INLINE int
l4re_rm_detach(void *addr) L4_NOTHROW
{
  return l4re_rm_detach_srv(l4re_global_env->rm,
                            (l4_addr_t)addr, 0, L4_BASE_TASK_CAP);
}

L4_CV L4_INLINE int
l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW
{
  return l4re_rm_detach_srv(l4re_global_env->rm, addr, 0, task);
}

L4_CV L4_INLINE int
l4re_rm_detach_ds(void *addr, l4re_ds_t *ds) L4_NOTHROW
{
  return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
                            ds, L4_BASE_TASK_CAP);
}

L4_CV L4_INLINE int
l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW
{
  return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
                            ds, task);
}

L4_CV L4_INLINE int
l4re_rm_find(l4_addr_t *addr, unsigned long *size,
             l4re_rm_offset_t *offset,
             l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW
{
  return l4re_rm_find_srv(l4re_global_env->rm, addr, size, offset, flags, m);
}

L4_CV L4_INLINE void
l4re_rm_show_lists(void) L4_NOTHROW
{
  l4re_rm_show_lists_srv(l4re_global_env->rm);
}

EXTERN_C_END
```

16.161 l4/re/c/util/cap_alloc.h File Reference

Capability allocator C interface.

```
#include <l4/sys/types.h>
#include <l4/sys/linkage.h>
```

Include dependency graph for cap_alloc.h:

Functions

- l4_cap_idx_t l4re_util_cap_alloc (void) L4_NOTHROW

  Get free capability index at capability allocator.
- void l4re_util_cap_free (l4_cap_idx_t cap) L4_NOTHROW

  Return capability index to capability allocator.
- void l4re_util_cap_free_um (l4_cap_idx_t cap) L4_NOTHROW

  Return capability index to capability allocator, and unmaps the object.
- long l4re_util_cap_last (void) L4_NOTHROW

  Return last capability index the allocator can return.

16.161.1 Detailed Description

Capability allocator C interface.

Definition in file cap_alloc.h.

16.162 cap_alloc.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
```

```
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/sys/types.h>
#include <l4/sys/linkage.h>

EXTERN_C_BEGIN

L4_CV l4_cap_idx_t
l4re_util_cap_alloc(void) L4_NOTHROW;

L4_CV void
l4re_util_cap_free(l4_cap_idx_t cap) L4_NOTHROW;

L4_CV void
l4re_util_cap_free_um(l4_cap_idx_t cap) L4_NOTHROW;

L4_CV long
l4re_util_cap_last(void) L4_NOTHROW;

EXTERN_C_END
```

16.163 l4/re/c/util/kumem_alloc.h File Reference

Kumem allocator utility C interface.

```
#include <l4/sys/types.h>
#include <l4/sys/linkage.h>
```

Include dependency graph for kumem_alloc.h:

Functions

- int l4re_util_kumem_alloc (l4_addr_t *mem, unsigned pages_order, l4_cap_idx_t task, l4_cap_idx_t rm) L4_NOTHROW

  Allocate state area.

16.163.1 Detailed Description

Kumem allocator utility C interface.

Definition in file kumem_alloc.h.

16.163.2 Function Documentation

16.163.2.1 l4re_util_kumem_alloc()

```
int l4re_util_kumem_alloc (
            l4_addr_t * mem,
            unsigned pages_order,
            l4_cap_idx_t task,
            l4_cap_idx_t rm )
```

Allocate state area.

Parameters

out	*mem*	Pointer to memory that has been allocated.
	pages_order	Size to allocate, in log2 pages.
	task	Task to use for allocation.
	rm	Region manager to use for allocation.

Return values

0	for success
<0	error code on failure

Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page. A portable implementation should not depend on allocations greater than 16KiB to succeed.

Examples

examples/sys/aliens/main.c, and examples/sys/utcb-ipc/main.c.

16.164 kumem_alloc.h

```
00001
00005 /*
00006  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *          Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
```

```
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/sys/types.h>
#include <l4/sys/linkage.h>

EXTERN_C_BEGIN

L4_CV int
l4re_util_kumem_alloc(l4_addr_t *mem, unsigned pages_order,
                      l4_cap_idx_t task, l4_cap_idx_t rm) L4_NOTHROW;

EXTERN_C_END
```

16.165 l4/re/c/util/video/goos_fb.h File Reference

Framebuffer utility functionality.

```
#include <l4/sys/compiler.h>
#include <l4/re/c/video/goos.h>
#include <l4/sys/err.h>
#include <l4/re/c/dataspace.h>
```

Include dependency graph for goos_fb.h:

16.165.1 Detailed Description

Framebuffer utility functionality.

Definition in file goos_fb.h.

16.166 goos_fb.h

```
00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <l4/sys/compiler.h>
00025 #include <l4/re/c/video/goos.h>
00026 #include <l4/sys/err.h>
00027 #include <l4/re/c/dataspace.h>
00028
00029 EXTERN_C_BEGIN
00030
00031 typedef struct
00032 {
00033   unsigned long _obj_buf[6];
00034 } l4re_util_video_goos_fb_t;
00035
00036 L4_CV int
00037 l4re_util_video_goos_fb_setup_name(l4re_util_video_goos_fb_t *goosfb,
00038                                    char const *name) L4_NOTHROW;
00039
00040 L4_CV void
00041 l4re_util_video_goos_fb_destroy(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00042
00043 L4_CV int
00044 l4re_util_video_goos_fb_view_info(l4re_util_video_goos_fb_t *goosfb,
00045                                   l4re_video_view_info_t *info) L4_NOTHROW;
00046
00047 L4_CV void *
00048 l4re_util_video_goos_fb_attach_buffer(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00049
00050 L4_CV int
00051 l4re_util_video_goos_fb_refresh(l4re_util_video_goos_fb_t *goosfb,
00052                                int x, int y, int w, int h) L4_NOTHROW;
00053
00054 L4_CV l4re_ds_t
00055 l4re_util_video_goos_fb_buffer(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00056
00057 L4_CV l4_cap_idx_t
00058 l4re_util_video_goos_fb_goos(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00059
00060 EXTERN_C_END
```

16.167 l4/re/c/video/colors.h File Reference

```
#include <l4/sys/compiler.h>
```

Include dependency graph for colors.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct l4re_video_color_component_t

  Color component structure.
- struct l4re_video_pixel_info_t

  Pixel_info structure.

Typedefs

- typedef struct l4re_video_color_component_t l4re_video_color_component_t

  Color component structure.
- typedef struct l4re_video_pixel_info_t l4re_video_pixel_info_t

  Pixel_info structure.

16.167.1 Detailed Description

Note

The C interface of L4Re::Video does *NOT* reflect the full C++ interface on purpose. Use the C++ interface where possible.

Definition in file colors.h.

16.168 colors.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/sys/compiler.h>
00026
00031 typedef struct l4re_video_color_component_t
00032 {
00033   unsigned char size;
00034   unsigned char shift;
00035 } __attribute__((packed)) l4re_video_color_component_t;
00036
00041 typedef struct l4re_video_pixel_info_t
00042 {
00043   l4re_video_color_component_t r, g, b, a;
00044   unsigned char bytes_per_pixel;
00045 } l4re_video_pixel_info_t;
00046
00047 EXTERN_C_BEGIN
00048
00049 L4_INLINE L4_CV int
00050 l4re_video_bits_per_pixel(l4re_video_pixel_info_t *p) L4_NOTHROW;
00051
00052 /* ************************************************************ */
00053 /* Implementations */
00054
00055 L4_INLINE L4_CV int
00056 l4re_video_bits_per_pixel(l4re_video_pixel_info_t *p) L4_NOTHROW
00057 {
00058   return p->r.size + p->b.size + p->g.size + p->a.size;
00059 }
00060
00061 EXTERN_C_END
```

16.169 l4/re/c/video/goos.h File Reference

```
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>
#include <l4/re/c/video/view.h>
```

Include dependency graph for goos.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct l4re_video_goos_info_t

  Goos information structure.

Typedefs

- typedef l4_cap_idx_t l4re_video_goos_t

  Goos object type.

Enumerations

- enum l4re_video_goos_info_flags_t { F_l4re_video_goos_auto_refresh = 0x01 , F_l4re_video_goos_pointer = 0x02 , F_l4re_video_goos_dynamic_views = 0x04 , F_l4re_video_goos_dynamic_buffers = 0x08 }

  Flags of information on the goos.

Functions

- int l4re_video_goos_info (l4re_video_goos_t goos, l4re_video_goos_info_t *ginfo) L4_NOTHROW

  Get information on a goos.
- int l4re_video_goos_refresh (l4re_video_goos_t goos, int x, int y, int w, int h) L4_NOTHROW

  Flush a rectangle of pixels of the goos screen.
- int l4re_video_goos_create_buffer (l4re_video_goos_t goos, unsigned long size, l4_cap_idx_t buffer) L4_NOTHROW

  Create a new buffer (memory buffer) for pixel data.
- int l4re_video_goos_delete_buffer (l4re_video_goos_t goos, unsigned idx) L4_NOTHROW

  Delete a pixel buffer.
- int l4re_video_goos_get_static_buffer (l4re_video_goos_t goos, unsigned idx, l4_cap_idx_t buffer) L4_NOTHROW

  Get the data-space capability of the static pixel buffer.
- int l4re_video_goos_create_view (l4re_video_goos_t goos, l4re_video_view_t *view) L4_NOTHROW

  Create a new view (.
- int l4re_video_goos_delete_view (l4re_video_goos_t goos, l4re_video_view_t *view) L4_NOTHROW

  Delete a view.
- int l4re_video_goos_get_view (l4re_video_goos_t goos, unsigned idx, l4re_video_view_t *view) L4_NOTHROW

  Get a view for the given index.

16.169.1 Detailed Description

Note

The C interface of L4Re::Video does *NOT* reflect the full C++ interface on purpose. Use the C++ where possible.

Definition in file goos.h.

16.170 goos.h

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>
#include <l4/re/c/video/view.h>

enum l4re_video_goos_info_flags_t
{
  F_l4re_video_goos_auto_refresh    = 0x01,
  F_l4re_video_goos_pointer         = 0x02,
  F_l4re_video_goos_dynamic_views   = 0x04,
  F_l4re_video_goos_dynamic_buffers = 0x08,
};

typedef struct
{
  unsigned long width;
  unsigned long height;
  unsigned flags;
  unsigned num_static_views;
  unsigned num_static_buffers;
  l4re_video_pixel_info_t pixel_info;
} l4re_video_goos_info_t;

typedef l4_cap_idx_t l4re_video_goos_t;

EXTERN_C_BEGIN

L4_CV int
l4re_video_goos_info(l4re_video_goos_t goos,
                     l4re_video_goos_info_t *ginfo) L4_NOTHROW;

L4_CV int
l4re_video_goos_refresh(l4re_video_goos_t goos, int x, int y, int w,
                        int h) L4_NOTHROW;

L4_CV int
l4re_video_goos_create_buffer(l4re_video_goos_t goos, unsigned long size,
                              l4_cap_idx_t buffer) L4_NOTHROW;

L4_CV int
l4re_video_goos_delete_buffer(l4re_video_goos_t goos, unsigned idx) L4_NOTHROW;

L4_CV int
l4re_video_goos_get_static_buffer(l4re_video_goos_t goos, unsigned idx,
                                  l4_cap_idx_t buffer) L4_NOTHROW;

L4_CV int
l4re_video_goos_create_view(l4re_video_goos_t goos,
                            l4re_video_view_t *view) L4_NOTHROW;

L4_CV int
l4re_video_goos_delete_view(l4re_video_goos_t goos,
                            l4re_video_view_t *view) L4_NOTHROW;


L4_CV int
l4re_video_goos_get_view(l4re_video_goos_t goos, unsigned idx,
                         l4re_video_view_t *view) L4_NOTHROW;

EXTERN_C_END
```

16.171 l4/re/c/video/view.h File Reference

```
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>
```

Include dependency graph for view.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct l4re_video_view_info_t

  View information structure.
- struct l4re_video_view_t

  C representation of a goos view.

Typedefs

- typedef struct l4re_video_view_info_t l4re_video_view_info_t

  View information structure.
- typedef struct l4re_video_view_t l4re_video_view_t

  C representation of a goos view.

Enumerations

- enum l4re_video_view_info_flags_t {
  F_l4re_video_view_none = 0x00 , F_l4re_video_view_set_buffer = 0x01 , F_l4re_video_view_set_buffer_offset = 0x02 , F_l4re_video_view_set_bytes_per_line = 0x04 ,
  F_l4re_video_view_set_pixel = 0x08 , F_l4re_video_view_set_position = 0x10 , F_l4re_video_view_dyn_allocated = 0x20 , F_l4re_video_view_set_background = 0x40 ,
  F_l4re_video_view_set_flags = 0x80 , **F_l4re_video_view_fully_dynamic** , F_l4re_video_view_above = 0x01000 , F_l4re_video_view_flags_mask = 0xff000 }

  Flags of information on a view.

Functions

- int l4re_video_view_refresh (l4re_video_view_t ∗view, int x, int y, int w, int h) L4_NOTHROW

  Flush the given rectangle of pixels of the given view.
- int l4re_video_view_get_info (l4re_video_view_t ∗view, l4re_video_view_info_t ∗info) L4_NOTHROW

  Retrieve information about the given view.
- int l4re_video_view_set_info (l4re_video_view_t ∗view, l4re_video_view_info_t ∗info) L4_NOTHROW

  Set properties of the view.
- int l4re_video_view_set_viewport (l4re_video_view_t ∗view, int x, int y, int w, int h, unsigned long bofs) L4_NOTHROW

  Set the viewport parameters of a view.
- int l4re_video_view_stack (l4re_video_view_t ∗view, l4re_video_view_t ∗pivot, int behind) L4_NOTHROW

  Change the stacking order in the stack of visible views.

16.171.1 Detailed Description

Note

The C interface of L4Re::Video does *NOT* reflect the full C++ interface on purpose. Use the C++ where possible.

Definition in file view.h.

16.172 view.h

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>

enum l4re_video_view_info_flags_t
{
  F_l4re_video_view_none               = 0x00,
  F_l4re_video_view_set_buffer         = 0x01,
  F_l4re_video_view_set_buffer_offset  = 0x02,
  F_l4re_video_view_set_bytes_per_line = 0x04,
  F_l4re_video_view_set_pixel          = 0x08,
  F_l4re_video_view_set_position       = 0x10,
  F_l4re_video_view_dyn_allocated      = 0x20,
  F_l4re_video_view_set_background     = 0x40,
  F_l4re_video_view_set_flags          = 0x80,
  F_l4re_video_view_fully_dynamic      =   F_l4re_video_view_set_buffer
                                         | F_l4re_video_view_set_buffer_offset
                                         | F_l4re_video_view_set_bytes_per_line
                                         | F_l4re_video_view_set_pixel
                                         | F_l4re_video_view_set_position
                                         | F_l4re_video_view_dyn_allocated,

  F_l4re_video_view_above      = 0x01000,
  F_l4re_video_view_flags_mask = 0xff000,
};

typedef struct l4re_video_view_info_t
{
  unsigned                flags;
  unsigned                view_index;
  unsigned long           xpos, ypos, width, height;
  unsigned long           buffer_offset;
  unsigned long           bytes_per_line;
  l4re_video_pixel_info_t pixel_info;
  unsigned                buffer_index;
} l4re_video_view_info_t;


typedef struct l4re_video_view_t
{
  l4_cap_idx_t goos;
  unsigned idx;
} l4re_video_view_t;


EXTERN_C_BEGIN

L4_CV int
l4re_video_view_refresh(l4re_video_view_t *view, int x, int y, int w,
                        int h) L4_NOTHROW;

L4_CV int
l4re_video_view_get_info(l4re_video_view_t *view,
                         l4re_video_view_info_t *info) L4_NOTHROW;

L4_CV int
l4re_video_view_set_info(l4re_video_view_t *view,
                         l4re_video_view_info_t *info) L4_NOTHROW;

L4_CV int
l4re_video_view_set_viewport(l4re_video_view_t *view, int x, int y, int w,
                             int h, unsigned long bofs) L4_NOTHROW;

L4_CV int
l4re_video_view_stack(l4re_video_view_t *view, l4re_video_view_t *pivot,
```

```
                        int behind) L4_NOTHROW;

EXTERN_C_END
```

16.173 l4/re/cap_alloc File Reference

Abstract capability-allocator interface.

```
#include <l4/sys/task>
#include <l4/sys/smart_capability>
#include <l4/re/consts>
#include <l4/cxx/type_traits>
```

Include dependency graph for cap_alloc:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4Re::Cap_alloc

  Capability allocator interface.
- class L4Re::Smart_cap_auto< Unmap_flags >

  Helper for Unique_cap and Unique_del_cap.
- class L4Re::Smart_count_cap< Unmap_flags >

  Helper for Ref_cap and Ref_del_cap.

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.173.1 Detailed Description

Abstract capability-allocator interface.

Definition in file cap_alloc.

16.174 cap_alloc

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *          Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task>
00028 #include <l4/sys/smart_capability>
00029 #include <l4/re/consts>
00030 #include <l4/cxx/type_traits>
00031
00032 namespace L4Re {
00033
00041 class Cap_alloc
00042 {
00043 private:
00044   void operator = (Cap_alloc const &);
00045
00046 protected:
00047   Cap_alloc(Cap_alloc const &) {}
00048   Cap_alloc() {}
00049
00050 public:
00051
00056   virtual L4::Cap<void> alloc() noexcept = 0;
00057   virtual void take(L4::Cap<void> cap) noexcept = 0;
00058
00063   template< typename T >
00064   L4::Cap<T> alloc() noexcept
00065   { return L4::cap_cast<T>(alloc()); }
```

```
00066
00073   virtual void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00074                     unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept = 0;
00075   virtual bool release(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00076                     unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept = 0;
00077
00081   virtual ~Cap_alloc() = 0;
00082
00088   template< typename CAP_ALLOC >
00089   static inline L4Re::Cap_alloc *
00090   get_cap_alloc(CAP_ALLOC &ca)
00091   {
00092     struct CA : public L4Re::Cap_alloc
00093     {
00094       CAP_ALLOC &_ca;
00095       L4::Cap<void> alloc() noexcept override { return _ca.alloc(); }
00096       void take(L4::Cap<void> cap) noexcept override { _ca.take(cap); }
00097
00098       void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00099             unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept override
00100       { _ca.free(cap, task, unmap_flags); }
00101
00102       bool release(L4::Cap<void> cap, l4_cap_idx_t task,
00103                    unsigned unmap_flags) noexcept override
00104       { return _ca.release(cap, task, unmap_flags); }
00105
00106       void operator delete(void *) {}
00107
00108       CA(CAP_ALLOC &ca) : _ca(ca) {}
00109     };
00110
00111     static CA _ca(ca);
00112     return &_ca;
00113   }
00114 };
00115
00116 template<typename ALLOC>
00117 struct Cap_alloc_t : ALLOC, L4Re::Cap_alloc
00118 {
00119   template<typename ...ARGS>
00120   Cap_alloc_t(ARGS &&...args) : ALLOC(cxx::forward<ARGS>(args)...) {}
00121
00122   L4::Cap<void> alloc() noexcept override { return ALLOC::alloc(); }
00123   void take(L4::Cap<void> cap) noexcept override { ALLOC::take(cap); }
00124
00125   void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00126             unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept override
00127   { ALLOC::free(cap, task, unmap_flags); }
00128
00129   bool release(L4::Cap<void> cap, l4_cap_idx_t task,
00130                unsigned unmap_flags) noexcept override
00131   { return ALLOC::release(cap, task, unmap_flags); }
00132
00133   void operator delete(void *) {}
00134 };
00135
00136 inline
00137 Cap_alloc::~Cap_alloc()
00138 {}
00139
00140 extern Cap_alloc *virt_cap_alloc;
00141
00146 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00147 class Smart_cap_auto
00148 {
00149 private:
00150   Cap_alloc *_ca;
00151
00152 public:
00153   Smart_cap_auto() : _ca(0) {}
00154   Smart_cap_auto(Cap_alloc *ca) : _ca(ca) {}
00155
00156   void free(L4::Cap_base &c)
00157   {
00158     if (c.is_valid() && _ca)
00159       _ca->free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00160
00161     invalidate(c);
00162   }
00163
00164   static void invalidate(L4::Cap_base &c)
00165   {
00166     if (c.is_valid())
00167       c.invalidate();
00168   }
00169
00170   static L4::Cap_base copy(L4::Cap_base const &src)
```

```
00171   {
00172     L4::Cap_base r = src;
00173     invalidate(const_cast<L4::Cap_base &>(src));
00174     return r;
00175   }
00176 };
00177
00181 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00182 class Smart_count_cap
00183 {
00184 private:
00185   Cap_alloc *_ca;
00186
00187 public:
00188   Smart_count_cap() : _ca(nullptr) {}
00189   Smart_count_cap(Cap_alloc *ca) : _ca(ca) {}
00194   void free(L4::Cap_base &c) noexcept
00195   {
00196     if (c.is_valid())
00197       {
00198         if (_ca && _ca->release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
00199           c.invalidate();
00200       }
00201   }
00202
00206   static void invalidate(L4::Cap_base &c) noexcept
00207   {
00208     if (c.is_valid())
00209       c.invalidate();
00210   }
00211
00215   L4::Cap_base copy(L4::Cap_base const &src)
00216   {
00217     if (src.is_valid())
00218       _ca->take(L4::Cap<void>(src.cap()));
00219     return src;
00220   }
00221 };
00224 }
```

16.175 l4/re/util/cap_alloc File Reference

Capability allocator.

```
#include <l4/re/util/cap_alloc_impl.h>
#include <l4/sys/smart_capability>
#include <l4/sys/task>
#include <l4/re/consts>
```

Include dependency graph for cap_alloc:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4Re::Util::Smart_cap_auto< Unmap_flags >

  Helper for Unique_cap and Unique_del_cap.
- class L4Re::Util::Smart_count_cap< Unmap_flags >

  Helper for Ref_cap and Ref_del_cap.
- struct L4Re::Util::Ref_cap< T >

  Automatic capability that implements automatic free and unmap of the capability selector.
- struct L4Re::Util::Ref_del_cap< T >

  Automatic capability that implements automatic free and unmap+delete of the capability selector.

Namespaces

- L4Re

  L4Re C++ Interfaces.
- L4Re::Util

  Documentation of the L4 Runtime Environment utility functionality in C++.

Functions

- template<typename T >
  Ref_cap< T >::Cap L4Re::Util::make_ref_cap ()

  Allocate a capability slot and wrap it in a Ref_cap.
- template<typename T >
  Ref_del_cap< T >::Cap L4Re::Util::make_ref_del_cap ()

  Allocate a capability slot and wrap it in a Ref_del_cap.

Variables

- _Cap_alloc & L4Re::Util::cap_alloc

  Capability allocator.

16.175.1 Detailed Description

Capability allocator.

Definition in file cap_alloc.

16.176 cap_alloc

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025
00026 #pragma once
00027
00028 #include <l4/re/util/cap_alloc_impl.h>
00029 #include <l4/sys/smart_capability>
00030 #include <l4/sys/task>
00031 #include <l4/re/consts>
00032
00033 namespace L4Re { namespace Util {
00034
00053 extern _Cap_alloc &cap_alloc;
00054
00058 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00059 class Smart_cap_auto
00060 {
00061 public:
00065   static void free(L4::Cap_base &c)
00066   {
00067     if (c.is_valid())
00068       {
00069         cap_alloc.free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00070         c.invalidate();
00071       }
00072   }
00073
00077   static void invalidate(L4::Cap_base &c)
```

```
00078   {
00079     if (c.is_valid())
00080       c.invalidate();
00081   }
00082
00086   static L4::Cap_base copy(L4::Cap_base const &src)
00087   {
00088     L4::Cap_base r = src;
00089     invalidate(const_cast<L4::Cap_base &>(src));
00090     return r;
00091   }
00092 };
00093
00094
00098 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00099 class Smart_count_cap
00100 {
00101 public:
00106   static void free(L4::Cap_base &c) noexcept
00107   {
00108     if (c.is_valid())
00109       {
00110         if (cap_alloc.release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
00111     c.invalidate();
00112       }
00113   }
00114
00118   static void invalidate(L4::Cap_base &c) noexcept
00119   {
00120     if (c.is_valid())
00121       c.invalidate();
00122   }
00123
00127   static L4::Cap_base copy(L4::Cap_base const &src)
00128   {
00129     cap_alloc.take(L4::Cap<void>(src.cap()));
00130     return src;
00131   }
00132 };
00133
00134
00164 template< typename T >
00165 struct Ref_cap
00166 {
00167   typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_ALL_SPACES> > Cap;
00168 };
00169
00205 template< typename T >
00206 struct Ref_del_cap
00207 {
00208   typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_DELETE_OBJ> > Cap;
00209 };
00210
00216 template< typename T >
00217 typename Ref_cap<T>::Cap
00218 make_ref_cap() { return typename Ref_cap<T>::Cap(cap_alloc.alloc<T>()); }
00219
00225 template< typename T >
00226 typename Ref_del_cap<T>::Cap
00227 make_ref_del_cap()
00228 { return typename Ref_del_cap<T>::Cap(cap_alloc.alloc<T>()); }
00229
00232 }}
00233
```

16.177 l4/re/dataspace File Reference

Dataspace interface.

```
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
```

```
#include <l4/sys/cxx/types>
```

Include dependency graph for dataspace:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4Re::Dataspace

  Interface for memory-like objects.
- struct L4Re::Dataspace::F

  Dataspace flags definitions.
- struct L4Re::Dataspace::Stats

  Information about the dataspace.

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.177.1 Detailed Description

Dataspace interface.

Definition in file dataspace.

16.178 dataspace

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU General Public License 2.
00016  * Please see the COPYING-GPL-2 file for details.
00017  *
00018  * As a special exception, you may use this file as part of a free software
00019  * library without restriction.  Specifically, if other files instantiate
00020  * templates or use macros or inline functions from this file, or you compile
00021  * this file and link it with other files to produce an executable, this
00022  * file does not by itself cause the resulting executable to be covered by
00023  * the GNU General Public License.  This exception does not however
00024  * invalidate any other reasons why the executable file might be covered by
00025  * the GNU General Public License.
00026  */
00027
00028 #pragma once
00029
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/l4int.h>
00032 #include <l4/sys/capability>
00033 #include <l4/re/protocols.h>
00034 #include <l4/sys/cxx/ipc_types>
00035 #include <l4/sys/cxx/ipc_iface>
00036 #include <l4/sys/cxx/types>
00037
00038 namespace L4Re
00039 {
00040
00041   // MISSING:
00042   // * size support in map, mapped size in reply
00043
00060 class L4_EXPORT Dataspace :
00061   public L4::Kobject_t<Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE,
00062                        L4::Type_info::Demand_t<1> >
00063 {
00064 public:
00065
00067   struct F
00068   {
00069     enum
00070     {
00071       Caching_shift = 4,
00072     };
00073
00077     enum Flags
00078     {
00080       R   = L4_FPAGE_RO,
00082       Ro  = L4_FPAGE_RO,
00084       RW  = L4_FPAGE_RW,
00086       W   = L4_FPAGE_W,
00087       X   = L4_FPAGE_X,
00088       RX  = L4_FPAGE_RX,
00089       RWX = L4_FPAGE_RWX,
00091       Rights_mask = 0x0f,
00092
00094       Normal        = 0x00,
00096       Cacheable     = Normal,
00098       Bufferable    = 0x10,
00100       Uncacheable   = 0x20,
00102       Caching_mask  = 0x30,
00103     };
00104
00105     L4_TYPES_FLAGS_OPS_DEF(Flags);
00106   };
00107
00108   struct Flags : L4::Types::Flags_ops_t<Flags>
00109   {
00110     unsigned long raw;
00111     Flags() = default;
00112     explicit constexpr Flags(unsigned long f) : raw(f) {}
00113     constexpr Flags(F::Flags f) : raw(f) {}
00114     constexpr bool r() const { return raw & L4_FPAGE_RO; }
00115     constexpr bool w() const { return raw & L4_FPAGE_W; }
00116     constexpr bool x() const { return raw & L4_FPAGE_X; }
00117
00118     constexpr unsigned long fpage_rights() const
00119     { return raw & 0xf; }
```

```
  };

  typedef l4_uint64_t Size;
  typedef l4_uint64_t Offset;
  typedef l4_uint64_t Map_addr;

  struct Stats
  {
    Size size;
    Flags flags;
  };


  long map(Offset offset, Flags flags, Map_addr local_addr,
           Map_addr min_addr, Map_addr max_addr) const noexcept;

  long map_region(Offset offset, Flags flags,
                  Map_addr min_addr, Map_addr max_addr) const noexcept;

  L4_RPC(long, clear, (Offset offset, Size size));

  L4_RPC(long, allocate, (Offset offset, Size size));

  L4_RPC(long, copy_in, (Offset dst_offs, L4::Ipc::Cap<Dataspace> src,
                         Offset src_offs, Size size));

  Size size() const noexcept;

  Flags flags() const noexcept;

  L4_RPC(long, info, (Stats *stats));

  L4_RPC_NF(long, map, (Offset offset, Map_addr spot,
                        Flags flags, L4::Ipc::Rcv_fpage r,
                        L4::Ipc::Snd_fpage &fp));

private:

  long __map(Offset offset, unsigned char *order, Flags flags,
             Map_addr local_addr) const noexcept;

public:
  typedef L4::Typeid::Rpcs<map_t, clear_t, info_t, copy_in_t,
                           allocate_t> Rpcs;

};

}
```

16.179 l4/re/dataspace-sys.h File Reference

Dataspace protocol defintion.

This graph shows which files directly or indirectly include this file:

Namespaces

- L4Re

  L4Re C++ Interfaces.

Enumerations

- enum L4Re::Dataspace_::Opcodes

  Data-space communication-protocol opcodes.

16.179.1 Detailed Description

Dataspace protocol defintion.

Definition in file dataspace-sys.h.

16.180 dataspace-sys.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027   namespace Dataspace_
00028   {
00034     enum Opcodes { Map, Clear, Stats, Copy, Take, Release, Allocate };
00035   };
00036 };
00037
```

16.181 l4/re/debug File Reference

Debug interface.

```
#include <l4/sys/capability>
#include <l4/re/protocols.h>
```

`#include <l4/sys/cxx/ipc_iface>`
Include dependency graph for debug:

Data Structures

- class L4Re::Debug_obj

  Debug interface.

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.181.1 Detailed Description

Debug interface.

Definition in file debug.

16.182 debug

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/capability>
00027 #include <l4/re/protocols.h>
00028 #include <l4/sys/cxx/ipc_iface>
00029
00030 namespace L4Re {
00051 class L4_EXPORT Debug_obj :
00052   public L4::Kobject_t<Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG>
00053 {
00054 public:
00055
00067   L4_INLINE_RPC(long, debug, (unsigned long function));
00068   typedef L4::Typeid::Rpc_nocode<debug_t> Rpcs;
00069 };
00070
00071 template<typename BASE>
00072 class Debug_obj_t :
00073   public L4::Kobject_2t<Debug_obj_t<BASE>, BASE, Debug_obj, L4::PROTO_EMPTY>
00074 {
00075   typedef L4::Typeid::Rpcs<> Rpcs;
00076 };
00077 }
```

16.183 l4/re/dma_space File Reference

```
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/dataspace>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/types>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for dma_space:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4Re::Dma_space

  DMA Address Space.

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.184 dma_space

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00006 /*
00007  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/sys/l4int.h>
00027 #include <l4/sys/capability>
00028 #include <l4/re/dataspace>
00029 #include <l4/re/protocols.h>
00030 #include <l4/sys/cxx/types>
00031 #include <l4/sys/cxx/ipc_types>
00032 #include <l4/sys/cxx/ipc_iface>
00033
00034 namespace L4Re
00035 {
00036
00057 class Dma_space :
00058   public L4::Kobject_0t< Dma_space,
00059                          L4RE_PROTO_DMA_SPACE,
00060                          L4::Type_info::Demand_t<1> >
00061 {
00062 public:
00064   typedef l4_uint64_t Dma_addr;
00065
00069   enum Direction
00070   {
00071     Bidirectional,
00072     To_device,
00073     From_device,
00074     None
00075   };
00076
00081   enum Attribute
00082   {
00094     No_sync
00095   };
00096
00102   typedef L4::Types::Flags<Attribute> Attributes;
00103
00109   enum Space_attrib
00110   {
00117     Coherent,
00118
00123     Phys_space
00124   };
00125
00127   typedef L4::Types::Flags<Space_attrib> Space_attribs;
00128
00153   L4_INLINE_RPC(
00154       long, map, (L4::Ipc::Cap<L4Re::Dataspace> src,
00155                   L4Re::Dataspace::Offset offset,
00156                   L4::Ipc::In_out<l4_size_t *> size,
00157                   Attributes attrs, Direction dir,
00158                   Dma_addr *dma_addr));
00159
00168   L4_INLINE_RPC(
00169       long, unmap, (Dma_addr dma_addr,
00170                     l4_size_t size, Attributes attrs, Direction dir));
00171
00185   L4_INLINE_RPC(
00186       long, associate, (L4::Ipc::Opt<L4::Ipc::Cap<L4::Task> > dma_task,
00187                         Space_attribs attr),
00188       L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00189
00195   L4_INLINE_RPC(
00196       long, disassociate, (),
00197       L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00198
```

```
  typedef L4::Typeid::Rpcs<map_t, unmap_t, associate_t, disassociate_t> Rpcs;
};

}
```

16.185 l4/re/elf_aux.h File Reference

Auxiliary information for binaries.

`#include <l4/sys/types.h>`
Include dependency graph for elf_aux.h:

Data Structures

- struct l4re_elf_aux_t

  Generic header for each auxiliary vector element.
- struct l4re_elf_aux_vma_t

  Auxiliary vector element for a reserved virtual memory area.
- struct l4re_elf_aux_mword_t

  Auxiliary vector element for a single unsigned data word.

Macros

- #define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".rol4re_elf_aux"), aligned(sizeof(l4_umword_t))))

  Define an auxiliary vector element.
- #define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...) static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}

  Define an auxiliary vector element.

Typedefs

- typedef struct l4re_elf_aux_t l4re_elf_aux_t

  Generic header for each auxiliary vector element.
- typedef struct l4re_elf_aux_vma_t l4re_elf_aux_vma_t

  Auxiliary vector element for a reserved virtual memory area.
- typedef struct l4re_elf_aux_mword_t l4re_elf_aux_mword_t

  Auxiliary vector element for a single unsigned data word.

Enumerations

- enum { L4RE_ELF_AUX_T_NONE = 0 , L4RE_ELF_AUX_T_VMA , L4RE_ELF_AUX_T_STACK_SIZE , L4RE_ELF_AUX_T_STACK_ADDR , L4RE_ELF_AUX_T_KIP_ADDR }

16.185.1 Detailed Description

Auxiliary information for binaries.

Definition in file elf_aux.h.

16.186 elf_aux.h

```
00001
00005 /*
00006  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <l4/sys/types.h>
00025
00026
00052 #define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".rol4re_elf_aux"),
       aligned(sizeof(l4_umword_t))))
00053
00067 #define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...) \
00068   static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}
00069
00070 enum
00071 {
00075   L4RE_ELF_AUX_T_NONE = 0,
00076
00080   L4RE_ELF_AUX_T_VMA,
00081
00086   L4RE_ELF_AUX_T_STACK_SIZE,
00087
00092   L4RE_ELF_AUX_T_STACK_ADDR,
00093
00098   L4RE_ELF_AUX_T_KIP_ADDR,
00099 };
00100
00104 typedef struct l4re_elf_aux_t
```

```
{
  l4_umword_t type;
  l4_umword_t length;
} l4re_elf_aux_t;

typedef struct l4re_elf_aux_vma_t
{
  l4_umword_t type;
  l4_umword_t length;
  l4_umword_t start;
  l4_umword_t end;
} l4re_elf_aux_vma_t;

typedef struct l4re_elf_aux_mword_t
{
  l4_umword_t type;
  l4_umword_t length;
  l4_umword_t value;
} l4re_elf_aux_mword_t;
```

16.187 l4/re/env File Reference

Environment interface.

```
#include <l4/sys/types.h>
#include <l4/re/rm>
#include <l4/re/parent>
#include <l4/re/mem_alloc>
#include <l4/re/log>
#include <l4/re/consts>
#include <l4/re/env.h>
```

Include dependency graph for env:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4Re::Env

  C++ interface of the initial environment that is provided to an L4 task.

Namespaces

- L4

  L4 low-level kernel interface.
- L4Re

  L4Re C++ Interfaces.

16.187.1 Detailed Description

Environment interface.

Definition in file env.

16.188 env

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *               Björn Döbel <doebel@os.inf.tu-dresden.de>
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction.  Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License.  This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027
00028 #include <l4/sys/types.h>
00029
00030 #include <l4/re/rm>
00031 #include <l4/re/parent>
00032 #include <l4/re/mem_alloc>
00033 #include <l4/re/log>
00034 #include <l4/re/consts>
00035
00036 #include <l4/re/env.h>
00037
00038 namespace L4 {
00039 class Scheduler;
00040 }
00041
00045 namespace L4Re
00046 {
00085   class L4_EXPORT Env
00086   {
00087   private:
00088     l4re_env_t _env;
00089   public:
00090
00094     typedef l4re_env_cap_entry_t Cap_entry;
00095
00103     static Env const *env() noexcept
00104     { return reinterpret_cast<Env*>(l4re_global_env); }
```

```
    L4::Cap<Parent> parent() const noexcept
    { return L4::Cap<Parent>(_env.parent); }
    L4::Cap<Mem_alloc> mem_alloc() const noexcept
    { return L4::Cap<Mem_alloc>(_env.mem_alloc); }
    L4::Cap<L4::Factory> user_factory() const noexcept
    { return L4::Cap<L4::Factory>(_env.mem_alloc); }
    L4::Cap<Rm> rm() const noexcept
    { return L4::Cap<Rm>(_env.rm); }
    L4::Cap<Log> log() const noexcept
    { return L4::Cap<Log>(_env.log); }
    L4::Cap<L4::Thread> main_thread() const noexcept
    { return L4::Cap<L4::Thread>(_env.main_thread); }
    L4::Cap<L4::Task> task() const noexcept
    { return L4::Cap<L4::Task>(L4RE_THIS_TASK_CAP); }
    L4::Cap<L4::Factory> factory() const noexcept
    { return L4::Cap<L4::Factory>(_env.factory); }
    l4_cap_idx_t first_free_cap() const noexcept
    { return _env.first_free_cap; }
    l4_fpage_t utcb_area() const noexcept
    { return _env.utcb_area; }
    l4_addr_t first_free_utcb() const noexcept
    { return _env.first_free_utcb; }

    Cap_entry const *initial_caps() const noexcept
    { return _env.caps; }

    Cap_entry const *get(char const *name, unsigned l) const noexcept
    { return l4re_env_get_cap_l(name, l, &_env); }

    template< typename T >
    L4::Cap<T> get_cap(char const *name, unsigned l) const noexcept
    {
      if (Cap_entry const *e = get(name, l))
  return L4::Cap<T>(e->cap);

      return L4::Cap<T>(-L4_ENOENT);
    }

    template< typename T >
    L4::Cap<T> get_cap(char const *name) const noexcept
    { return get_cap<T>(name, __builtin_strlen(name)); }

    void parent(L4::Cap<Parent> const &c) noexcept
    { _env.parent = c.cap(); }
    void mem_alloc(L4::Cap<Mem_alloc> const &c) noexcept
    { _env.mem_alloc = c.cap(); }
    void rm(L4::Cap<Rm> const &c) noexcept
    { _env.rm = c.cap(); }
    void log(L4::Cap<Log> const &c) noexcept
    { _env.log = c.cap(); }
    void main_thread(L4::Cap<L4::Thread> const &c) noexcept
    { _env.main_thread = c.cap(); }
    void factory(L4::Cap<L4::Factory> const &c) noexcept
    { _env.factory = c.cap(); }
    void first_free_cap(l4_cap_idx_t c) noexcept
    { _env.first_free_cap = c; }
    void utcb_area(l4_fpage_t utcbs) noexcept
    { _env.utcb_area = utcbs; }
    void first_free_utcb(l4_addr_t u) noexcept
    { _env.first_free_utcb = u; }

    L4::Cap<L4::Scheduler> scheduler() const noexcept
    { return L4::Cap<L4::Scheduler>(_env.scheduler); }

    void scheduler(L4::Cap<L4::Scheduler> const &c) noexcept
    { _env.scheduler = c.cap(); }

    void initial_caps(Cap_entry *first) noexcept
    { _env.caps = first; }
  };
};
```

16.189 l4/re/env.h File Reference

Environment interface.

```
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
```

```
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>
#include <l4/re/consts.h>
```

Include dependency graph for env.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct l4re_env_cap_entry_t

  Entry in the L4Re environment array for the named inital objects.
- struct l4re_env_t

  Initial environment data structure.

Typedefs

- typedef struct l4re_env_cap_entry_t l4re_env_cap_entry_t

  Entry in the L4Re environment array for the named inital objects.
- typedef struct l4re_env_t l4re_env_t

  Initial environment data structure.

Functions

- l4re_env_t * l4re_env (void) L4_NOTHROW

  Get L4Re initial environment.
- l4_kernel_info_t * l4re_kip (void) L4_NOTHROW

  Get Kernel Info Page.
- l4_cap_idx_t l4re_env_get_cap (char const *name) L4_NOTHROW

  Get the capability selector for the object named name.
- l4_cap_idx_t l4re_env_get_cap_e (char const *name, l4re_env_t const *e) L4_NOTHROW

  Get the capability selector for the object named name.
- l4re_env_cap_entry_t const * l4re_env_get_cap_l (char const *name, unsigned l, l4re_env_t const *e) L4_NOTHROW

  Get the full l4re_env_cap_entry_t for the object named name.

16.189.1 Detailed Description

Environment interface.

Definition in file env.h.

16.189.2 Typedef Documentation

16.189.2.1 l4re_env_t

```
typedef struct l4re_env_t l4re_env_t
```

Initial environment data structure.

See also

Initial environment

16.190 env.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *              Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
```

```
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>

#include <l4/re/consts.h>

typedef struct l4re_env_cap_entry_t
{
  l4_cap_idx_t cap;

  l4_umword_t flags;

  char name[16];
#ifdef __cplusplus

  l4re_env_cap_entry_t() L4_NOTHROW : cap(L4_INVALID_CAP), flags(~0) {}

  l4re_env_cap_entry_t(char const *n, l4_cap_idx_t c, l4_umword_t f = 0) L4_NOTHROW
  : cap(c), flags(f)
  {
    for (unsigned i = 0; n && i < sizeof(name); ++i, ++n)
      {
        name[i] = *n;
  if (!*n)
    break;
      }
  }

  static bool is_valid_name(char const *n) L4_NOTHROW
  {
    for (unsigned i = 0; *n; ++i, ++n)
      if (i > sizeof(name))
        return false;

    return true;
  }
#endif
} l4re_env_cap_entry_t;


typedef struct l4re_env_t
{
  l4_cap_idx_t parent;
  l4_cap_idx_t rm;
  l4_cap_idx_t mem_alloc;
  l4_cap_idx_t log;
  l4_cap_idx_t main_thread;
  l4_cap_idx_t factory;
  l4_cap_idx_t scheduler;
  l4_cap_idx_t first_free_cap;
  l4_fpage_t utcb_area;
  l4_addr_t first_free_utcb;
  l4re_env_cap_entry_t *caps;
} l4re_env_t;

extern l4re_env_t *l4re_global_env;


L4_INLINE l4re_env_t *l4re_env(void) L4_NOTHROW;

/*
 * FIXME: this seems to be at the wrong place here
 */
L4_INLINE l4_kernel_info_t *l4re_kip(void) L4_NOTHROW;


L4_INLINE l4_cap_idx_t
l4re_env_get_cap(char const *name) L4_NOTHROW;

L4_INLINE l4_cap_idx_t
l4re_env_get_cap_e(char const *name, l4re_env_t const *e) L4_NOTHROW;

L4_INLINE l4re_env_cap_entry_t const *
l4re_env_get_cap_l(char const *name, unsigned l, l4re_env_t const *e) L4_NOTHROW;

L4_INLINE
l4re_env_t *l4re_env() L4_NOTHROW
{ return l4re_global_env; }

L4_INLINE
l4_kernel_info_t *l4re_kip() L4_NOTHROW
{
  extern char __L4_KIP_ADDR__[];
  return (l4_kernel_info_t *)__L4_KIP_ADDR__;
}
```

```
L4_INLINE l4re_env_cap_entry_t const *
l4re_env_get_cap_l(char const *name, unsigned l, l4re_env_t const *e) L4_NOTHROW
{
  l4re_env_cap_entry_t const *c = e->caps;
  for (; c && c->flags != ~0UL; ++c)
    {
      unsigned i;
      for (i = 0;
           i < sizeof(c->name) && i < l && c->name[i] && name[i] && name[i] == c->name[i];
           ++i)
  ;

      if (i == l && (i == sizeof(c->name) || !c->name[i]))
  return c;
    }
  return NULL;
}

L4_INLINE l4_cap_idx_t
l4re_env_get_cap_e(char const *name, l4re_env_t const *e) L4_NOTHROW
{
  unsigned l;
  l4re_env_cap_entry_t const *r;
  for (l = 0; name[l]; ++l) ;
  r = l4re_env_get_cap_l(name, l, e);
  if (r)
    return r->cap;

  return L4_INVALID_CAP;
}

L4_INLINE l4_cap_idx_t
l4re_env_get_cap(char const *name) L4_NOTHROW
{ return l4re_env_get_cap_e(name, l4re_env()); }

```

16.191 l4/re/error_helper File Reference

Error helper.

```
#include <l4/sys/types.h>
#include <l4/cxx/exceptions>
#include <l4/cxx/type_traits>
#include <l4/sys/err.h>
```

Include dependency graph for error_helper:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4Re

  L4Re C++ Interfaces.

Functions

- void L4Re::throw_error (long err, char const ∗extra="")

  Generate C++ exception.
- long L4Re::chksys (long err, char const ∗extra="", long ret=0)

  Generate C++ exception on error.
- long L4Re::chksys (l4_msgtag_t const &t, char const ∗extra="", l4_utcb_t ∗utcb=l4_utcb(), long ret=0)

  Generate C++ exception on error.
- long L4Re::chksys (l4_msgtag_t const &t, l4_utcb_t ∗utcb, char const ∗extra="")

  Generate C++ exception on error.
- template<typename T >
  T L4Re::chkcap (T &&cap, char const ∗extra="", long err=-L4_ENOMEM)

  Check for valid capability or raise C++ exception.
- l4_msgtag_t L4Re::chkipc (l4_msgtag_t tag, char const ∗extra="", l4_utcb_t ∗utcb=l4_utcb())

  Test a message tag for IPC errors.

16.191.1 Detailed Description

Error helper.

Definition in file error_helper.

16.192 error_helper

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/cxx/exceptions>
00029 #include <l4/cxx/type_traits>
00030 #include <l4/sys/err.h>
00031
00032 namespace L4Re {
00033
00034 #ifdef __EXCEPTIONS
00035
00045 [[noreturn]] inline void throw_error(long err, char const *extra = "")
00046 {
00047   switch (err)
00048     {
00049     case -L4_ENOENT: throw (L4::Element_not_found(extra));
00050     case -L4_ENOMEM: throw (L4::Out_of_memory(extra));
00051     case -L4_EEXIST: throw (L4::Element_already_exists(extra));
00052     case -L4_ERANGE: throw (L4::Bounds_error(extra));
00053     default: throw (L4::Runtime_error(err, extra));
00054     }
00055 }
00056
00067 inline
00068 long chksys(long err, char const *extra = "", long ret = 0)
00069 {
00070   if (L4_UNLIKELY(err < 0))
00071     throw_error(ret ? ret : err, extra);
00072
00073   return err;
00074 }
00075
00088 inline
00089 long chksys(l4_msgtag_t const &t, char const *extra = "",
00090             l4_utcb_t *utcb = l4_utcb(), long ret = 0)
00091 {
00092   if (L4_UNLIKELY(t.has_error()))
00093     throw_error(ret ? ret : l4_error_u(t, utcb), extra);
00094   else if (L4_UNLIKELY(t.label() < 0))
00095     throw_error(ret ? ret: t.label(), extra);
00096
00097   return t.label();
00098 }
00099
00111 inline
00112 long chksys(l4_msgtag_t const &t, l4_utcb_t *utcb, char const *extra = "")
00113 { return chksys(t, extra, utcb); }
00114
00115 #if 0
00116 inline
00117 long chksys(long ret, long err, char const *extra = "")
00118 {
00119   if (L4_UNLIKELY(ret < 0))
00120     throw_error(err, extra);
00121
00122   return ret;
00123 }
00124 #endif
00125
00142 template<typename T>
00143 inline
00144 #if __cplusplus >= 201103L
00145 T chkcap(T &&cap, char const *extra = "", long err = -L4_ENOMEM)
00146 #else
00147 T chkcap(T cap, char const *extra = "", long err = -L4_ENOMEM)
```

```
#endif
{
  if (L4_UNLIKELY(!cap.is_valid()))
    throw_error(err ? err : cap.cap(), extra);

#if __cplusplus >= 201103L
  return cxx::forward<T>(cap);
#else
  return cap;
#endif
}

inline
l4_msgtag_t
chkipc(l4_msgtag_t tag, char const *extra = "",
       l4_utcb_t *utcb = l4_utcb())
{
  if (L4_UNLIKELY(tag.has_error()))
    chksys(l4_error_u(tag, utcb), extra);

  return tag;
}
#endif

}
```

16.193 l4/re/util/event File Reference

```
#include <l4/re/cap_alloc>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/unique_cap>
#include <l4/re/env>
#include <l4/re/rm>
#include <l4/re/util/event_buffer>
#include <l4/sys/factory>
#include <l4/cxx/type_traits>
```

Include dependency graph for event:

Data Structures

- class L4Re::Util::Event_t< PAYLOAD >

  Convenience wrapper for getting access to an event object.

Namespaces

- L4Re

  L4Re C++ Interfaces.
- L4Re::Util

  Documentation of the L4 Runtime Environment utility functionality in C++.

16.194 event

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/re/cap_alloc>
00026 #include <l4/re/util/cap_alloc>
00027 #include <l4/re/util/unique_cap>
00028 #include <l4/re/env>
00029 #include <l4/re/rm>
00030 #include <l4/re/util/event_buffer>
00031 #include <l4/sys/factory>
00032 #include <l4/cxx/type_traits>
00033
00034 namespace L4Re { namespace Util {
00035
00042 template< typename PAYLOAD >
00043 class Event_t
00044 {
00045 public:
00049   enum Mode
00050   {
00051     Mode_irq,
00052     Mode_polling,
00053   };
00054
00069   template<typename IRQ_TYPE>
00070   int init(L4::Cap<L4Re::Event> event,
00071            L4Re::Env const *env = L4Re::Env::env(),
00072            L4Re::Cap_alloc *ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))
00073   {
00074     Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<L4Re::Dataspace>());
00075     if (!ev_ds.is_valid())
00076       return -L4_ENOMEM;
00077
00078     int r;
00079
00080     Unique_del_cap<IRQ_TYPE> ev_irq(ca->alloc<IRQ_TYPE>());
00081     if (!ev_irq.is_valid())
00082       return -L4_ENOMEM;
00083
00084     if ((r = l4_error(env->factory()->create(ev_irq.get()))))
00085       return r;
00086
00087     if ((r = l4_error(event->bind(0, ev_irq.get()))))
00088       return r;
00089
00090     if ((r = event->get_buffer(ev_ds.get())))
00091       return r;
00092
00093     long sz = ev_ds->size();
00094     if (sz < 0)
00095       return sz;
00096
00097     Rm::Unique_region<void*> buf;
```

```
00098
00099     if ((r = env->rm()->attach(&buf, sz,
00100                                L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00101                                L4::Ipc::make_cap_rw(ev_ds.get()))))
00102       return r;
00103
00104     _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
00105     _ev_ds     = cxx::move(ev_ds);
00106     _ev_irq    = cxx::move(ev_irq);
00107     _buf       = cxx::move(buf);
00108
00109     return 0;
00110   }
00111
00123   int init_poll(L4::Cap<L4Re::Event> event,
00124                 L4Re::Env const *env = L4Re::Env::env(),
00125                 L4Re::Cap_alloc *ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))
00126   {
00127     Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<L4Re::Dataspace>());
00128     if (!ev_ds.is_valid())
00129       return -L4_ENOMEM;
00130
00131     int r;
00132
00133     if ((r = event->get_buffer(ev_ds.get())))
00134       return r;
00135
00136     long sz = ev_ds->size();
00137     if (sz < 0)
00138       return sz;
00139
00140     Rm::Unique_region<void*> buf;
00141
00142     if ((r = env->rm()->attach(&buf, sz,
00143                                L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00144                                L4::Ipc::make_cap_rw(ev_ds.get()))))
00145       return r;
00146
00147     _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
00148     _ev_ds     = cxx::move(ev_ds);
00149     _buf       = cxx::move(buf);
00150
00151     return 0;
00152   }
00153
00159   L4Re::Event_buffer_t<PAYLOAD> &buffer() { return _ev_buffer; }
00160
00166   L4::Cap<L4::Triggerable> irq() const { return _ev_irq.get(); }
00167
00168 private:
00169   Unique_cap<L4Re::Dataspace> _ev_ds;
00170   Unique_del_cap<L4::Triggerable> _ev_irq;
00171   L4Re::Event_buffer_t<PAYLOAD> _ev_buffer;
00172   Rm::Unique_region<void*> _buf;
00173 };
00174
00175 typedef Event_t<Default_event_payload> Event;
00176
00177 }}
```

16.195 l4/re/impl/dataspace_impl.h File Reference

Dataspace client stub implementation.

```
#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/cxx/consts>
```

Include dependency graph for dataspace_impl.h:

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.195.1 Detailed Description

Dataspace client stub implementation.

Definition in file dataspace_impl.h.

16.196 dataspace_impl.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
```

```
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/cxx/consts>

L4_RPC_DEF(L4Re::Dataspace::clear);
L4_RPC_DEF(L4Re::Dataspace::allocate);
L4_RPC_DEF(L4Re::Dataspace::copy_in);
L4_RPC_DEF(L4Re::Dataspace::info);

namespace L4Re {


long
Dataspace::__map(Dataspace::Offset offset, unsigned char *size,
                 Dataspace::Flags flags,
                 Dataspace::Map_addr local_addr) const noexcept
{
  Map_addr spot = local_addr & ~(~0ULL << l4_umword_t(*size));
  Map_addr base = local_addr & (~0ULL << l4_umword_t(*size));
  L4::Ipc::Rcv_fpage r;
  r = L4::Ipc::Rcv_fpage::mem(base, *size, 0);

  L4::Ipc::Snd_fpage fp;
  long err = map_t::call(c(), offset, spot, flags, r, fp, l4_utcb());
  if (L4_UNLIKELY(err < 0))
    return err;

  *size = fp.rcv_order();
  return err;
}

long
Dataspace::map_region(Dataspace::Offset offset, Dataspace::Flags flags,
                      Dataspace::Map_addr min_addr,
                      Dataspace::Map_addr max_addr) const noexcept
{
  min_addr   = L4::trunc_page(min_addr);
  max_addr   = L4::round_page(max_addr);
  unsigned char order = L4_LOG2_PAGESIZE;

  long err = 0;

  while (min_addr < max_addr)
    {
      unsigned char order_mapped;
      order_mapped = order
        = L4::max_order(order, min_addr, min_addr, max_addr, min_addr);

      err = __map(offset, &order_mapped, flags, min_addr);
      if (L4_UNLIKELY(err < 0))
        return err;

      if (order > order_mapped)
        order = order_mapped;

      min_addr += Map_addr(1) << order;
      offset   += Map_addr(1) << order;

      if (min_addr >= max_addr)
        return 0;

      while (min_addr != L4::trunc_order(min_addr, order)
             || max_addr < L4::round_order(min_addr + 1, order))
        --order;
    }

  return 0;
}


long
Dataspace::map(Dataspace::Offset offset, Dataspace::Flags flags,
               Dataspace::Map_addr local_addr,
               Dataspace::Map_addr min_addr,
               Dataspace::Map_addr max_addr) const noexcept
{
  min_addr   = L4::trunc_page(min_addr);
  max_addr   = L4::round_page(max_addr);
  local_addr = L4::trunc_page(local_addr);
  unsigned char order
    = L4::max_order(L4_LOG2_PAGESIZE, local_addr, min_addr, max_addr, local_addr);
```

```
  return __map(offset, &order, flags, local_addr);
}

Dataspace::Size
Dataspace::size() const noexcept
{
  Stats stats = Stats();
  int err = info(&stats);
  if (err < 0)
    return 0;
  return stats.size;
}

Dataspace::Flags
Dataspace::flags() const noexcept
{
  Stats stats = Stats();
  int err = info(&stats);
  if (err < 0)
    return Flags(0);
  return stats.flags;
}

};
```

16.197 l4/re/impl/mem_alloc_impl.h File Reference

Memory allocator client stub implementation.

```
#include <l4/re/mem_alloc>
#include <l4/re/mem_alloc-sys.h>
#include <l4/re/dataspace>
#include <l4/re/error_helper>
#include <l4/sys/factory>
```

Include dependency graph for mem_alloc_impl.h:

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.197.1 Detailed Description

Memory allocator client stub implementation.

Definition in file mem_alloc_impl.h.

16.198 mem_alloc_impl.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *              Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #include <l4/re/mem_alloc>
00024 #include <l4/re/mem_alloc-sys.h>
00025 #include <l4/re/dataspace>
00026 #include <l4/re/error_helper>
00027
00028 #include <l4/sys/factory>
00029
00030
00031 namespace L4Re
00032 {
00033
00034 long
00035 Mem_alloc::alloc(long size,
00036                  L4::Cap<Dataspace> mem, unsigned long flags,
00037                  unsigned long align) const noexcept
00038 {
00039   L4::Cap<L4::Factory> f(cap());
00040   return l4_error(f->create(mem, L4Re::Dataspace::Protocol)
00041                   « l4_mword_t(size)
00042                   « l4_umword_t(flags)
00043                   « l4_umword_t(align));
00044 }
00045
00046 };
```

16.199 l4/re/impl/namespace_impl.h File Reference

Namespace client stub implementation.

```
#include <l4/re/namespace>
#include <l4/util/util.h>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/assert.h>
```

`#include <cstring>`

Include dependency graph for namespace_impl.h:

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.199.1 Detailed Description

Namespace client stub implementation.

Definition in file namespace_impl.h.

16.200 namespace_impl.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *              Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
```

```
#include <l4/re/namespace>

#include <l4/util/util.h>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/assert.h>

#include <cstring>

L4_RPC_DEF(L4Re::Namespace::query);
L4_RPC_DEF(L4Re::Namespace::register_obj);
L4_RPC_DEF(L4Re::Namespace::unlink);

namespace L4Re {

long
Namespace::_query(char const *name, unsigned len,
                  L4::Cap<void> const &target,
                  l4_umword_t *local_id, bool iterate) const noexcept
{
  l4_assert(target.is_valid());

  L4::Cap<Namespace> ns = c();
  L4::Ipc::Array<char const, unsigned long> _name(len, name);

  while (_name.length > 0)
    {
      L4::Ipc::Snd_fpage cap;
      L4::Opcode dummy;
      int err = query_t::call(ns, _name,
                              L4::Ipc::Small_buf(target.cap(),
                                                 local_id
                                                   ? L4_RCV_ITEM_LOCAL_ID
                                                   : 0),
                              cap, dummy, _name);
      if (err < 0)
        return err;

      bool const partly = err & Partly_resolved;
      if (cap.id_received())
        {
          *local_id = cap.data();
          return _name.length;
        }

      if (partly && iterate)
        ns = L4::cap_cast<Namespace>(target);
      else
        return err;
    }

  return _name.length;
}

long
Namespace::query(char const *name, unsigned len, L4::Cap<void> const &target,
                 int timeout, l4_umword_t *local_id, bool iterate) const noexcept
{
  if (L4_UNLIKELY(len == 0))
    return -L4_EINVAL;

  if (L4_UNLIKELY(timeout < 0))
    return -L4_EINVAL;

  long ret;
  long rem = timeout;
  long to = 0;

  if (rem)
    to = 10;

  do
    {
      ret = _query(name, len, target, local_id, iterate);

      if (ret >= 0)
        return ret;

      if (L4_UNLIKELY(ret != -L4_EAGAIN))
        return ret;

      if (rem == to)
        return ret;

      l4_sleep(to);

      if (rem > 0)
        {
```

```
          rem -= to;
          if (rem < 0)
            {
              to = rem = 0;
              continue; // one final try
            }
        }

      if (to < 100)
        to += to;
    }
  while (486);
}

long
Namespace::query(char const *name, L4::Cap<void> const &target,
                 int timeout, l4_umword_t *local_id,
                 bool iterate) const noexcept
{
  return query(name, __builtin_strlen(name), target,
               timeout, local_id, iterate);
}

}
```

16.201 l4/re/impl/rm_impl.h File Reference

Region map client stub implementation.

```
#include <l4/re/rm>
#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/task>
#include <l4/sys/err.h>
```

Include dependency graph for rm_impl.h:

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.201.1 Detailed Description

Region map client stub implementation.

Definition in file rm_impl.h.

16.202 rm_impl.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #include <l4/re/rm>
00024 #include <l4/re/dataspace>
00025
00026 #include <l4/sys/cxx/ipc_client>
00027
00028 #include <l4/sys/task>
00029 #include <l4/sys/err.h>
00030
00031 L4_RPC_DEF(L4Re::Rm::reserve_area);
00032 L4_RPC_DEF(L4Re::Rm::free_area);
00033 L4_RPC_DEF(L4Re::Rm::attach);
00034 L4_RPC_DEF(L4Re::Rm::detach);
00035 L4_RPC_DEF(L4Re::Rm::get_regions);
00036 L4_RPC_DEF(L4Re::Rm::get_areas);
00037 L4_RPC_DEF(L4Re::Rm::find);
00038
00039 namespace L4Re
00040 {
00041
00042 long
00043 Rm::attach(l4_addr_t *start, unsigned long size, Rm::Flags flags,
00044            L4::Ipc::Cap<Dataspace> mem, Rm::Offset offs,
00045            unsigned char align) const noexcept
00046 {
00047   if (((flags & F::Rights_mask) == Flags(0)) || (flags & F::Reserved))
00048     mem = L4::Ipc::Cap<L4Re::Dataspace>();
00049
00050   long e = attach_t::call(c(), start, size, flags, mem, offs, align, mem.cap().cap());
00051   if (e < 0)
00052     return e;
00053
00054   if (flags & F::Eager_map)
00055     e = mem.cap()->map_region(offs, map_flags(flags), *start, *start + size);
00056
00057   return e;
00058 }
00059
00060 int
00061 Rm::detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00062            L4::Cap<L4::Task> task, unsigned flags) const noexcept
00063 {
00064   l4_addr_t rstart = 0, rsize = 0;
00065   l4_cap_idx_t mem_cap = L4_INVALID_CAP;
00066   long e = detach_t::call(c(), start, size, flags, rstart, rsize, mem_cap);
00067   if (L4_UNLIKELY(e < 0))
00068     return e;
00069
00070   if (mem)
00071     *mem = L4::Cap<L4Re::Dataspace>(mem_cap);
00072
00073   if (!task.is_valid())
00074     return e;
00075
```

```
  rsize = l4_round_page(rsize);
  unsigned order = L4_LOG2_PAGESIZE;
  unsigned long sz = (1UL << order);
  for (unsigned long p = rstart; rsize; p += sz, rsize -= sz)
    {
      while (sz > rsize)
        {
          --order;
          sz >>= 1;
        }

      for (;;)
        {
          unsigned long m = sz << 1;
          if (m > rsize)
            break;

          if (p & (m - 1))
            break;

          ++order;
          sz <<= 1;
        }

      task->unmap(l4_fpage(p, order, L4_FPAGE_RWX),
                  L4_FP_ALL_SPACES);
    }

  return e;
}
}
```

16.203 l4/re/l4aux.h File Reference

Auxiliary definitions.

`#include <l4/sys/types.h>`

Include dependency graph for l4aux.h:

Data Structures

- struct l4re_aux_t

  Auxiliary descriptor.

Typedefs

- typedef struct l4re_aux_t l4re_aux_t

  Auxiliary descriptor.

Enumerations

- enum l4re_aux_ldr_flags_t

  Flags for program loading.

16.203.1 Detailed Description

Auxiliary definitions.

Definition in file l4aux.h.

16.204 l4aux.h

```
00001 #pragma once
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025
00026 #include <l4/sys/types.h>
00027
00039 enum l4re_aux_ldr_flags_t
00040 {
00041   L4RE_AUX_LDR_FLAG_EAGER_MAP    = 0x1,
00042   L4RE_AUX_LDR_FLAG_ALL_SEGS_COW = 0x2,
00043   L4RE_AUX_LDR_FLAG_PINNED_SEGS  = 0x4,
00044 };
00045
00051 typedef struct l4re_aux_t
00052 {
00053   char const *    binary;
00054   l4_cap_idx_t    kip_ds;
00055   l4_umword_t     dbg_lvl;
00056   l4_umword_t     ldr_flags;
00057 } l4re_aux_t;
00058
```

16.205 l4/re/log File Reference

Log interface.

`#include <l4/sys/vcon>`

Include dependency graph for log:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4Re::Log

  Log interface class.

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.205.1 Detailed Description

Log interface.

Definition in file log.

16.206 log

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/vcon>
00028
00029 namespace L4Re {
00030
00044 class L4_EXPORT Log : public L4::Kobject_t<Log, L4::Vcon, L4::PROTO_EMPTY>
00045 {
00046 public:
00047
00054   void printn(char const *string, int len) const noexcept;
00055
00061   void print(char const *string) const noexcept;
00062 };
00063 }
```

16.207 l4/re/log-sys.h File Reference

Log protocol definition.

Namespaces

- L4Re

  L4Re C++ Interfaces.

Enumerations

- enum L4Re::Log_::Opcodes

  Logging-service communication-protocol opcodes.

16.207.1 Detailed Description

Log protocol definition.

Definition in file log-sys.h.

16.208 log-sys.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027   namespace Log_
00028   {
00034     enum Opcodes { Print };
00035   };
00036 };
```

16.209 l4/re/mem_alloc File Reference

Memory allocator interface.

```
#include <l4/sys/capability>
#include <l4/sys/factory>
```

Include dependency graph for mem_alloc:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4Re::Mem_alloc

  Memory allocation interface.

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.209.1 Detailed Description

Memory allocator interface.

Definition in file mem_alloc.

16.210 mem_alloc

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * Copyright (C) 2015 Kernkonzept GmbH.
00009  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00010  */
00011 /*
00012  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00013  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00014  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00015  *     economic rights: Technische Universität Dresden (Germany)
00016  *
00017  * This file is part of TUD:OS and distributed under the terms of the
00018  * GNU General Public License 2.
00019  * Please see the COPYING-GPL-2 file for details.
00020  *
00021  * As a special exception, you may use this file as part of a free software
00022  * library without restriction.  Specifically, if other files instantiate
00023  * templates or use macros or inline functions from this file, or you compile
00024  * this file and link it with other files to produce an executable, this
00025  * file does not by itself cause the resulting executable to be covered by
00026  * the GNU General Public License.  This exception does not however
00027  * invalidate any other reasons why the executable file might be covered by
00028  * the GNU General Public License.
00029  */
00030 #pragma once
00031
00032 #include <l4/sys/capability>
00033 #include <l4/sys/factory>
00034
00035 namespace L4Re {
00036 class Dataspace;
00037
00038 // MISSING:
00039 // * alignment constraints
```

```
// * shall we support superpages in noncont memory?

class L4_EXPORT Mem_alloc :
  public L4::Kobject_t<Mem_alloc, L4::Factory, L4::PROTO_EMPTY>
{
public:
  enum Mem_alloc_flags
  {
    Continuous   = 0x01,
    Pinned       = 0x02,
    Super_pages  = 0x04,
  };

  long alloc(long size, L4::Cap<Dataspace> mem,
             unsigned long flags = 0, unsigned long align = 0) const noexcept;
};

};
```

16.211 l4/re/mem_alloc-sys.h File Reference

Memory allocator protocol definitions.

This graph shows which files directly or indirectly include this file:

Namespaces

- L4Re

  L4Re C++ Interfaces.

Enumerations

- enum L4Re::Mem_alloc_::Opcodes

  Memory-allocator communication-protocol opcodes.

16.211.1 Detailed Description

Memory allocator protocol definitions.

Definition in file mem_alloc-sys.h.

16.212 mem_alloc-sys.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027   namespace Mem_alloc_
00028   {
00034     enum Opcodes { Alloc, Free };
00035   };
00036 };
```

16.213 l4/re/mmio_space File Reference

Interface definition to emit MMIO-like accesses via IPC.

```
#include <l4/re/protocols.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for mmio_space:

Data Structures

- struct L4Re::Mmio_space

  Interface for memory-like address space accessible via IPC.

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.213.1 Detailed Description

Interface definition to emit MMIO-like accesses via IPC.

Definition in file mmio_space.

16.214 mmio_space

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * Copyright (C) 2017 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  * This file is distributed under the terms of the GNU General Public
00008  * License, version 2.  Please see the COPYING-GPL-2 file for details.
00009  */
00014 #pragma once
00015
00016 #include <l4/re/protocols.h>
00017 #include <l4/sys/capability>
00018 #include <l4/sys/cxx/ipc_types>
00019 #include <l4/sys/cxx/ipc_iface>
00020
00021 namespace L4Re
00022 {
00023
00046 struct L4_EXPORT Mmio_space
00047 : public L4::Kobject_t<Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE>
00048 {
00050   enum Access_width
00051   {
00052     Wd_8bit = 0,
00053     Wd_16bit = 1,
00054     Wd_32bit = 2,
00055     Wd_64bit = 3
00056   };
00057
00059   typedef l4_uint64_t Addr;
00060
00075   L4_INLINE_RPC(long, mmio_read, (Addr addr, char width, l4_uint64_t *value));
00076
00091   L4_INLINE_RPC(long, mmio_write, (Addr addr, char width, l4_uint64_t value));
00092
00093   typedef L4::Typeid::Rpcs<mmio_read_t, mmio_write_t> Rpcs;
00094 };
00095
00096 }
```

16.215 l4/re/namespace File Reference

Namespace interface.

```
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_array>
#include <l4/sys/cxx/ipc_string>
```

Include dependency graph for namespace:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4Re::Namespace

  Name-space interface.

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.215.1 Detailed Description

Namespace interface.

Definition in file namespace.

16.216 namespace

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *               Björn Döbel <doebel@os.inf.tu-dresden.de>
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction.  Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License.  This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029 #include <l4/re/protocols.h>
00030 #include <l4/sys/cxx/ipc_iface>
00031 #include <l4/sys/cxx/ipc_array>
00032 #include <l4/sys/cxx/ipc_string>
00033
00034 namespace L4Re {
00035
00060 class L4_EXPORT Namespace :
00061   public L4::Kobject_t<Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE,
00062                        L4::Type_info::Demand_t<1> >
00063 {
00064 public:
00068   enum Register_flags
00069   {
00070     Ro        = L4_CAP_FPAGE_RO,
00071     Rw        = L4_CAP_FPAGE_RW,
00072     Rs        = L4_CAP_FPAGE_RS,
00073     Rws       = L4_CAP_FPAGE_RWS,
00074     Strong    = L4_CAP_FPAGE_S,
00075     Trusted   = 0x008,
00076
00077     Cap_flags = Ro | Rw | Strong | Trusted,
00078
00079     Link      = 0x100,
00080     Overwrite = 0x200,
00081   };
00082
00088   enum Query_result_flags
00089   {
00090     Partly_resolved = 0x020,
00091   };
00092
00093   enum Query_timeout
00094   {
00095     To_default      = 3600000,
00096     To_non_blocking = 0,
00097     To_forever      = -1,
00098   };
00099
00100   L4_RPC_NF(
00101       long, query, (L4::Ipc::Array_ref<char const, unsigned long> name,
00102                     L4::Ipc::Small_buf cap,
00103                     L4::Ipc::Snd_fpage &snd_cap, L4::Ipc::Opt<L4::Opcode &> dummy,
00104                     L4::Ipc::Opt<L4::Ipc::Array_ref<char const, unsigned long> &> out_name));
00105
00131   long query(char const *name, L4::Cap<void> const &cap,
00132              int timeout = To_default,
00133              l4_umword_t *local_id = 0, bool iterate = true) const noexcept;
```

```
00134
00144   long query(char const *name, unsigned len, L4::Cap<void> const &cap,
00145              int timeout = To_default,
00146              l4_umword_t *local_id = 0, bool iterate = true) const noexcept;
00147
00148   L4_RPC_NF(long, register_obj, (unsigned flags,
00149                                  L4::Ipc::Array<char const, unsigned long> name,
00150                                  L4::Ipc::Opt< L4::Ipc::Cap<void> > obj),
00151             L4::Ipc::Call_t<L4_CAP_FPAGE_W>);
00152
00176   long register_obj(char const *name, L4::Ipc::Cap<void> obj,
00177                     unsigned flags = Rw) const noexcept
00178   {
00179     return register_obj_t::call(c(), flags,
00180                                 L4::Ipc::Array<char const, unsigned long>(
00181                                   __builtin_strlen(name), name),
00182                                 obj);
00183   }
00184
00185   L4_RPC_NF_OP(3, // backward compatibility opcode
00186                long, unlink, (L4::Ipc::Array<char const, unsigned long> name),
00187                L4::Ipc::Call_t<L4_CAP_FPAGE_W>);
00188
00202   long unlink(char const* name)
00203   {
00204     return unlink_t::call(c(), L4::Ipc::Array<char const, unsigned long>(
00205                                  __builtin_strlen(name), name));
00206   }
00207
00208   typedef L4::Typeid::Rpcs<query_t, register_obj_t, unlink_t> Rpcs;
00209
00210 private:
00211   long _query(char const *name, unsigned len,
00212               L4::Cap<void> const &target, l4_umword_t *local_id,
00213               bool iterate) const noexcept;
00214
00215 };
00216
00217 };
```

16.217 l4/re/namespace-sys.h File Reference

Namespace protocol definitions.

Namespaces

- L4Re

  L4Re C++ Interfaces.

Enumerations

- enum L4Re::Namespace_::Opcodes

  Name-space communication-protocol opcodes.

16.217.1 Detailed Description

Namespace protocol definitions.

Definition in file namespace-sys.h.

16.218 namespace-sys.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4Re {
00026   namespace Namespace_
00027   {
00033     enum Opcodes { Query, Register, Link, Unlink };
00034   };
00035 };
```

16.219 l4/re/parent File Reference

Parent interface.

```
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for parent:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4Re::Parent

  Parent interface.

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.219.1 Detailed Description

Parent interface.

Definition in file parent.

16.220 parent

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/capability>
00028 #include <l4/re/protocols.h>
00029 #include <l4/sys/cxx/ipc_iface>
00030
00031 namespace L4Re {
00032
00053 class L4_EXPORT Parent :
00054   public L4::Kobject_t<Parent, L4::Kobject, L4RE_PROTO_PARENT>
00055 {
00056 public:
00066   L4_INLINE_RPC(long, signal, (unsigned long sig, unsigned long val));
00067   typedef L4::Typeid::Rpcs<signal_t> Rpcs;
00068 };
00069 };
00070
```

16.221 l4/re/parent-sys.h File Reference

Parent protocol definition.

Namespaces

- L4Re

  L4Re C++ Interfaces.

Enumerations

- enum L4Re::Parent_::Opcodes

  Parent communication-protocol opcodes.

16.221.1 Detailed Description

Parent protocol definition.

Definition in file parent-sys.h.

16.222 parent-sys.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027   namespace Parent_
00028   {
00034     enum Opcodes { Signal };
00035   };
00036 };
```

16.223 l4/re/protocols.h File Reference

L4Re Protocol Constants (C version)

This graph shows which files directly or indirectly include this file:

Enumerations

- enum L4re_protocols {
  L4RE_PROTO_DATASPACE = 0x4000 , L4RE_PROTO_NAMESPACE , L4RE_PROTO_PARENT ,
  L4RE_PROTO_GOOS ,
  L4RE_PROTO_RSVD_1 , L4RE_PROTO_RM , L4RE_PROTO_EVENT , L4RE_PROTO_INHIBITOR ,
  L4RE_PROTO_DMA_SPACE , L4RE_PROTO_MMIO_SPACE , L4RE_PROTO_DEBUG = ~0x7fffL }

16.223.1 Detailed Description

L4Re Protocol Constants (C version)

Definition in file protocols.h.

16.223.2 Enumeration Type Documentation

16.223.2.1 L4re_protocols

enum L4re_protocols

Enumerator

L4RE_PROTO_DATASPACE	ID for L4Re::Dataspace RPCs
L4RE_PROTO_NAMESPACE	ID for L4Re::Namespace RPCs
L4RE_PROTO_PARENT	ID for L4Re::Parent RPCs
L4RE_PROTO_GOOS	ID for L4Re::Video::Goos RPCs
L4RE_PROTO_RSVD_1	Reserved ID
L4RE_PROTO_RM	ID for L4Re::Rm RPCs
L4RE_PROTO_EVENT	ID for L4Re::Event RPCs
L4RE_PROTO_INHIBITOR	ID for L4Re::Inhibitor RPCs
L4RE_PROTO_DMA_SPACE	ID for L4Re::Dma_space RPCs
L4RE_PROTO_MMIO_SPACE	ID for L4Re::Mmio_space
L4RE_PROTO_DEBUG	ID for debugging RPCs

Definition at line 30 of file protocols.h.

16.224 protocols.h

```
00001
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00030 enum L4re_protocols
00031 {
00032   L4RE_PROTO_DATASPACE = 0x4000,
```

```
00033   L4RE_PROTO_NAMESPACE,
00034   L4RE_PROTO_PARENT,
00035   L4RE_PROTO_GOOS,
00036   L4RE_PROTO_RSVD_1,
00037   L4RE_PROTO_RM,
00038   L4RE_PROTO_EVENT,
00039   L4RE_PROTO_INHIBITOR,
00040   L4RE_PROTO_DMA_SPACE,
00041   L4RE_PROTO_MMIO_SPACE,
00043   L4RE_PROTO_DEBUG = ~0x7fffL
00044 };
00045
```

16.225 l4/re/random File Reference

Random number generator interface definition.

```
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/icu>
```

Include dependency graph for random:

Data Structures

- struct L4Re::Random

  Low-bandwidth interface for random number generators.

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.225.1 Detailed Description

Random number generator interface definition.

Definition in file random.

16.226 random

```
00001 // -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: ((GPL-2.0-only WITH mif-exception) OR LicenseRef-kk-custom) */
00003 /*
00004  * Copyright (C) 2019-2020 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/capability>
00015 #include <l4/sys/cxx/ipc_types>
00016 #include <l4/sys/cxx/ipc_iface>
00017 #include <l4/sys/icu>
00018
00019 namespace L4Re
00020 {
00021
00033 struct L4_EXPORT Random
00034 : public L4::Kobject_t<Random, L4::Icu>
00035 {
00060   L4_INLINE_RPC(long, get_random, (l4_size_t size,
00061                                    L4::Ipc::Array<char, unsigned long> *buffer));
00062
00063   typedef L4::Typeid::Rpcs<get_random_t> Rpcs;
00064 };
00065
00066 } // namespace
```

16.227 l4/re/rm File Reference

Region mapper interface.

```
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/pager>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_ret_array>
#include <l4/sys/cxx/types>
#include <l4/re/consts>
#include <l4/re/dataspace>
```

Include dependency graph for rm:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4Re::Rm

  Region map.
- struct L4Re::Rm::F

  Rm flags definitions.
- struct L4Re::Rm::Range

  A range of virtual addresses.

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.227.1 Detailed Description

Region mapper interface.

Definition in file rm.

16.228 rm

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU General Public License 2.
00016  * Please see the COPYING-GPL-2 file for details.
00017  *
00018  * As a special exception, you may use this file as part of a free software
00019  * library without restriction.  Specifically, if other files instantiate
00020  * templates or use macros or inline functions from this file, or you compile
00021  * this file and link it with other files to produce an executable, this
00022  * file does not by itself cause the resulting executable to be covered by
00023  * the GNU General Public License.  This exception does not however
00024  * invalidate any other reasons why the executable file might be covered by
00025  * the GNU General Public License.
00026  */
00027 #pragma once
00028
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/l4int.h>
00031 #include <l4/sys/capability>
00032 #include <l4/re/protocols.h>
00033 #include <l4/sys/pager>
00034 #include <l4/sys/cxx/ipc_iface>
00035 #include <l4/sys/cxx/ipc_ret_array>
00036 #include <l4/sys/cxx/types>
00037 #include <l4/re/consts>
00038 #include <l4/re/dataspace>
00039
00040 namespace L4Re {
00041
00080 class L4_EXPORT Rm :
00081   public L4::Kobject_t<Rm, L4::Pager, L4RE_PROTO_RM,
00082                        L4::Type_info::Demand_t<1> >
00083 {
00084 public:
00085   typedef L4Re::Dataspace::Offset Offset;
00086
00088   enum Detach_result
00089   {
00090     Detached_ds  = 0,
00091     Kept_ds      = 1,
00092     Split_ds     = 2,
00093     Detach_result_mask = 3,
00094
00095     Detach_again = 4,
00096   };
00097
00098
00099   enum Region_flag_shifts
00100   {
00102     Caching_shift    = Dataspace::F::Caching_shift,
00103   };
00104
00106   struct F
00107   {
00109     enum Attach_flags : l4_uint32_t
00110     {
00112       Search_addr  = 0x20000,
00114       In_area      = 0x40000,
00116       Eager_map    = 0x80000,
00118       Attach_mask  = 0xf0000,
00119     };
00120
00121     L4_TYPES_FLAGS_OPS_DEF(Attach_flags);
00122
00123     enum Region_flags : l4_uint16_t
00124     {
00126       Rights_mask = 0x0f,
00128       R         = Dataspace::F::R,
00130       W         = Dataspace::F::W,
00132       X         = Dataspace::F::X,
00134       RW        = Dataspace::F::RW,
00136       RX        = Dataspace::F::RX,
00138       RWX       = Dataspace::F::RWX,
00139
00141       Detach_free = 0x200,
00143       Pager       = 0x400,
00145       Reserved    = 0x800,
```

```
00146
00147
00149       Caching_mask   = Dataspace::F::Caching_mask,
00151       Cache_normal   = Dataspace::F::Normal,
00153       Cache_buffered = Dataspace::F::Bufferable,
00155       Cache_uncached = Dataspace::F::Uncacheable,
00156
00158       Ds_map_mask    = 0xff,
00159
00161       Region_flags_mask = 0xffff,
00162     };
00163
00164     L4_TYPES_FLAGS_OPS_DEF(Region_flags);
00165
00166     friend constexpr Dataspace::Flags map_flags(Region_flags rf)
00167     {
00168       return Dataspace::Flags((l4_uint16_t)rf & Ds_map_mask);
00169     }
00170
00171     struct Flags : L4::Types::Flags_ops_t<Flags>
00172     {
00173       l4_uint32_t raw;
00174       Flags() = default;
00175       explicit constexpr Flags(l4_uint32_t f) : raw(f) {}
00176       constexpr Flags(Attach_flags rf) : raw((l4_uint32_t)rf) {}
00177       constexpr Flags(Region_flags rf) : raw((l4_uint32_t)rf) {}
00178
00179       friend constexpr Dataspace::Flags map_flags(Flags f)
00180       {
00181         return Dataspace::Flags(f.raw & Ds_map_mask);
00182       }
00183
00184       constexpr Region_flags region_flags() const
00185       {
00186         return Region_flags(raw & Region_flags_mask);
00187       }
00188
00189       constexpr Attach_flags attach_flags() const
00190       {
00191         return Attach_flags(raw & Attach_mask);
00192       }
00193
00194       constexpr bool r() const { return raw & L4_FPAGE_RO; }
00195       constexpr bool w() const { return raw & L4_FPAGE_W; }
00196       constexpr bool x() const { return raw & L4_FPAGE_X; }
00197       constexpr unsigned cap_rights() const
00198       { return w() ? L4_CAP_FPAGE_RW : L4_CAP_FPAGE_RO; }
00199     };
00200
00201     friend constexpr Flags operator | (Region_flags l, Attach_flags r)
00202     { return Flags(l) | Flags(r); }
00203
00204     friend constexpr Flags operator | (Attach_flags l, Region_flags r)
00205     { return Flags(l) | Flags(r); }
00206   };
00207
00208   using Attach_flags = F::Attach_flags;
00209   using Region_flags = F::Region_flags;
00210   using Flags = F::Flags;
00211
00213   enum Detach_flags
00214   {
00224     Detach_exact   = 1,
00234     Detach_overlap = 2,
00235
00243     Detach_keep = 4,
00244   };
00245
00272   long reserve_area(l4_addr_t *start, unsigned long size,
00273                     Flags flags = Flags(0),
00274                     unsigned char align = L4_PAGESHIFT) const noexcept
00275   { return reserve_area_t::call(c(), start, size, flags, align); }
00276
00277   L4_RPC_NF(long, reserve_area, (L4::Ipc::In_out<l4_addr_t *> start,
00278                                  unsigned long size,
00279                                  Flags flags,
00280                                  unsigned char align));
00281
00297   template< typename T >
00298   long reserve_area(T **start, unsigned long size,
00299                     Flags flags = Flags(0),
00300                     unsigned char align = L4_PAGESHIFT) const noexcept
00301   { return reserve_area_t::call(c(), (l4_addr_t*)start, size, flags, align); }
00302
00315   L4_RPC(long, free_area, (l4_addr_t addr));
00316
00317   L4_RPC_NF(long, attach, (L4::Ipc::In_out<l4_addr_t *> start,
```

```
                              unsigned long size, Flags flags,
                              L4::Ipc::Opt<L4::Ipc::Cap<Dataspace> > mem,
                              Offset offs, unsigned char align,
                              L4::Ipc::Opt<l4_cap_idx_t> client_cap));

  L4_RPC_NF(long, detach, (l4_addr_t addr, unsigned long size, unsigned flags,
                              l4_addr_t &start, l4_addr_t &rsize,
                              l4_cap_idx_t &mem_cap));

  long attach(l4_addr_t *start, unsigned long size, Flags flags,
              L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
              unsigned char align = L4_PAGESHIFT) const noexcept;

  template< typename T >
  long attach(T **start, unsigned long size, Flags flags,
              L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
              unsigned char align = L4_PAGESHIFT) const noexcept
  {
    union X { l4_addr_t a; T* t; };
    X *x = reinterpret_cast<X*>(start);
    return attach(&x->a, size, flags, mem, offs, align);
  }

#if __cplusplus >= 201103L
  template< typename T >
  class Unique_region
  {
  private:
    T _addr;
    L4::Cap<Rm> _rm;

  public:
    Unique_region(Unique_region const &) = delete;
    Unique_region &operator = (Unique_region const &) = delete;

    Unique_region() noexcept
    : _addr(0), _rm(L4::Cap<Rm>::Invalid) {}

    explicit Unique_region(T addr) noexcept
    : _addr(addr), _rm(L4::Cap<Rm>::Invalid) {}

    Unique_region(T addr, L4::Cap<Rm> const &rm) noexcept
    : _addr(addr), _rm(rm) {}

    Unique_region(Unique_region &&o) noexcept : _addr(o.get()), _rm(o._rm)
    { o.release(); }

    Unique_region &operator = (Unique_region &&o) noexcept
    {
      if (&o != this)
        {
          if (_rm.is_valid())
            _rm->detach(l4_addr_t(_addr), 0);
          _rm = o._rm;
          _addr = o.release();
        }
      return *this;
    }

    ~Unique_region() noexcept
    {
      if (_rm.is_valid())
        _rm->detach(l4_addr_t(_addr), 0);
    }

    T get() const noexcept
    { return _addr; }

    T release() noexcept
    {
      _rm = L4::Cap<Rm>::Invalid;
      return _addr;
    }

    void reset(T addr, L4::Cap<Rm> const &rm) noexcept
    {
      if (_rm.is_valid())
        _rm->detach(l4_addr_t(_addr), 0);

      _rm = rm;
      _addr = addr;
    }

    void reset() noexcept
    { reset(0, L4::Cap<Rm>::Invalid); }

    bool is_valid() const noexcept
```

```
    { return _rm.is_valid(); }

    T operator * () const noexcept { return _addr; }

    T operator -> () const noexcept { return _addr; }
  };

  template< typename T >
  long attach(Unique_region<T> *start, unsigned long size, Flags flags,
              L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
              unsigned char align = L4_PAGESHIFT) const noexcept
  {
    l4_addr_t addr = (l4_addr_t)start->get();

    long res = attach(&addr, size, flags, mem, offs, align);
    if (res < 0)
      return res;

    start->reset((T)addr, L4::Cap<Rm>(cap()));
    return res;
  }
#endif

  int detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
             L4::Cap<L4::Task> const &task = This_task) const noexcept;

  int detach(void *addr, L4::Cap<Dataspace> *mem,
             L4::Cap<L4::Task> const &task = This_task) const noexcept;

  int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
             L4::Cap<L4::Task> const &task) const noexcept;

  int find(l4_addr_t *addr, unsigned long *size, Offset *offset,
           L4Re::Rm::Flags *flags, L4::Cap<Dataspace> *m) noexcept
  { return find_t::call(c(), addr, size, flags, offset, m); }

  L4_RPC_NF(int, find, (L4::Ipc::In_out<l4_addr_t *> addr,
                        L4::Ipc::In_out<unsigned long *> size,
                        L4Re::Rm::Flags *flags, Offset *offset,
                        L4::Ipc::As_value<L4::Cap<Dataspace> > *m));

  struct Range
  {
    l4_addr_t start;
    l4_addr_t end;
  };

  using Region = Range;

  using Area = Range;

  L4_RPC(long, get_regions, (l4_addr_t start, L4::Ipc::Ret_array<Range> regions));

  L4_RPC(long, get_areas, (l4_addr_t start, L4::Ipc::Ret_array<Range> areas));

  int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
             L4::Cap<L4::Task> task, unsigned flags) const noexcept;

  typedef L4::Typeid::Rpcs<attach_t, detach_t, find_t,
                           reserve_area_t, free_area_t,
                           get_regions_t, get_areas_t> Rpcs;
};


inline int
Rm::detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
           L4::Cap<L4::Task> const &task) const noexcept
{  return detach(addr, 1, mem, task, Detach_overlap); }

inline int
Rm::detach(void *addr, L4::Cap<Dataspace> *mem,
           L4::Cap<L4::Task> const &task) const noexcept
{  return detach((l4_addr_t)addr, 1, mem, task, Detach_overlap); }

inline int
Rm::detach(l4_addr_t addr, unsigned long size, L4::Cap<Dataspace> *mem,
           L4::Cap<L4::Task> const &task) const noexcept
{  return detach(addr, size, mem, task, Detach_exact); }

};
```

16.229 l4/re/rm-sys.h File Reference

Region mapper protocol definitions.

Namespaces

- L4Re

  L4Re C++ Interfaces.

Enumerations

- enum L4Re::Rm_::Opcodes

  Region-map communication-protocol opcodes.

16.229.1 Detailed Description

Region mapper protocol definitions.

Definition in file rm-sys.h.

16.230 rm-sys.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027   namespace Rm_
00028   {
00034     enum Opcodes
00035     {
00036       Attach, Detach, Find, Attach_area, Detach_area, Get_regions, Get_areas
00037     };
00038   };
00039 };
```

16.231 l4/re/shared_cap File Reference

Shared_cap / Shared_del_cap.

```
#include <l4/re/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>
```

Include dependency graph for shared_cap:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4Re

  L4Re C++ Interfaces.

Typedefs

- template<typename T >
  using L4Re::Shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >

  Shared capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using L4Re::shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >

  Shared capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using L4Re::Shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ > >

  Shared capability that implements automatic free and unmap+delete of the capability selector.
- template<typename T >
  using L4Re::shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ > >

  Shared capability that implements automatic free and unmap+delete of the capability selector.

Functions

- template<typename T >
  Shared_cap< T > L4Re::make_shared_cap (L4Re::Cap_alloc *ca)

  Allocate a capability slot and wrap it in a Shared_cap.
- template<typename T >
  Shared_del_cap< T > L4Re::make_shared_del_cap (L4Re::Cap_alloc *ca)

  Allocate a capability slot and wrap it in a Shared_del_cap.

16.231.1 Detailed Description

Shared_cap / Shared_del_cap.

Definition in file shared_cap.

16.232 shared_cap

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2018 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/re/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_1x>
00027
00028 namespace L4Re {
00029
00043 template< typename T >
00044 using Shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES»;
```

```
template< typename T >
using shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES»;

template< typename T >
Shared_cap<T>
make_shared_cap(L4Re::Cap_alloc *ca)
{ return Shared_cap<T>(ca->alloc<T>(), ca); }

template< typename T >
using Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ»;
template<typename T>
using shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ»;

template< typename T >
Shared_del_cap<T>
make_shared_del_cap(L4Re::Cap_alloc *ca)
{ return Shared_del_cap<T>(ca->alloc<T>(), ca); }

} // namespace L4Re
```

16.233 l4/re/util/shared_cap File Reference

Shared_cap / Shared_del_cap.

```
#include <l4/re/util/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>
```

Include dependency graph for shared_cap:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4Re

  L4Re C++ Interfaces.
- L4Re::Util

  Documentation of the L4 Runtime Environment utility functionality in C++.

Typedefs

- template<typename T >
  using L4Re::Util::Shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >

  Shared capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using L4Re::Util::shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >

  Shared capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using L4Re::Util::Shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ > >

  Shared capability that implements automatic free and unmap+delete of the capability selector.
- template<typename T >
  using L4Re::Util::shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ > >

  Shared capability that implements automatic free and unmap+delete of the capability selector.

Functions

- template<typename T >
  Shared_cap< T > L4Re::Util::make_shared_cap ()

  Allocate a capability slot and wrap it in a Shared_cap.
- template<typename T >
  Shared_del_cap< T > L4Re::Util::make_shared_del_cap ()

  Allocate a capability slot and wrap it in a Shared_del_cap.

16.233.1 Detailed Description

Shared_cap / Shared_del_cap.

Definition in file shared_cap.

16.234 shared_cap

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/re/util/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_1x>
00027
00028 namespace L4Re { namespace Util {
00029
00058 template< typename T >
00059 using Shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES»;
00061 template< typename T >
00062 using shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES»;
00063
00069 template< typename T >
00070 Shared_cap<T>
00071 make_shared_cap()
00072 { return Shared_cap<T>(cap_alloc.alloc<T>()); }
00073
00108 template< typename T >
00109 using Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ»;
00111 template< typename T >
00112 using shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ»;
00113
00119 template< typename T >
00120 Shared_del_cap<T>
00121 make_shared_del_cap()
00122 { return Shared_del_cap<T>(cap_alloc.alloc<T>()); }
00123
00124 }} // namespace L4Re::Util
00125
```

16.235 l4/re/unique_cap File Reference

Unique_cap / Unique_del_cap.

```
#include <l4/re/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>
```

Include dependency graph for unique_cap:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4Re

  L4Re C++ Interfaces.

Typedefs

- template<typename T >
  using L4Re::Unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >

  Unique capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using L4Re::unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >

  Unique capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using L4Re::Unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >

  Unique capability that implements automatic free and unmap+delete of the capability selector.
- template<typename T >
  using L4Re::unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >

  Unique capability that implements automatic free and unmap+delete of the capability selector.

Functions

- template<typename T >
  Unique_cap< T > L4Re::make_unique_cap (L4Re::Cap_alloc *ca)

  Allocate a capability slot and wrap it in an Unique_cap.
- template<typename T >
  Unique_del_cap< T > L4Re::make_unique_del_cap (L4Re::Cap_alloc *ca)

  Allocate a capability slot and wrap it in an Unique_del_cap.

16.235.1 Detailed Description

Unique_cap / Unique_del_cap.

Definition in file unique_cap.

16.236 unique_cap

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/re/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_1x>
00027
00028 namespace L4Re {
00029
00041 template< typename T >
00042 using Unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>;
```

```
00044 template< typename T >
00045 using unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES»;
00046
00056 template< typename T >
00057 Unique_cap<T>
00058 make_unique_cap(L4Re::Cap_alloc *ca)
00059 { return Unique_cap<T>(ca->alloc<T>(), ca); }
00060
00074 template< typename T >
00075 using Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ»;
00077 template<typename T>
00078 using unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ»;
00079
00089 template< typename T >
00090 Unique_del_cap<T>
00091 make_unique_del_cap(L4Re::Cap_alloc *ca)
00092 { return Unique_del_cap<T>(ca->alloc<T>(), ca); }
00093
00094 }
```

16.237 l4/re/util/unique_cap File Reference

Unique_cap / Unique_del_cap.

```
#include <l4/re/util/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>
```

Include dependency graph for unique_cap:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4Re

  L4Re C++ Interfaces.
- L4Re::Util

  Documentation of the L4 Runtime Environment utility functionality in C++.

Typedefs

- template<typename T >
  using L4Re::Util::Unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >

  Unique capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using L4Re::Util::unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >

  Unique capability that implements automatic free and unmap of the capability selector.
- template<typename T >
  using L4Re::Util::Unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >

  Unique capability that implements automatic free and unmap+delete of the capability selector.
- template<typename T >
  using L4Re::Util::unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >

  Unique capability that implements automatic free and unmap+delete of the capability selector.

Functions

- template<typename T >
  Unique_cap< T > L4Re::Util::make_unique_cap ()

  Allocate a capability slot and wrap it in an Unique_cap.
- template<typename T >
  Unique_del_cap< T > L4Re::Util::make_unique_del_cap ()

  Allocate a capability slot and wrap it in an Unique_del_cap.

16.237.1 Detailed Description

Unique_cap / Unique_del_cap.

Definition in file unique_cap.

16.238 unique_cap

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/re/util/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_1x>
00027
00028 namespace L4Re { namespace Util {
00029
00053 template< typename T >
00054 using Unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES»;
00056 template< typename T >
00057 using unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES»;
00058
00064 template< typename T >
00065 Unique_cap<T>
00066 make_unique_cap()
00067 { return Unique_cap<T>(cap_alloc.alloc<T>()); }
00068
00096 template< typename T >
00097 using Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ»;
00098 template< typename T >
00100 using unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ»;
00101
00107 template< typename T >
00108 Unique_del_cap<T>
00109 make_unique_del_cap()
00110 { return Unique_del_cap<T>(cap_alloc.alloc<T>()); }
00111
00112 }}
00113
```

16.239 l4/re/util/bitmap_cap_alloc File Reference

Bitmap capability allocator.

```
#include <l4/re/util/item_alloc>
#include <l4/sys/capability>
```

```
#include <l4/sys/task.h>
```

Include dependency graph for bitmap_cap_alloc:

Data Structures

- class L4Re::Util::Cap_alloc_base

  Capability allocator.

Namespaces

- L4Re

  L4Re C++ Interfaces.
- L4Re::Util

  Documentation of the L4 Runtime Environment utility functionality in C++.

16.239.1 Detailed Description

Bitmap capability allocator.

Definition in file bitmap_cap_alloc.

16.240 bitmap_cap_alloc

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025
00026 #pragma once
00027
00028 #include <l4/re/util/item_alloc>
00029 #include <l4/sys/capability>
00030 #include <l4/sys/task.h>
00031
00032 namespace L4Re { namespace Util {
00033
00038 class Cap_alloc_base
00039 {
00040 private:
00041   long _bias;
00042   Item_alloc_base _items;
00043
00044 public:
00045   enum State { Free = 0, Allocated, Unknown };
00046   Cap_alloc_base(long max, void *mem, long bias = 0)
00047     noexcept : _bias(bias), _items(max, mem) {}
00048
00049   L4::Cap<void> alloc() noexcept
00050   {
00051     long cap = _items.alloc();
00052     if (cap < 0)
00053       return L4::Cap<void>::Invalid;
00054
00055     return L4::Cap<void>((cap + _bias) « L4_CAP_SHIFT);
00056   }
00057
00058   long hint() const { return _items.hint(); }
00059
00063   template< typename T >
00064   L4::Cap<T> alloc() noexcept
00065   { return L4::Cap<T>(alloc().cap()); }
00066
00067   State is_allocated(L4::Cap<void> c) const noexcept
00068   {
00069     long idx = (c.cap() » L4_CAP_SHIFT);
00070
00071     if (idx < _bias)
00072       return Unknown;
00073
00074     idx -= _bias;
00075     return _items.is_allocated(idx) ? Allocated : Free;
00076   }
00077
00081   template< typename T>
00082   void free(L4::Cap<T> const &cap, l4_cap_idx_t task = L4_INVALID_CAP,
00083             l4_umword_t unmap_flags = L4_FP_ALL_SPACES) noexcept
00084   {
00085     long idx = (cap.cap() » L4_CAP_SHIFT);
00086     if (idx < _bias)
00087       return;
00088
00089     idx -= _bias;
00090
00091     _items.free(idx);
00092
00093     if (l4_is_valid_cap(task))
00094       l4_task_unmap(task, cap.fpage(), unmap_flags | 2);
00095   }
00096
00097   // since we have no counters assume counter always > 0
00098   void take(L4::Cap<void>) noexcept {}
00099   bool release(L4::Cap<void>, l4_cap_idx_t task = L4_INVALID_CAP,
```

```
                unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
  { (void)task; (void)unmap_flags; return false; }

  long last() noexcept
  {
    return _items.size() + _bias - 1;
  }
};

template< long Size >
class Cap_alloc : public Cap_alloc_base
{
private:
  typename Bitmap_base::Word<Size>::Type _bits[Bitmap_base::Word<Size>::Size];

public:
  explicit Cap_alloc(long bias = 0) noexcept
    : Cap_alloc_base(Size, _bits, bias) {}

};

}
}
```

16.241 l4/re/util/cap File Reference

Capability utility functions.

`#include <l4/sys/task>`

Include dependency graph for cap:

Namespaces

- L4Re

  L4Re C++ Interfaces.
- L4Re::Util

  Documentation of the L4 Runtime Environment utility functionality in C++.

16.241.1 Detailed Description

Capability utility functions.

Definition in file cap.

16.242 cap

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task>
00028
00029 namespace L4Re { namespace Util {
00030
00038 L4_CV static inline l4_msgtag_t cap_release(L4::Cap<void> cap)
00039 {
00040   return l4_task_unmap(L4_BASE_TASK_CAP,
00041                        l4_obj_fpage(cap.cap(), 0, L4_FPAGE_RWX),
00042            L4_FP_ALL_SPACES);
00043 }
00044
00045 }}
```

16.243 l4/re/util/cap_alloc_impl.h File Reference

Capability allocator implementation.

```
#include <l4/re/util/counting_cap_alloc>
```

Include dependency graph for cap_alloc_impl.h:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4Re

  L4Re C++ Interfaces.
- L4Re::Util

  Documentation of the L4 Runtime Environment utility functionality in C++.

16.243.1 Detailed Description

Capability allocator implementation.

Definition in file cap_alloc_impl.h.

16.244 cap_alloc_impl.h

```
// -*- Mode: C++ -*-
// vim:ft=cpp
/*
 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */

#pragma once

//#define L4RE_STATIC_CAP_ALLOC
#if defined(L4RE_STATIC_CAP_ALLOC)

#include <l4/re/util/bitmap_cap_alloc>

namespace L4Re { namespace Util {

typedef Cap_alloc_base _Cap_alloc;

}}

#else
#include <l4/re/util/counting_cap_alloc>

namespace L4Re { namespace Util {

typedef Counting_cap_alloc<L4Re::Util::Counter<unsigned char> > _Cap_alloc;

}}
#endif


```

16.245 l4/re/util/counting_cap_alloc File Reference

Reference-counting capability allocator.

```
#include <l4/sys/task>
#include <l4/sys/assert.h>
#include <l4/re/consts>
```

Include dependency graph for counting_cap_alloc:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct L4Re::Util::Counter< COUNTER >

  Counter for Counting_cap_alloc with variable data width.
- class L4Re::Util::Counting_cap_alloc< COUNTERTYPE >

  Internal reference-counting cap allocator.

Namespaces

- L4Re

  L4Re C++ Interfaces.
- L4Re::Util

  Documentation of the L4 Runtime Environment utility functionality in C++.

16.245.1 Detailed Description

Reference-counting capability allocator.

Definition in file counting_cap_alloc.

16.246 counting_cap_alloc

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023
00024 #pragma once
00025
00026 #include <l4/sys/task>
00027 #include <l4/sys/assert.h>
00028 #include <l4/re/consts>
00029
00030 namespace L4Re { namespace Util {
00031
00035 template< typename COUNTER = unsigned char >
00036 struct Counter
00037 {
00038   typedef COUNTER Type;
00039   Type _cnt;
00040
00041   static Type nil() { return 0; }
00042
00043   void free() { _cnt = 0; }
00044   bool is_free() const { return _cnt == 0; }
00045   void inc() { ++_cnt; }
00046   Type dec() { return --_cnt; }
00047   void alloc() { _cnt = 1; }
00048 };
00049
00074 template <typename COUNTERTYPE = L4Re::Util::Counter<unsigned char> >
00075 class Counting_cap_alloc
00076 {
00077 private:
00078   void operator = (Counting_cap_alloc const &) { }
00079   typedef COUNTERTYPE Counter;
00080
00081   COUNTERTYPE *_items;
00082   long _free_hint;
00083   long _bias;
00084   long _capacity;
00085
00086
00087 public:
00088
00089   template <unsigned COUNT>
00090   struct Counter_storage
00091   {
00092     COUNTERTYPE _buf[COUNT];
00093     typedef COUNTERTYPE Buf_type[COUNT];
00094     enum { Size = COUNT };
00095   };
00096
00097 protected:
00098
00104   Counting_cap_alloc() noexcept
00105   : _items(0), _free_hint(0), _bias(0), _capacity(0)
00106   {}
00107
00121   void setup(void *m, long capacity, long bias) noexcept
```

```
00122   {
00123     _items = (Counter*)m;
00124     _capacity = capacity;
00125     _bias = bias;
00126   }
00127
00128 public:
00135   L4::Cap<void> alloc() noexcept
00136   {
00137     for (long i = _free_hint; i < _capacity; ++i)
00138       {
00139   if (_items[i].is_free())
00140     {
00141       _items[i].alloc();
00142       _free_hint = i + 1;
00143
00144       return L4::Cap<void>((i + _bias) « L4_CAP_SHIFT);
00145     }
00146       }
00147
00148     return L4::Cap<void>::Invalid;
00149   }
00150
00152   template <typename T>
00153   L4::Cap<T> alloc() noexcept
00154   {
00155     return L4::cap_cast<T>(alloc());
00156   }
00157
00158
00168   void take(L4::Cap<void> cap) noexcept
00169   {
00170     long c;
00171     if (!range_check_and_get_idx(cap, &c))
00172       return;
00173
00174     _items[c].inc();
00175   }
00176
00177
00191   bool free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00192             unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00193   {
00194     long c;
00195     if (!range_check_and_get_idx(cap, &c))
00196       return false;
00197
00198     l4_assert(!_items[c].is_free());
00199
00200     if (l4_is_valid_cap(task))
00201       l4_task_unmap(task, cap.fpage(), unmap_flags);
00202
00203     if (c < _free_hint)
00204       _free_hint = c;
00205
00206     _items[c].free();
00207
00208     return true;
00209   }
00210
00229   bool release(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00230                unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00231   {
00232     long c;
00233     if (!range_check_and_get_idx(cap, &c))
00234       return false;
00235
00236     l4_assert(!_items[c].is_free());
00237
00238     if (_items[c].dec() == Counter::nil())
00239       {
00240   if (task != L4_INVALID_CAP)
00241     l4_task_unmap(task, cap.fpage(), unmap_flags);
00242
00243   if (c < _free_hint)
00244     _free_hint = c;
00245
00246   return true;
00247       }
00248     return false;
00249   }
00250
00254   long last() noexcept
00255   {
00256     return _capacity + _bias - 1;
00257   }
00258
```

```
private:
  bool range_check_and_get_idx(L4::Cap<void> cap, long *c)
  {
    *c = cap.cap() » L4_CAP_SHIFT;
    if (*c < _bias)
      return false;

    *c -= _bias;

    return *c < _capacity;
  }
};

}}
```

16.247 l4/re/util/item_alloc File Reference

Item allocator.

`#include <l4/cxx/bitmap>`

Include dependency graph for item_alloc:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4Re::Util::Item_alloc_base

  Item allocator.

Namespaces

- L4Re

  L4Re C++ Interfaces.
- L4Re::Util

  Documentation of the L4 Runtime Environment utility functionality in C++.

16.247.1 Detailed Description

Item allocator.

Definition in file item_alloc.

16.248 item_alloc

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025
00026 #pragma once
00027
00028 #include <l4/cxx/bitmap>
00029
00030 namespace L4Re { namespace Util {
00031
00032 using cxx::Bitmap_base;
00033 using cxx::Bitmap;
00034
00038 class Item_alloc_base
00039 {
00040 private:
00041   long _capacity;
00042   long _free_hint;
00043   Bitmap_base _bits;
00044
00045 public:
00046   bool is_allocated(long item) const noexcept
00047   { return _bits[item]; }
00048
00049   long hint() const { return _free_hint; }
00050
00051   bool alloc(long item) noexcept
00052   {
00053     if (!_bits[item])
00054       {
00055   _bits.set_bit(item);
00056   return true;
00057       }
00058     return false;
00059   }
00060
00061   void free(long item) noexcept
00062   {
00063     if (item < _free_hint)
00064       _free_hint = item;
00065
00066     _bits.clear_bit(item);
```

```
  }

  Item_alloc_base(long size, void *mem) noexcept
    : _capacity(size), _free_hint(0), _bits(mem)
  {}

  long alloc() noexcept
  {
    if (_free_hint >= _capacity)
      return -1;

    long free = _bits.scan_zero(_capacity, _free_hint);
    if (free >= 0)
      {
  _bits.set_bit(free);
  _free_hint += 1;
      }
    return free;
  }

  long size() const noexcept
  {
    return _capacity;
  }
};

template< long Bits >
class Item_alloc : public Item_alloc_base
{
private:
  typename Bitmap_base::Word<Bits>::Type _bits[Bitmap_base::Word<Bits>::Size];

public:
  Item_alloc() noexcept : Item_alloc_base(Bits, _bits) {}
};

}}
```

16.249 l4/re/util/kumem_alloc File Reference

Kumem allocator helper.

`#include <l4/re/env>`

Include dependency graph for kumem_alloc:

Namespaces

- L4Re

L4Re C++ Interfaces.

- L4Re::Util

  Documentation of the L4 Runtime Environment utility functionality in C++.

Functions

- int L4Re::Util::kumem_alloc (l4_addr_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) noexcept

  Allocate state area.

16.249.1 Detailed Description

Kumem allocator helper.

Definition in file kumem_alloc.

16.250 kumem_alloc

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *          Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025
00026 #pragma once
00027
00028 #include <l4/re/env>
00029
00030 namespace L4Re { namespace Util {
00031
00055 int
00056 kumem_alloc(l4_addr_t *mem, unsigned pages_order,
00057             L4::Cap<L4::Task> task = L4Re::Env::env()->task(),
00058             L4::Cap<L4Re::Rm> rm = L4Re::Env::env()->rm()) noexcept;
00059
00061 }}
```

16.251 l4/re/util/region_mapping File Reference

Region handling.

```
#include <l4/cxx/avl_map>
#include <l4/sys/types.h>
```

```
#include <l4/re/rm>
```

Include dependency graph for region_mapping:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4Re

  L4Re C++ Interfaces.
- L4Re::Util

  Documentation of the L4 Runtime Environment utility functionality in C++.

16.251.1 Detailed Description

Region handling.

Definition in file region_mapping.

16.252 region_mapping

```
// -*- Mode: C++ -*-
// vim:ft=cpp
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */

#pragma once

#include <l4/cxx/avl_map>
#include <l4/sys/types.h>
#include <l4/re/rm>


namespace L4Re { namespace Util {
class Region
{
private:
  l4_addr_t _start, _end;

public:
  Region() noexcept : _start(~0UL), _end(~0UL) {}
  Region(l4_addr_t addr) noexcept : _start(addr), _end(addr) {}
  Region(l4_addr_t start, l4_addr_t end) noexcept
    : _start(start), _end(end) {}
  l4_addr_t start() const noexcept { return _start; }
  l4_addr_t end() const noexcept { return _end; }
  unsigned long size() const noexcept { return end() - start() + 1; }
  bool invalid() const noexcept { return _start == ~0UL && _end == ~0UL; }
  bool operator < (Region const &o) const noexcept
  { return end() < o.start(); }
  bool contains(Region const &o) const noexcept
  { return o.start() >= start() && o.end() <= end(); }
  bool operator == (Region const &o) const noexcept
  { return o.start() == start() && o.end() == end(); }
  ~Region() noexcept {}
};

template< typename DS, typename OPS >
class Region_handler
{
private:
  L4Re::Rm::Offset _offs;
  DS _mem;
  l4_cap_idx_t _client_cap = L4_INVALID_CAP;
  L4Re::Rm::Region_flags _flags;

public:
  typedef DS Dataspace;
  typedef OPS Ops;
  typedef typename OPS::Map_result Map_result;

  Region_handler() noexcept : _offs(0), _mem(), _flags() {}
  Region_handler(Dataspace const &mem, l4_cap_idx_t client_cap,
      L4Re::Rm::Offset offset = 0,
      L4Re::Rm::Region_flags flags = L4Re::Rm::Region_flags(0)) noexcept
    : _offs(offset), _mem(mem), _client_cap(client_cap), _flags(flags)
  {}

  Dataspace const &memory() const noexcept
  {
    return _mem;
  }

  l4_cap_idx_t client_cap_idx() const noexcept
  {
    return _client_cap;
  }

  L4Re::Rm::Offset offset() const noexcept
```

```
  {
    return _offs;
  }

  constexpr bool is_ro() const noexcept
  {
    return !(_flags & L4Re::Rm::F::W);
  }

  L4Re::Rm::Region_flags caching() const noexcept
  {
    return _flags & L4Re::Rm::F::Caching_mask;
  }

  L4Re::Rm::Region_flags flags() const noexcept
  {
    return _flags;
  }

  Region_handler operator + (l4_int64_t offset) const noexcept
  {
    Region_handler n = *this; n._offs += offset; return n;
  }

  void free(l4_addr_t start, unsigned long size) const noexcept
  {
    Ops::free(this, start, size);
  }

  int map(l4_addr_t addr, Region const &r, bool writable,
          Map_result *result) const
  {
    return Ops::map(this, addr, r, writable, result);
  }

};


template< typename Hdlr, template<typename T> class Alloc >
class Region_map
{
protected:
  typedef cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc > Tree;
  Tree _rm;
  Tree _am;

private:
  l4_addr_t _start;
  l4_addr_t _end;

protected:
  void set_limits(l4_addr_t start, l4_addr_t end) noexcept
  {
    _start = start;
    _end = end;
  }

public:
  typedef typename Tree::Item_type  Item;
  typedef typename Tree::Node       Node;
  typedef typename Tree::Key_type   Key_type;
  typedef Hdlr Region_handler;

  typedef typename Tree::Iterator Iterator;
  typedef typename Tree::Const_iterator Const_iterator;
  typedef typename Tree::Rev_iterator Rev_iterator;
  typedef typename Tree::Const_rev_iterator Const_rev_iterator;

  Iterator begin() noexcept { return _rm.begin(); }
  Const_iterator begin() const noexcept { return _rm.begin(); }
  Iterator end() noexcept { return _rm.end(); }
  Const_iterator end() const noexcept { return _rm.end(); }

  Iterator area_begin() noexcept { return _am.begin(); }
  Const_iterator area_begin() const noexcept { return _am.begin(); }
  Iterator area_end() noexcept { return _am.end(); }
  Const_iterator area_end() const noexcept { return _am.end(); }
  Node area_find(Key_type const &c) const noexcept { return _am.find_node(c); }

  l4_addr_t min_addr() const noexcept { return _start; }
  l4_addr_t max_addr() const noexcept { return _end; }


  Region_map(l4_addr_t start, l4_addr_t end) noexcept : _start(start), _end(end) {}

  Node find(Key_type const &key) const noexcept
  {
```

```
  Node n = _rm.find_node(key);
  if (!n)
    return Node();

  // 'find' should find any region overlapping with the searched one, the
  // caller should check for further requirements
  if (0)
    if (!n->first.contains(key))
      return Node();

  return n;
}

Node lower_bound(Key_type const &key) const noexcept
{
  Node n = _rm.lower_bound_node(key);
  return n;
}

Node lower_bound_area(Key_type const &key) const noexcept
{
  Node n = _am.lower_bound_node(key);
  return n;
}

l4_addr_t attach_area(l4_addr_t addr, unsigned long size,
                      L4Re::Rm::Flags flags = L4Re::Rm::Flags(0),
                      unsigned char align = L4_PAGESHIFT) noexcept
{
  if (size < 2)
    return L4_INVALID_ADDR;


  Region c;

  if (!(flags & L4Re::Rm::F::Search_addr))
    {
c = Region(addr, addr + size - 1);
Node r = _am.find_node(c);
if (r)
  return L4_INVALID_ADDR;
    }

  while (flags &  L4Re::Rm::F::Search_addr)
    {
if (addr < min_addr() || (addr + size - 1) > max_addr())
  addr = min_addr();
addr = find_free(addr, max_addr(), size, align, flags);
if (addr == L4_INVALID_ADDR)
  return L4_INVALID_ADDR;

c = Region(addr, addr + size - 1);
Node r = _am.find_node(c);
if (!r)
  break;

if (r->first.end() >= max_addr())
  return L4_INVALID_ADDR;

addr = r->first.end() + 1;
    }

  if (_am.insert(c, Hdlr(typename Hdlr::Dataspace(), 0, 0, flags.region_flags())).second == 0)
    return addr;

  return L4_INVALID_ADDR;
}

bool detach_area(l4_addr_t addr) noexcept
{
  if (_am.remove(addr))
    return false;

  return true;
}

void *attach(void *addr, unsigned long size, Hdlr const &hdlr,
             L4Re::Rm::Flags flags = L4Re::Rm::Flags(0),
             unsigned char align = L4_PAGESHIFT) noexcept
{
  if (size < 2)
    return L4_INVALID_PTR;

  l4_addr_t end = max_addr();
  l4_addr_t beg = (l4_addr_t)addr;

  if (flags & L4Re::Rm::F::In_area)
```

```
    {
  Node r = _am.find_node(Region(beg, beg + size - 1));
  if (!r || (r->second.flags() & L4Re::Rm::F::Reserved))
    return L4_INVALID_PTR;

  end = r->first.end();
    }

    if (flags & L4Re::Rm::F::Search_addr)
      {
        beg = find_free(beg, end, size, align, flags);
        if (beg == L4_INVALID_ADDR)
          return L4_INVALID_PTR;
      }

    if (!(flags & (L4Re::Rm::F::Search_addr | L4Re::Rm::F::In_area))
        && _am.find_node(Region(beg, beg + size - 1)))
      return L4_INVALID_PTR;

    if (beg < min_addr() || beg + size -1 > end)
      return L4_INVALID_PTR;

    if (_rm.insert(Region(beg, beg + size -1), hdlr).second == 0)
      return (void*)beg;

    return L4_INVALID_PTR;
  }

  int detach(void *addr, unsigned long sz, unsigned flags,
             Region *reg, Hdlr *hdlr) noexcept
  {
    Region dr((l4_addr_t)addr, (l4_addr_t)addr + sz - 1);
    Region res(~0UL,0);

    Node r = find(dr);
    if (!r)
      return -L4_ENOENT;

    Region g = r->first;
    Hdlr const &h = r->second;

    if (flags & L4Re::Rm::Detach_overlap || dr.contains(g))
      {
  if (_rm.remove(g))
    return -L4_ENOENT;

  if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
    h.free(0, g.size());

  if (hdlr) *hdlr = h;
  if (reg) *reg = g;

  if (find(dr))
    return Rm::Detached_ds | Rm::Detach_again;
  else
    return Rm::Detached_ds;
      }
    else if (dr.start() <= g.start())
      {
  // move the start of a region

  if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
    h.free(0, dr.end() + 1 - g.start());

  unsigned long sz = dr.end() + 1 - g.start();
  Item *cn = const_cast<Item*>((Item const *)r);
  cn->first = Region(dr.end() + 1, g.end());
  cn->second = cn->second + sz;
  if (hdlr) *hdlr = Hdlr();
  if (reg) *reg = Region(g.start(), dr.end());
  if (find(dr))
    return Rm::Kept_ds | Rm::Detach_again;
  else
    return Rm::Kept_ds;
      }
    else if (dr.end() >= g.end())
      {
  // move the end of a region

  if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
    h.free(dr.start() - g.start(), g.end() + 1 - dr.start());

  Item *cn = const_cast<Item*>((Item const*)r);
  cn->first = Region(g.start(), dr.start() -1);
  if (hdlr) *hdlr = Hdlr();
  if (reg) *reg = Region(dr.start(), g.end());

```

```
  if (find(dr))
    return Rm::Kept_ds | Rm::Detach_again;
  else
    return Rm::Kept_ds;
      }
    else if (g.contains(dr))
      {
  // split a single region that contains the new region

  if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
    h.free(dr.start() - g.start(), dr.size());

  // first move the end off the existing region before the new one
  const_cast<Item*>((Item const *)r)->first = Region(g.start(), dr.start()-1);

  int err;

  // insert a second region for the remaining tail of
  // the old existing region
  err = _rm.insert(Region(dr.end() + 1, g.end()), h + (dr.end() + 1 - g.start())).second;

  if (err)
    return err;

  if (hdlr) *hdlr = h;
  if (reg) *reg = dr;
  return Rm::Split_ds;
      }
    return -L4_ENOENT;
  }

  l4_addr_t find_free(l4_addr_t start, l4_addr_t end, l4_addr_t size,
                      unsigned char align, L4Re::Rm::Flags flags) const noexcept;

};


template< typename Hdlr, template<typename T> class Alloc >
l4_addr_t
Region_map<Hdlr, Alloc>::find_free(l4_addr_t start, l4_addr_t end,
    unsigned long size, unsigned char align, L4Re::Rm::Flags flags) const noexcept
{
  l4_addr_t addr = start;

  if (addr == ~0UL || addr < min_addr() || addr >= end)
    addr = min_addr();

  addr = l4_round_size(addr, align);
  Node r;

  for(;;)
    {
      if (addr > 0 && addr - 1 > end - size)
  return L4_INVALID_ADDR;

      Region c(addr, addr + size - 1);
      r = _rm.find_node(c);

      if (!r)
  {
    if (!(flags & L4Re::Rm::F::In_area) && (r = _am.find_node(c)))
      {
        if (r->first.end() > end - size)
    return L4_INVALID_ADDR;

        addr = l4_round_size(r->first.end() + 1, align);
        continue;
      }
    break;
  }
      else if (r->first.end() > end - size)
  return L4_INVALID_ADDR;

      addr = l4_round_size(r->first.end() + 1, align);
    }

  if (!r)
    return addr;

  return L4_INVALID_ADDR;
}

}}
```

16.253 l4/re/video/goos-sys.h File Reference

Goos protocol definition.

This graph shows which files directly or indirectly include this file:

Namespaces

- L4Re

  L4Re C++ Interfaces.

Enumerations

- enum L4Re::Video::Goos_::Opcodes

  Frame buffer communication-protocol opcodes.

16.253.1 Detailed Description

Goos protocol definition.

Definition in file goos-sys.h.

16.254 goos-sys.h

```
#pragma once
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
```

```
 * the GNU General Public License.
 */

namespace L4Re { namespace Video {
  namespace Goos_
  {
    enum Opcodes
    {
      Info, Get_buffer, Create_buffer, Create_view,
      Delete_buffer, Delete_view,
      View_info, View_set_info, View_stack, View_refresh,
      Screen_refresh
    };
  };
}}
```

16.255 l4/shmc/shmc.h File Reference

Shared memory library header file.

```
#include <l4/sys/linkage.h>
#include <l4/sys/types.h>
#include <l4/sys/err.h>
#include <l4/shmc/types.h>
#include <l4/shmc/internal.h>
```

Include dependency graph for shmc.h:

Functions

- long l4shmc_create (const char *shmc_name, l4_umword_t shm_size)

  Create a shared memory area.
- long l4shmc_attach (const char *shmc_name, l4shmc_area_t *shmarea)

  Attach to a shared memory area.
- long l4shmc_attach_to (const char *shmc_name, l4_umword_t timeout_ms, l4shmc_area_t *shmarea)

  Attach to a shared memory area, with limited waiting.
- long l4shmc_add_chunk (l4shmc_area_t *shmarea, const char *chunk_name, l4_umword_t chunk_capacity, l4shmc_chunk_t *chunk)

  Add a chunk in the shared memory area.
- long l4shmc_add_signal (l4shmc_area_t *shmarea, const char *signal_name, l4shmc_signal_t *signal)

Add a signal for the shared memory area.

- long l4shmc_trigger (l4shmc_signal_t *signal)

  Trigger a signal.
- long l4shmc_chunk_try_to_take (l4shmc_chunk_t *chunk)

  Try to mark chunk busy.
- long l4shmc_chunk_ready (l4shmc_chunk_t *chunk, l4_umword_t size)

  Mark chunk as filled (ready).
- long l4shmc_chunk_ready_sig (l4shmc_chunk_t *chunk, l4_umword_t size)

  Mark chunk as filled (ready) and signal consumer.
- long l4shmc_get_chunk (l4shmc_area_t *shmarea, const char *chunk_name, l4shmc_chunk_t *chunk)

  Get chunk out of shared memory area.
- long l4shmc_get_chunk_to (l4shmc_area_t *shmarea, const char *chunk_name, l4_umword_t timeout_ms, l4shmc_chunk_t *chunk)

  Get chunk out of shared memory area, with timeout.
- long l4shmc_iterate_chunk (l4shmc_area_t *shmarea, const char **chunk_name, long offs)

  Iterate over names of all existing chunks.
- long l4shmc_attach_signal (l4shmc_area_t *shmarea, const char *signal_name, l4_cap_idx_t thread, l4shmc_signal_t *signal)

  Attach to signal.
- long l4shmc_attach_signal_to (l4shmc_area_t *shmarea, const char *signal_name, l4_cap_idx_t thread, l4_umword_t timeout_ms, l4shmc_signal_t *signal)

  Attach to signal, with timeout.
- long l4shmc_get_signal_to (l4shmc_area_t *shmarea, const char *signal_name, l4_umword_t timeout_ms, l4shmc_signal_t *signal)

  Get signal object from the shared memory area.
- long l4shmc_enable_signal (l4shmc_signal_t *signal)

  Enable a signal.
- long l4shmc_enable_chunk (l4shmc_chunk_t *chunk)

  Enable a signal connected with a chunk.
- long l4shmc_wait_any (l4shmc_signal_t **retsignal)

  Wait on any signal.
- long l4shmc_wait_any_try (l4shmc_signal_t **retsignal)

  Check whether any waited signal has an event pending.
- long l4shmc_wait_any_to (l4_timeout_t timeout, l4shmc_signal_t **retsignal)

  Wait for any signal with timeout.
- long l4shmc_wait_signal (l4shmc_signal_t *signal)

  Wait on a specific signal.
- long l4shmc_wait_signal_to (l4shmc_signal_t *signal, l4_timeout_t timeout)

  Wait on a specific signal, with timeout.
- long l4shmc_wait_signal_try (l4shmc_signal_t *signal)

  Check whether a specific signal has an event pending.
- long l4shmc_wait_chunk (l4shmc_chunk_t *chunk)

  Wait on a specific chunk.
- long l4shmc_wait_chunk_to (l4shmc_chunk_t *chunk, l4_timeout_t timeout)

  Check whether a specific chunk has an event pending, with timeout.
- long l4shmc_wait_chunk_try (l4shmc_chunk_t *chunk)

  Check whether a specific chunk has an event pending.
- long l4shmc_chunk_consumed (l4shmc_chunk_t *chunk)

  Mark a chunk as free.
- long l4shmc_connect_chunk_signal (l4shmc_chunk_t *chunk, l4shmc_signal_t *signal)

  Connect a signal with a chunk.

- long l4shmc_is_chunk_ready (l4shmc_chunk_t *chunk)

  Check whether data is available.
- long l4shmc_is_chunk_clear (l4shmc_chunk_t *chunk)

  Check whether chunk is free.
- void * l4shmc_chunk_ptr (l4shmc_chunk_t *chunk)

  Get data pointer to chunk.
- long l4shmc_chunk_size (l4shmc_chunk_t *chunk)

  Get current size of a chunk.
- long l4shmc_chunk_capacity (l4shmc_chunk_t *chunk)

  Get capacity of a chunk.
- l4shmc_signal_t * l4shmc_chunk_signal (l4shmc_chunk_t *chunk)

  Get the signal of a chunk.
- l4_cap_idx_t l4shmc_signal_cap (l4shmc_signal_t *signal)

  Get the signal capability of a signal.
- long l4shmc_check_magic (l4shmc_chunk_t *chunk)

  Check magic value of a chunk.
- long l4shmc_area_size (l4shmc_area_t *shmarea)

  Get size of shared memory area.
- long l4shmc_area_size_free (l4shmc_area_t *shmarea)

  Get free size of shared memory area.
- long l4shmc_area_overhead (void)

  Get memory overhead per area that is not available for chunks.
- long l4shmc_chunk_overhead (void)

  Get memory overhead required in addition to the chunk capacity for adding one chunk.

16.255.1 Detailed Description

Shared memory library header file.

Definition in file shmc.h.

16.256 shmc.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Björn Döbel <doebel@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/linkage.h>
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/err.h>
00018
00069 #define __INCLUDED_FROM_L4SHMC_H__
00070 #include <l4/shmc/types.h>
00071
00072 __BEGIN_DECLS
00073
00084 L4_CV long
00085 l4shmc_create(const char *shmc_name, l4_umword_t shm_size);
00086
00098 L4_CV L4_INLINE long
00099 l4shmc_attach(const char *shmc_name, l4shmc_area_t *shmarea);
00100
00113 L4_CV long
```

```
l4shmc_attach_to(const char *shmc_name, l4_umword_t timeout_ms,
                 l4shmc_area_t *shmarea);

L4_CV long
l4shmc_add_chunk(l4shmc_area_t *shmarea,
                 const char *chunk_name,
                 l4_umword_t chunk_capacity,
                 l4shmc_chunk_t *chunk);

L4_CV long
l4shmc_add_signal(l4shmc_area_t *shmarea,
                  const char *signal_name,
                  l4shmc_signal_t *signal);

L4_CV L4_INLINE long
l4shmc_trigger(l4shmc_signal_t *signal);

L4_CV L4_INLINE long
l4shmc_chunk_try_to_take(l4shmc_chunk_t *chunk);

L4_CV L4_INLINE long
l4shmc_chunk_ready(l4shmc_chunk_t *chunk, l4_umword_t size);

L4_CV L4_INLINE long
l4shmc_chunk_ready_sig(l4shmc_chunk_t *chunk, l4_umword_t size);

L4_CV L4_INLINE long
l4shmc_get_chunk(l4shmc_area_t *shmarea,
                 const char *chunk_name,
                 l4shmc_chunk_t *chunk);

L4_CV long
l4shmc_get_chunk_to(l4shmc_area_t *shmarea,
                    const char *chunk_name,
                    l4_umword_t timeout_ms,
                    l4shmc_chunk_t *chunk);

L4_CV long
l4shmc_iterate_chunk(l4shmc_area_t *shmarea, const char **chunk_name,
                     long offs);

L4_CV L4_INLINE long
l4shmc_attach_signal(l4shmc_area_t *shmarea,
                     const char *signal_name,
                     l4_cap_idx_t thread,
                     l4shmc_signal_t *signal);

L4_CV long
l4shmc_attach_signal_to(l4shmc_area_t *shmarea,
                        const char *signal_name,
                        l4_cap_idx_t thread,
                        l4_umword_t timeout_ms,
                        l4shmc_signal_t *signal);

L4_CV long
l4shmc_get_signal_to(l4shmc_area_t *shmarea,
                     const char *signal_name,
                     l4_umword_t timeout_ms,
                     l4shmc_signal_t *signal);

L4_CV L4_INLINE long
l4shmc_get_signal(l4shmc_area_t *shmarea,
                  const char *signal_name,
                  l4shmc_signal_t *signal);


L4_CV long
l4shmc_enable_signal(l4shmc_signal_t *signal);

L4_CV long
l4shmc_enable_chunk(l4shmc_chunk_t *chunk);

L4_CV L4_INLINE long
l4shmc_wait_any(l4shmc_signal_t **retsignal);

L4_CV L4_INLINE long
l4shmc_wait_any_try(l4shmc_signal_t **retsignal);

L4_CV long
l4shmc_wait_any_to(l4_timeout_t timeout, l4shmc_signal_t **retsignal);

L4_CV L4_INLINE long
l4shmc_wait_signal(l4shmc_signal_t *signal);

L4_CV long
l4shmc_wait_signal_to(l4shmc_signal_t *signal, l4_timeout_t timeout);
```

```
L4_CV L4_INLINE long
l4shmc_wait_signal_try(l4shmc_signal_t *signal);

L4_CV L4_INLINE long
l4shmc_wait_chunk(l4shmc_chunk_t *chunk);

L4_CV long
l4shmc_wait_chunk_to(l4shmc_chunk_t *chunk, l4_timeout_t timeout);

L4_CV L4_INLINE long
l4shmc_wait_chunk_try(l4shmc_chunk_t *chunk);

L4_CV L4_INLINE long
l4shmc_chunk_consumed(l4shmc_chunk_t *chunk);

L4_CV long
l4shmc_connect_chunk_signal(l4shmc_chunk_t *chunk,
                            l4shmc_signal_t *signal);

L4_CV L4_INLINE long
l4shmc_is_chunk_ready(l4shmc_chunk_t *chunk);

L4_CV L4_INLINE long
l4shmc_is_chunk_clear(l4shmc_chunk_t *chunk);

L4_CV L4_INLINE void *
l4shmc_chunk_ptr(l4shmc_chunk_t *chunk);

L4_CV L4_INLINE long
l4shmc_chunk_size(l4shmc_chunk_t *chunk);

L4_CV L4_INLINE long
l4shmc_chunk_capacity(l4shmc_chunk_t *chunk);

L4_CV L4_INLINE l4shmc_signal_t *
l4shmc_chunk_signal(l4shmc_chunk_t *chunk);

L4_CV L4_INLINE l4_cap_idx_t
l4shmc_signal_cap(l4shmc_signal_t *signal);

L4_CV L4_INLINE long
l4shmc_check_magic(l4shmc_chunk_t *chunk);

L4_CV L4_INLINE long
l4shmc_area_size(l4shmc_area_t *shmarea);

L4_CV long
l4shmc_area_size_free(l4shmc_area_t *shmarea);

L4_CV long
l4shmc_area_overhead(void);

L4_CV long
l4shmc_chunk_overhead(void);

#include <l4/shmc/internal.h>

__END_DECLS
```

16.257 l4/sigma0/sigma0.h File Reference

Sigma0 interface.

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>
```

Include dependency graph for sigma0.h:

Macros

- #define SIGMA0_REQ_MAGIC ~0xFFUL

  Request magic.
- #define SIGMA0_REQ_MASK ~0xFFUL

  Request mask.
- #define SIGMA0_REQ_ID_MASK 0xF0

  ID mask.
- #define SIGMA0_REQ_ID_FPAGE_RAM 0x60

  RAM.
- #define SIGMA0_REQ_ID_FPAGE_IOMEM 0x70

  I/O memory.
- #define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED 0x80

  Cached I/O memory.
- #define SIGMA0_REQ_ID_FPAGE_ANY 0x90

  Any.
- #define SIGMA0_REQ_ID_KIP 0xA0

  KIP.
- #define SIGMA0_REQ_ID_DEBUG_DUMP 0xC0

  Debug dump.
- #define SIGMA0_REQ_ID_NEW_CLIENT 0xD0

  New client.
- #define SIGMA0_IS_MAGIC_REQ(d1) ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)

  Check if magic.
- #define SIGMA0_REQ(x) (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## x)

  Construct.
- #define SIGMA0_REQ_FPAGE_RAM (SIGMA0_REQ(FPAGE_RAM))

  RAM.
- #define SIGMA0_REQ_FPAGE_IOMEM (SIGMA0_REQ(FPAGE_IOMEM))

I/O memory.

- #define SIGMA0_REQ_FPAGE_IOMEM_CACHED (SIGMA0_REQ(FPAGE_IOMEM_CACHED))

  Cache I/O memory.
- #define SIGMA0_REQ_FPAGE_ANY (SIGMA0_REQ(FPAGE_ANY))

  Any.
- #define SIGMA0_REQ_KIP (SIGMA0_REQ(KIP))

  KIP.
- #define SIGMA0_REQ_DEBUG_DUMP (SIGMA0_REQ(DEBUG_DUMP))

  Debug dump.
- #define SIGMA0_REQ_NEW_CLIENT (SIGMA0_REQ(NEW_CLIENT))

  New client.

Enumerations

- enum l4sigma0_return_flags_t {
  L4SIGMA0_OK , L4SIGMA0_NOTALIGNED , L4SIGMA0_IPCERROR , L4SIGMA0_NOFPAGE ,
  L4SIGMA0_4 , **L4SIGMA0_5** , L4SIGMA0_SMALLERFPAGE }

  Return flags of libsigma0 functions.

Functions

- l4_kernel_info_t ∗ l4sigma0_map_kip (l4_cap_idx_t sigma0, void ∗addr, unsigned log2_size)

  Map the kernel info page from sigma0 to addr.
- int l4sigma0_map_mem (l4_cap_idx_t sigma0, l4_addr_t phys, l4_addr_t virt, l4_addr_t size)

  Request a memory mapping from sigma0.
- int l4sigma0_map_iomem (l4_cap_idx_t sigma0, l4_addr_t phys, l4_addr_t virt, l4_addr_t size, int cached)

  Request IO memory from sigma0.
- int l4sigma0_map_anypage (l4_cap_idx_t sigma0, l4_addr_t map_area, unsigned log2_map_size, l4_addr_t ∗base, unsigned sz)

  Request an arbitrary free page of RAM.
- void l4sigma0_debug_dump (l4_cap_idx_t sigma0)

  Request sigma0 to dump internal debug information.
- int l4sigma0_new_client (l4_cap_idx_t sigma0, l4_cap_idx_t gate)

  Create a new IPC gate for a new Sigma0 client.
- char const ∗ l4sigma0_map_errstr (int err)

  Get user readable error messages for the return codes.

16.257.1 Detailed Description

Sigma0 interface.

Definition in file sigma0.h.

16.258 sigma0.h

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */
#ifndef __L4_SIGMA0_SIGMA0_H
#define __L4_SIGMA0_SIGMA0_H

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>

#undef SIGMA0_REQ_MAGIC
#undef SIGMA0_REQ_MASK

# define SIGMA0_REQ_MAGIC   ~0xFFUL
# define SIGMA0_REQ_MASK    ~0xFFUL
/* Starting with 0x60 allows to detect components which still use the old
 * constants (0x00 ... 0x50) */
#define SIGMA0_REQ_ID_MASK      0xF0
#define SIGMA0_REQ_ID_FPAGE_RAM     0x60
#define SIGMA0_REQ_ID_FPAGE_IOMEM   0x70
#define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED  0x80
#define SIGMA0_REQ_ID_FPAGE_ANY     0x90
#define SIGMA0_REQ_ID_KIP     0xA0
#define SIGMA0_REQ_ID_DEBUG_DUMP    0xC0
#define SIGMA0_REQ_ID_NEW_CLIENT    0xD0
#define SIGMA0_IS_MAGIC_REQ(d1) \
  ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)
#define SIGMA0_REQ(x) \
  (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## x)
/* Use these constants in your code! */
#define SIGMA0_REQ_FPAGE_RAM            (SIGMA0_REQ(FPAGE_RAM))
#define SIGMA0_REQ_FPAGE_IOMEM          (SIGMA0_REQ(FPAGE_IOMEM))
#define SIGMA0_REQ_FPAGE_IOMEM_CACHED   (SIGMA0_REQ(FPAGE_IOMEM_CACHED))
#define SIGMA0_REQ_FPAGE_ANY            (SIGMA0_REQ(FPAGE_ANY))
#define SIGMA0_REQ_KIP                  (SIGMA0_REQ(KIP))
#define SIGMA0_REQ_DEBUG_DUMP           (SIGMA0_REQ(DEBUG_DUMP))
#define SIGMA0_REQ_NEW_CLIENT           (SIGMA0_REQ(NEW_CLIENT))
enum l4sigma0_return_flags_t {
  L4SIGMA0_OK,
  L4SIGMA0_NOTALIGNED,
  L4SIGMA0_IPCERROR,
  L4SIGMA0_NOFPAGE,
  L4SIGMA0_4,
  L4SIGMA0_5,
  L4SIGMA0_SMALLERFPAGE,
};

EXTERN_C_BEGIN

L4_CV l4_kernel_info_t *
l4sigma0_map_kip(l4_cap_idx_t sigma0, void *addr, unsigned log2_size);

L4_CV int l4sigma0_map_mem(l4_cap_idx_t sigma0,
                           l4_addr_t phys, l4_addr_t virt, l4_addr_t size);

L4_CV int l4sigma0_map_iomem(l4_cap_idx_t sigma0, l4_addr_t phys,
                             l4_addr_t virt, l4_addr_t size, int cached);
L4_CV int l4sigma0_map_anypage(l4_cap_idx_t sigma0, l4_addr_t map_area,
                               unsigned log2_map_size, l4_addr_t *base,
                               unsigned sz);

L4_CV void l4sigma0_debug_dump(l4_cap_idx_t sigma0);

L4_CV int l4sigma0_new_client(l4_cap_idx_t sigma0, l4_cap_idx_t gate);

L4_INLINE char const *l4sigma0_map_errstr(int err);

/* Implementations */

L4_INLINE char const *l4sigma0_map_errstr(int err)
{
  switch (err)
    {
    case  0: return "No error";
    case -1: return "Phys, virt or size not aligned";
    case -2: return "IPC error";
    case -3: return "No fpage received";
#ifndef SIGMA0_REQ_MAGIC
```

```
    case -4: return "Bad physical address (old protocol only)";
#endif
    case -6: return "Superpage requested but smaller flexpage received";
    case -7: return "Cannot map I/O memory cacheable (old protocol only)";
    default: return "Unknown error";
    }
}


EXTERN_C_END

#endif /* ! __L4_SIGMA0_SIGMA0_H */
```

16.259 l4/sys/__kernel_object_impl.h File Reference

Low-level kernel debugger functions.

`#include <l4/sys/ipc.h>`

Include dependency graph for __kernel_object_impl.h:

This graph shows which files directly or indirectly include this file:

16.259.1 Detailed Description

Low-level kernel debugger functions.

Definition in file __kernel_object_impl.h.

16.260 __kernel_object_impl.h

```
#pragma once

#include <l4/sys/ipc.h>

L4_INLINE l4_msgtag_t
l4_invoke_debugger(l4_cap_idx_t obj, l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msgtag_t t2;
  unsigned const words = l4_msgtag_words(tag);
  l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);

  if (l4_is_invalid_cap(obj))
    return l4_msgtag(-L4_EINVAL, 0, 0, 0);

  if (words + 2 > L4_UTCB_GENERIC_DATA_SIZE)
    return l4_msgtag(-L4_EMSGTOOLONG, 0, 0, 0);

  mr->mr[0] += 0x100;
  mr->mr[words]     = L4_ITEM_MAP;
  mr->mr[words + 1] = l4_obj_fpage(obj, 0, L4_CAP_FPAGE_RWS).raw;
  t2 = l4_msgtag(L4_PROTO_DEBUGGER, words, 1, l4_msgtag_flags(tag));

  return l4_ipc_call(L4_BASE_DEBUGGER_CAP, utcb, t2, L4_IPC_NEVER);
}
```

16.261 l4/sys/__ktrace-impl.h File Reference

L4 kernel event tracing.

```
#include <l4/sys/types.h>
#include <l4/sys/kdebug.h>
```

Include dependency graph for __ktrace-impl.h:

This graph shows which files directly or indirectly include this file:

Functions

- l4_umword_t fiasco_tbuf_log (const char *text)

  Create new trace-buffer entry with describing <text>.
- l4_umword_t fiasco_tbuf_log_3val (const char *text, l4_umword_t v1, l4_umword_t v2, l4_umword_t v3)

  Create new trace-buffer entry with describing <text> and three additional values.
- void fiasco_tbuf_clear (void)

  Clear trace-buffer.
- void fiasco_tbuf_dump (void)

  Dump trace-buffer to kernel console.
- l4_umword_t fiasco_tbuf_log_binary (const unsigned char *data)

  Create new trace-buffer entry with binary data.

16.261.1 Detailed Description

L4 kernel event tracing.

Definition in file __ktrace-impl.h.

16.262 __ktrace-impl.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
```

```
#include <l4/sys/types.h>
#include <l4/sys/kdebug.h>

/*****************************************************************************
 *** Implementation
 *****************************************************************************/

L4_INLINE l4_umword_t
fiasco_tbuf_log(const char *text)
{
  enum { TBUF_LOG = 0x201 };
  return l4_error(__kdebug_text(TBUF_LOG, text, __builtin_strlen(text)));
}

L4_INLINE l4_umword_t
fiasco_tbuf_log_3val(const char *text, l4_umword_t v1, l4_umword_t v2,
                     l4_umword_t v3)
{
  enum { TBUF_LOG_3VAL = 0x204 };
  return l4_error(__kdebug_3_text(TBUF_LOG_3VAL, text,
                                  __builtin_strlen(text), v1, v2, v3));
}

L4_INLINE void
fiasco_tbuf_clear(void)
{
  enum { TBUF_CLEAR = 0x202 };
  __kdebug_op(TBUF_CLEAR);
}

L4_INLINE void
fiasco_tbuf_dump(void)
{
  enum { TBUF_DUMP = 0x203 };
  __kdebug_op(TBUF_DUMP);
}

L4_INLINE l4_umword_t
fiasco_tbuf_log_binary(const unsigned char *data)
{
  enum { TBUF_LOG_BIN = 0x208 };
  return l4_error(__kdebug_text(TBUF_LOG_BIN, (const char *)data, 24));
}
```

16.263 l4/sys/__typeinfo.h File Reference

Type information handling.

```
#include "cxx/types"
#include "cxx/ipc_basics"
#include "cxx/capability.h"
```

Include dependency graph for __typeinfo.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct L4::Typeid::P_dispatch< LIST >

  Use for protocol based dispatch stage.
- struct L4::Typeid::Detail::Rpcs_end

  Internal end-of-list marker.
- struct L4::Typeid::Detail::_Rpcs< OPCODE, O, X >

  Empty list of RPCs.
- struct L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >

  Non-empty list of RPCs.
- struct L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y >

  Find the given RPC in the list.
- struct L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y >

  Find the given RPC in the list.
- struct L4::Typeid::Raw_ipc< CLASS >

  RPCs list for passing raw incoming IPC to the server object.
- struct L4::Typeid::Rpcs< RPCS >

Standard list of RPCs of an interface.

- struct L4::Typeid::Rpcs_code< OPCODE_TYPE >

  List of RPCs of an interface using a special opcode type.
- struct L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS >
- struct L4::Typeid::Rpc_nocode< OPERATION >

  List of RPCs of an interface using a single operation without an opcode.
- struct L4::Typeid::Rpcs_sys< ARG >

  List of RPCs typically used for kernel interfaces.
- struct L4::Type_info

  Dynamic Type Information for L4Re Interfaces.
- class L4::Type_info::Demand

  Data type for expressing the needed receive buffers at the server-side of an interface.
- struct L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >

  Template type statically describing demand of receive buffers.
- struct L4::Type_info::Demand_union_t< D1, D2 >

  Template type statically describing the combination of two Demand object.
- struct L4::Kobject_typeid< T >

  Meta object for handling access to type information of Kobjects.
- struct L4::Kobject_typeid< void >

  Minimalistic ID for `void` interface.
- class L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >

  Helper class to create an L4Re interface class that is derived from a single base class.
- class L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >

  Helper class to create an L4Re interface class that is derived from two base classes (see `L4::Kobject_t`).
- struct L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >

  Helper class to create an L4Re interface class that is derived from three base classes (see `L4::Kobject_t`).
- struct L4::Proto_t< P >

  Data type for defining protocol numbers.

Namespaces

- L4

  L4 low-level kernel interface.
- L4::Typeid

  Definition of interface data-type helpers.

Typedefs

- typedef int L4::Opcode

  Data type for RPC opcodes.

Enumerations

- enum { L4::PROTO_ANY = 0 , L4::PROTO_EMPTY = -19 }

Functions

- template<typename T >
  Type_info const * L4::kobject_typeid () noexcept

  Get the L4::Type_info for the L4Re interface given in `T`.

16.263.1 Detailed Description

Type information handling.

Definition in file __typeinfo.h.

16.264 __typeinfo.h

```
00001
00005 /*
00006  * Copyright (C) 2015 Kernkonzept GmbH.
00007  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00008  */
00009 /*
00010  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction.  Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License.  This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027 #pragma GCC system_header
00028
00029 #include "cxx/types"
00030 #include "cxx/ipc_basics"
00031 #include "cxx/capability.h"
00032
00033 #if defined(__GXX_RTTI) && !defined(L4_NO_RTTI)
00034 #  include <typeinfo>
00035    typedef std::type_info const *L4_std_type_info_ptr;
00036 #  define L4_KOBJECT_META_RTTI(type) (&typeid(type))
00037    inline char const *L4_kobject_type_name(L4_std_type_info_ptr n) noexcept
00038    { return n ? n->name() : 0; }
00039 #else
00040    typedef void const *L4_std_type_info_ptr;
00041 #  define L4_KOBJECT_META_RTTI(type) (0)
00042    inline char const *L4_kobject_type_name(L4_std_type_info_ptr) noexcept
00043    { return 0; }
00044 #endif
00045
00046 namespace L4 {
00047   typedef int Opcode;
00048 // internal max helpers
00049 namespace __I {
00050   // internal max of A nd B helper
00051   template< unsigned char A, unsigned char B>
00052   struct Max { enum { Res = A > B ? A : B }; };
00053 } // namespace __I
00054
00055 enum
00056 {
00058   PROTO_ANY   = 0,
00060   PROTO_EMPTY = -19,
00061 };
00062
00077 namespace Typeid {
00083     using namespace L4::Types;
00084
00085   /*********************/
00093   template<long P, typename T>
00094   struct Iface
00095   {
00096     typedef Iface type;
00097     typedef T iface_type;
00098     enum { Proto = P };
00099   };
00100
00101
00102   /*********************/
00108   struct Iface_list_end
```

```
{
  typedef Iface_list_end type;
  static bool contains(long) noexcept { return false; }
};


template<typename I, typename N = Iface_list_end>
struct Iface_list
{
  typedef Iface_list<I, N> type;

  typedef typename I::iface_type iface_type;
  typedef N Next;

  enum { Proto = I::Proto };

  static bool contains(long proto) noexcept
  { return (proto == Proto) || Next::contains(proto); }
};

// do not insert PROTO_EMPTY interfaces
template<typename I, typename N>
struct Iface_list<Iface<PROTO_EMPTY, I>, N> : N {};

// do not insert 'void' type interfaces
template<long P, typename N>
struct Iface_list<Iface<P, void>, N> : N {};


/*********************/
/*
 * \internal
 * Test if an interface I is in list L
 * \tparam I  Interface for lookup
 * \tparam L  Iface_list for search
 */
template< typename I, typename L >
struct _In_list;

template< typename I >
struct _In_list<I, Iface_list_end> : False {};

template< typename I, typename N >
struct _In_list<I, Iface_list<I, N> > : True {};

template< typename I, typename I2, typename N >
struct _In_list<I, Iface_list<I2, N> > : _In_list<I, typename N::type> {};

template<typename I, typename L>
struct In_list : _In_list<typename I::type, typename L::type> {};


/************/
/*
 * \internal
 * Add Helper: add I to interface list L if ADD is true
 * \ingroup l4_cxx_ipc_internal
 */
template< bool ADD, typename I, typename L>
struct _Iface_list_add;

template< typename I, typename L>
struct _Iface_list_add<false, I, L> : L {};

template< typename I, typename L>
struct _Iface_list_add<true, I, L> : Iface_list<I, L> {};

/*
 * \internal
 * Add Helper: add I to interface list L if not already in L.
 * \ingroup l4_cxx_ipc_internal
 */
template< typename I, typename L >
struct Iface_list_add :
  _Iface_list_add<
    !In_list<I, typename L::type>::value, I, typename L::type>
{};

/************/
/*
 * \internal
 * Helper: checking for a conflict between I2 and I2.
 * A conflict means I1 and I2 have the same protocol ID but a different
 * iface_type.
 */
template< typename I1, typename I2 >
struct __Iface_conflict : Bool<I1::Proto != PROTO_EMPTY && I1::Proto == I2::Proto> {};
```

```
template< typename I >
struct __Iface_conflict<I, I> : False {};

/*
 * \internal
 * Helper: checking for a conflict between I and any interface in LIST.
 */
template< typename I, typename LIST >
struct _Iface_conflict;

template< typename I >
struct _Iface_conflict<I, Iface_list_end> : False {};

template< typename I, typename I2, typename LIST >
struct _Iface_conflict<I, Iface_list<I2, LIST> > :
  Bool<__Iface_conflict<I, I2>::value || _Iface_conflict<I, typename LIST::type>::value>
{};

template< typename I, typename LIST >
struct Iface_conflict : _Iface_conflict<typename I::type, typename LIST::type> {};

/**************/
/*
 * \internal
 * Helper: merge two interface lists
 */
template< typename L1, typename L2 >
struct _Merge_list;

template< typename L >
struct _Merge_list<Iface_list_end, L> : L {};

template< typename I, typename L1, typename L2 >
struct _Merge_list<Iface_list<I, L1>, L2> :
  _Merge_list<typename L1::type, typename Iface_list_add<I, L2>::type> {};

template<typename L1, typename L2>
struct Merge_list : _Merge_list<typename L1::type, typename L2::type> {};

/**************/
/*
 * \internal
 * check for conflicts among all interfaces in L1 with any interfaces in L2.
 */
template< typename L1, typename L2 >
struct _Conflict;

template< typename L >
struct _Conflict<Iface_list_end, L> : False {};

template< typename I, typename L1, typename L2 >
struct _Conflict<Iface_list<I, L1>, L2> :
  Bool<Iface_conflict<I, typename L2::type>::value
         || _Conflict<typename L1::type, typename L2::type>::value> {};

template< typename L1, typename L2 >
struct Conflict : _Conflict<typename L1::type, typename L2::type> {};

// to be removed ---------------------------------------
// p_dispatch code -- for legacy dispatch ------------------------------
/**********************/
/*
 * \internal
 * helper: Dispatch helper for calling server-side p_dispatch() functions.
 */
template<typename LIST>
struct _P_dispatch;

// No matching dispatcher found
template<>
struct _P_dispatch<Iface_list_end>
{
  template< typename THIS, typename A1, typename A2 >
  static int f(THIS *, long, A1, A2 &) noexcept
  { return -L4_EBADPROTO; }
};


// call matching p_dispatch() function
template< typename I, typename LIST >
struct _P_dispatch<Iface_list<I, LIST> >
{
  // special handling for the meta protocol, to avoid 'using' murx
  template< typename THIS, typename A1, typename A2 >
  static int _f(THIS self, A1, A2 &a2, True::type)
  {
```

```
      return self->dispatch_meta_request(a2);
    }

    // normal p_dispatch() dispatching
    template< typename THIS, typename A1, typename A2 >
    static int _f(THIS self, A1 a1, A2 &a2, False::type)
    {
      return self->p_dispatch(reinterpret_cast<typename I::iface_type *>(0),
                              a1, a2);
    }

    // dispatch function with switch for meta protocol
    template< typename THIS, typename A1, typename A2 >
    static int f(THIS *self, long proto, A1 a1, A2 &a2)
    {
      if (I::Proto == proto)
        return _f(self, a1, a2, Bool<I::Proto == (long)L4_PROTO_META>());

      return _P_dispatch<typename LIST::type>::f(self, proto, a1, a2);
    }
  };

  template<typename LIST>
  struct P_dispatch : _P_dispatch<typename LIST::type> {};
  // end: p_dispatch -------------------------------------------------------
  // end: to be removed ---------------------------------------

  template<typename RPC> struct Default_op;

  namespace Detail {

  struct Rpcs_end
  {
    typedef void opcode_type;
    typedef Rpcs_end rpc;
    typedef Rpcs_end type;
  };

  template<typename O1, typename O2, typename RPCS>
  struct _Rpc : _Rpc<typename RPCS::next::rpc, O2, typename RPCS::next>::type {};

  template<typename O1, typename O2>
  struct _Rpc<O1, O2, Rpcs_end> {};

  template<typename OP, typename RPCS>
  struct _Rpc<OP, OP, RPCS> : RPCS
  {
    typedef _Rpc type;
  };

  template<typename OP, typename RPCS>
  struct Rpc : _Rpc<typename RPCS::rpc, OP, RPCS> {};

  template<typename T, unsigned CODE>
  struct _Get_opcode
  {
    template<bool, typename> struct Invalid_opcode {};
    template<typename X> struct Invalid_opcode<true, X>;

  private:
    template<typename U, U> struct _chk;
    template<typename U> static long _opc(_chk<int, U::Opcode> *);
    template<typename U> static char _opc(...);

    template<unsigned SZ, typename U>
    struct _Opc { enum { value = CODE }; };

    template<typename U>
    struct _Opc<sizeof(long), U> { enum { value = U::Opcode }; };

  public:
    enum { value = _Opc<sizeof(_opc<T>(0)), T>::value };
    Invalid_opcode<(value < CODE), T> invalid_opcode;
  };

  template<typename OPCODE, unsigned O, typename ...X>
  struct _Rpcs : Rpcs_end {};

  template<typename OPCODE, unsigned O, typename R, typename ...X>
  struct _Rpcs<OPCODE, O, R, X...>
  {
    typedef _Rpcs type;
    typedef OPCODE opcode_type;
    typedef R rpc;
    typedef typename _Rpcs<OPCODE, _Get_opcode<R, O>::value + 1, X...>::type next;
    enum { Opcode = _Get_opcode<R, O>::value };
    template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpcs> {};
```

```
  };

  template<typename OPCODE, unsigned O, typename R>
  struct _Rpcs<OPCODE, O, Default_op<R> >
  {
    typedef _Rpcs type;
    typedef void opcode_type;
    typedef R rpc;
    typedef Rpcs_end next;
    enum { Opcode = -99 };
    template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpcs> {};
  };

  } // namespace Detail

  template<typename CLASS>
  struct Raw_ipc
  {
    typedef Raw_ipc type;
    typedef Detail::Rpcs_end next;
    typedef void opcode_type;
  };

  template<typename ...RPCS>
  struct Rpcs : Detail::_Rpcs<L4::Opcode, 0, RPCS...> {};

  template<typename OPCODE_TYPE>
  struct Rpcs_code
  {
    template<typename ...RPCS>
    struct F : Detail::_Rpcs<OPCODE_TYPE, 0, RPCS...> {};
  };

  template<typename OPERATION>
  struct Rpc_nocode : Detail::_Rpcs<void, 0, OPERATION> {};

  template<typename ...ARG>
  struct Rpcs_sys : Detail::_Rpcs<l4_umword_t, 0, ARG...> {};

  template<typename CLASS>
  struct Rights
  {
    unsigned rights;
    Rights(unsigned rights) noexcept : rights(rights) {}
    unsigned operator & (unsigned rhs) const noexcept { return rights & rhs; }
  };

} // namespace Typeid

struct L4_EXPORT Type_info
{
  class L4_EXPORT Demand
  {
  private:
    static unsigned char max(unsigned char a, unsigned char b) noexcept
    { return a > b ? a : b; }

  public:
    unsigned char caps;
    unsigned char flags;
    unsigned char mem;
    unsigned char ports;

    explicit
    Demand(unsigned char caps = 0, unsigned char flags = 0,
           unsigned char mem = 0,  unsigned char ports = 0) noexcept
    : caps(caps), flags(flags), mem(mem), ports(ports) {}

    bool no_demand() const noexcept
    { return caps == 0 && mem == 0 && ports == 0 && flags == 0; }

    Demand operator | (Demand const &rhs) const noexcept
    {
      return Demand(max(caps, rhs.caps), flags | rhs.flags,
                    max(mem, rhs.mem), max(ports, rhs.ports));
    }
  };

  template<unsigned char CAPS = 0, unsigned char FLAGS = 0,
           unsigned char MEM  = 0, unsigned char PORTS = 0>
  struct Demand_t : Demand
  {
    enum
    {
      Caps  = CAPS,
      Flags = FLAGS,
      Mem   = MEM,
```

```
      Ports = PORTS
    };
    Demand_t() noexcept : Demand(CAPS, FLAGS, MEM, PORTS) {}
  };

  template<typename D1, typename D2>
  struct Demand_union_t : Demand_t<__I::Max<D1::Caps,  D2::Caps>::Res,
                                   D1::Flags | D2::Flags,
                                   __I::Max<D1::Mem,   D2::Mem>::Res,
                                   __I::Max<D1::Ports, D2::Ports>::Res>
  {};

  L4_std_type_info_ptr _type;
  Type_info const *const *_bases;
  unsigned _num_bases;
  long _proto;

  L4_std_type_info_ptr type() const noexcept { return _type; }
  Type_info const *base(unsigned idx) const noexcept { return _bases[idx]; }
  unsigned num_bases() const noexcept { return _num_bases; }
  long proto() const noexcept { return _proto; }
  char const *name() const noexcept { return L4_kobject_type_name(type()); }
  bool has_proto(long proto) const noexcept
  {
    if (_proto && _proto == proto)
      return true;

    if (!proto)
      return false;

    for (unsigned i = 0; i < _num_bases; ++i)
      if (base(i)->has_proto(proto))
        return true;

    return false;
  }
};

template<typename T> struct Kobject_typeid
{
  typedef typename T::__Kobject_typeid::Demand Demand;
  typedef typename T::__Iface::iface_type Iface;
  typedef typename T::__Iface_list Iface_list;

  static Type_info const *id() noexcept { return &T::__Kobject_typeid::_m; }

  static Type_info::Demand demand() noexcept
  { return T::__Kobject_typeid::Demand(); }

  // to be removed ---------------------------------------
  // p_dispatch -------------------------------------------------------------
  template<typename THIS, typename A1, typename A2>
  static int proto_dispatch(THIS *self, long proto, A1 a1, A2 &a2)
  { return Typeid::P_dispatch<typename T::__Iface_list>::f(self, proto, a1, a2);  }
  // p_dispatch -------------------------------------------------------------
  // end: to be removed ----------------------------------------
};

template<> struct Kobject_typeid<void>
{
  typedef Type_info::Demand_t<> Demand;
};

template<typename T>
inline
Type_info const *kobject_typeid() noexcept
{ return Kobject_typeid<T>::id(); }

#define L4____GEN_TI(t...)                                                  \
Type_info const t::__Kobject_typeid::_m =                                   \
{                                                                           \
  L4_KOBJECT_META_RTTI(Derived),                                            \
  &t::__Kobject_typeid::_b[0],                                              \
  sizeof(t::__Kobject_typeid::_b) / sizeof(t::__Kobject_typeid::_b[0]),     \
  PROTO                                                                     \
}

#define L4____GEN_TI_MEMBERS(BASE_DEMAND...)                                \
private:                                                                    \
  template< typename T > friend struct Kobject_typeid;                      \
protected:                                                                  \
  struct __Kobject_typeid {                                                 \
    typedef Type_info::Demand_union_t<S_DEMAND, BASE_DEMAND> Demand;        \
    static Type_info const *const _b[];                                     \
    static Type_info const _m;                                              \
  };                                                                        \
public:                                                                     \
```

```
  static long const Protocol = PROTO;                                    \
  typedef L4::Typeid::Rights<Class> Rights;

template<
  typename Derived,
  typename Base,
  long PROTO = PROTO_ANY,
  typename S_DEMAND = Type_info::Demand_t<>
>
class Kobject_t : public Base
{
protected:
  typedef Derived Class;
  typedef Typeid::Iface<PROTO, Derived> __Iface;
  typedef Typeid::Merge_list<
    Typeid::Iface_list<__Iface>, typename Base::__Iface_list
  > __Iface_list;

  static void __check_protocols__() noexcept
  {
    typedef Typeid::Iface_conflict<__Iface, typename Base::__Iface_list> Base_conflict;
    static_assert(!Base_conflict::value, "ambiguous protocol ID: protocol also used by Base");
  }

  L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }

  // Generate the remaining type information
  L4____GEN_TI_MEMBERS(typename Base::__Kobject_typeid::Demand)
};


template< typename Derived, typename Base, long PROTO, typename S_DEMAND>
Type_info const *const
Kobject_t<Derived, Base, PROTO, S_DEMAND>::
  __Kobject_typeid::_b[] = { &Base::__Kobject_typeid::_m };

template< typename Derived, typename Base, long PROTO, typename S_DEMAND>
L4____GEN_TI(Kobject_t<Derived, Base, PROTO, S_DEMAND>);


template<
  typename Derived,
  typename Base1,
  typename Base2,
  long PROTO = PROTO_ANY,
  typename S_DEMAND = Type_info::Demand_t<>
>
class Kobject_2t : public Base1, public Base2
{
protected:
  typedef Derived Class;
  typedef Typeid::Iface<PROTO, Derived> __Iface;
  typedef Typeid::Merge_list<
    Typeid::Iface_list<__Iface>,
    Typeid::Merge_list<
      typename Base1::__Iface_list,
      typename Base2::__Iface_list
    >
  > __Iface_list;

  static void __check_protocols__() noexcept
  {
    typedef typename Base1::__Iface_list Base1_proto_list;
    typedef typename Base2::__Iface_list Base2_proto_list;

    typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
    typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
    static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
    static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");

    typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Bases_conflict;
    static_assert(!Bases_conflict::value, "ambiguous protocol IDs in base classes");
  }

  // disambiguate cap()
  l4_cap_idx_t cap() const noexcept
  { return Base1::cap(); }

  L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }

  L4____GEN_TI_MEMBERS(Type_info::Demand_union_t<
    typename Base1::__Kobject_typeid::Demand,
    typename Base2::__Kobject_typeid::Demand>
  )

public:
  // Provide non-ambiguous conversion to Kobject
```

```
  operator Kobject const & () const noexcept
  { return *static_cast<Base1 const *>(this); }

  // Provide non-ambiguous access of dec_refcnt()
  l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
    noexcept(noexcept(((Base1*)0)->dec_refcnt(diff, utcb)))
  { return Base1::dec_refcnt(diff, utcb); }
};


template< typename Derived, typename Base1, typename Base2,
          long PROTO, typename S_DEMAND >
Type_info const *const
Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>::__Kobject_typeid::_b[] =
{
  &Base1::__Kobject_typeid::_m,
  &Base2::__Kobject_typeid::_m
};

template< typename Derived, typename Base1, typename Base2,
          long PROTO, typename S_DEMAND >
L4____GEN_TI(Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>);



template<
  typename Derived,
  typename Base1,
  typename Base2,
  typename Base3,
  long PROTO = PROTO_ANY,
  typename S_DEMAND = Type_info::Demand_t<>
>
struct Kobject_3t : Base1, Base2, Base3
{
protected:
  typedef Derived Class;
  typedef Typeid::Iface<PROTO, Derived> __Iface;
  typedef Typeid::Merge_list<
    Typeid::Iface_list<__Iface>,
    Typeid::Merge_list<
      typename Base1::__Iface_list,
      Typeid::Merge_list<
        typename Base2::__Iface_list,
        typename Base3::__Iface_list
      >
    >
  > __Iface_list;

  static void __check_protocols__() noexcept
  {
    typedef typename Base1::__Iface_list Base1_proto_list;
    typedef typename Base2::__Iface_list Base2_proto_list;
    typedef typename Base3::__Iface_list Base3_proto_list;

    typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
    typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
    typedef Typeid::Iface_conflict<__Iface, Base3_proto_list> Base3_conflict;

    static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
    static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
    static_assert(!Base3_conflict::value, "ambiguous protocol ID, also in Base3");

    typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Conflict_bases12;
    typedef Typeid::Conflict<Base1_proto_list, Base3_proto_list> Conflict_bases13;
    typedef Typeid::Conflict<Base2_proto_list, Base3_proto_list> Conflict_bases23;

    static_assert(!Conflict_bases12::value, "ambiguous protocol IDs in base classes: Base1 and Base2");
    static_assert(!Conflict_bases13::value, "ambiguous protocol IDs in base classes: Base1 and Base3");
    static_assert(!Conflict_bases23::value, "ambiguous protocol IDs in base classes: Base2 and Base3");
  }

  // disambiguate cap()
  l4_cap_idx_t cap() const noexcept
  { return Base1::cap(); }

  L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }

  L4____GEN_TI_MEMBERS(Type_info::Demand_union_t<Type_info::Demand_union_t<
    typename Base1::__Kobject_typeid::Demand,
    typename Base2::__Kobject_typeid::Demand>,
    typename Base3::__Kobject_typeid::Demand>
  )

```

```
public:
  // Provide non-ambiguous conversion to Kobject
  operator Kobject const & () const noexcept
  { return *static_cast<Base1 const *>(this); }

  // Provide non-ambiguous access of dec_refcnt()
  l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
    noexcept(noexcept(((Base1*)0)->dec_refcnt(diff, utcb)))
  { return Base1::dec_refcnt(diff, utcb); }
};


template< typename Derived, typename Base1, typename Base2, typename Base3,
          long PROTO, typename S_DEMAND >
Type_info const *const
Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>::__Kobject_typeid::_b[] =
{
  &Base1::__Kobject_typeid::_m,
  &Base2::__Kobject_typeid::_m,
  &Base3::__Kobject_typeid::_m
};

template< typename Derived, typename Base1, typename Base2, typename Base3,
          long PROTO, typename S_DEMAND >
L4____GEN_TI(Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>);

}

#if __cplusplus >= 201103L

namespace L4 {

template< typename ...T >
struct Kobject_demand;

template<>
struct Kobject_demand<> : Type_info::Demand_t<> {};

template<typename T>
struct Kobject_demand<T> : Kobject_typeid<T>::Demand {};

template<typename T1, typename ...T2>
struct Kobject_demand<T1, T2...> :
  Type_info::Demand_union_t<typename Kobject_typeid<T1>::Demand,
                            Kobject_demand<T2...> >
{};

namespace Typeid_xx {

  template<typename ...LISTS>
  struct Merge_list;

  template<typename L>
  struct Merge_list<L> : L {};

  template<typename L1, typename L2>
  struct Merge_list<L1, L2> : Typeid::Merge_list<L1, L2> {};

  template<typename L1, typename L2, typename ...LISTS>
  struct Merge_list<L1, L2, LISTS...> :
    Merge_list<typename Typeid::Merge_list<L1, L2>::type, LISTS...> {};

  template< typename I, typename ...LIST >
  struct Iface_conflict;

  template< typename I >
  struct Iface_conflict<I> : Typeid::False {};

  template< typename I, typename L, typename ...LIST >
  struct Iface_conflict<I, L, LIST...> :
    Typeid::Bool<Typeid::Iface_conflict<typename I::type, typename L::type>::value
                 || Iface_conflict<I, LIST...>::value>
  {};

  template< typename ...LIST >
  struct Conflict;

  template< typename L >
  struct Conflict<L> : Typeid::False {};

  template< typename L1, typename L2, typename ...LIST >
  struct Conflict<L1, L2, LIST...> :
    Typeid::Bool<Typeid::Conflict<typename L1::type, typename L2::type>::value
                 || Conflict<L1, LIST...>::value
                 || Conflict<L2, LIST...>::value>
  {};

```

```
  template< typename T >
  struct Is_demand
  {
    static long test(Type_info::Demand const *);
    static char test(...);
    enum { value = sizeof(test((T*)0)) == sizeof(long) };
  };

  template< typename T, typename ... >
  struct First : T { typedef T type; };
} // Typeid

template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
struct __Kobject_base : BASES...
{
protected:
  typedef Derived Class;
  typedef Typeid::Iface<PROTO, Derived> __Iface;
  typedef Typeid_xx::Merge_list<
    Typeid::Iface_list<__Iface>,
    typename BASES::__Iface_list...
  > __Iface_list;

  static void __check_protocols__() noexcept
  {
    typedef Typeid_xx::Iface_conflict<__Iface, typename BASES::__Iface_list...> Conflict;
    static_assert(!Conflict::value, "ambiguous protocol ID, protocol also used in base class");

    typedef Typeid_xx::Conflict<typename BASES::__Iface_list...> Base_conflict;
    static_assert(!Base_conflict::value, "ambiguous protocol IDs in base classes");
  }

  // disambiguate cap()
  l4_cap_idx_t cap() const noexcept
  { return Typeid_xx::First<BASES...>::type::cap(); }

  L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }

  L4____GEN_TI_MEMBERS(Kobject_demand<BASES...>)

private:
  // This function returns the first base class (used below)
  template<typename B1, typename ...> struct Base1 { typedef B1 type; };

public:
  // Provide non-ambiguous conversion to Kobject
  operator Kobject const & () const noexcept
  { return *static_cast<typename Base1<BASES...>::type const *>(this); }

  // Provide non-ambiguous access of dec_refcnt()
  l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
    noexcept(noexcept(((typename Base1<BASES...>::type *)0)->dec_refcnt(diff, utcb)))
  { return Base1<BASES...>::type::dec_refcnt(diff, utcb); }
};

template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
Type_info const *const
__Kobject_base<Derived, PROTO, S_DEMAND, BASES...>::__Kobject_typeid::_b[] =
{
  (&BASES::__Kobject_typeid::_m)...
};

template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
L4____GEN_TI(__Kobject_base<Derived, PROTO, S_DEMAND, BASES...>);


// Test if the there is a Demand argument to Kobject_x
template< typename Derived, long PROTO, bool HAS_DEMAND, typename DEMAND, typename ...ARGS >
struct __Kobject_x_proto;

// YES: pass it to __Kobject_base
template< typename Derived, long PROTO, typename DEMAND, typename ...BASES>
struct __Kobject_x_proto<Derived, PROTO, true, DEMAND, BASES...> :
  __Kobject_base<Derived, PROTO, DEMAND, BASES...> {};

// NO: pass it empty Type_info::Demand_t
template< typename Derived, long PROTO, typename B1, typename ...BASES>
struct __Kobject_x_proto<Derived, PROTO, false, B1, BASES...> :
  __Kobject_base<Derived, PROTO, Type_info::Demand_t<>, B1, BASES...> {};

template< long P = PROTO_EMPTY >
struct Proto_t {};

template< typename Derived, typename ...ARGS >
struct Kobject_x;

template< typename Derived, typename A, typename ...ARGS >
```

```
struct Kobject_x<Derived, A, ARGS...> :
  __Kobject_x_proto<Derived, PROTO_ANY, Typeid_xx::Is_demand<A>::value, A, ARGS...>
{};

template< typename Derived, long PROTO, typename A, typename ...ARGS >
struct Kobject_x<Derived, Proto_t<PROTO>, A, ARGS...> :
  __Kobject_x_proto<Derived, PROTO, Typeid_xx::Is_demand<A>::value, A, ARGS...>
{};

}
#endif

#undef L4____GEN_TI
#undef L4____GEN_TI_MEMBERS

```

16.265 l4/sys/__vm-arm.h File Reference

Virtualization interface.

`#include <l4/sys/task>`

Include dependency graph for __vm-arm.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Vm

  Virtual machine host address space.

Namespaces

- L4

  L4 low-level kernel interface.

16.265.1 Detailed Description

Virtualization interface.

Definition in file __vm-arm.h.

16.266 __vm-arm.h

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <l4/sys/task>
00025
00026 namespace L4 {
00027
00028 class Vm : public Kobject_t<Vm, Task, L4_PROTO_VM>
00029 {
```

```
public:
  /*
   * Map the GIC's virtual GICC page to the task.
   *
   * \param vgicc_fpage   Flexpage that describes an area in the address space
   *                      of the destination task to map the vGICC page to.
   * \utcb{utcb}
   *
   * \return Syscall return tag.
   */
  l4_msgtag_t vgicc_map(l4_fpage_t const vgicc_fpage,
                        l4_utcb_t *utcb = l4_utcb()) noexcept
  { return l4_task_vgicc_map_u(cap(), vgicc_fpage, utcb); }

protected:
  Vm();

private:
  Vm(Vm const &);
  void operator = (Vm const &);
};

}
```

16.267 l4/sys/arm_smccc File Reference

ARM secure monitor call functions.

```
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for arm_smccc:

Data Structures

- class L4::Arm_smccc

  Wrapper for function calls that follow the ARM SMC/HVC calling convention.

Namespaces

- L4

  L4 low-level kernel interface.

16.267.1 Detailed Description

ARM secure monitor call functions.

Definition in file arm_smccc.

16.268 arm_smccc

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2018 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  *
00006  * This file is distributed under the terms of the GNU General Public
00007  * License, version 2.  Please see the COPYING-GPL-2 file for details.
00008  */
00013 #pragma once
00014
00015 #include <l4/sys/capability>
00016 #include <l4/sys/cxx/ipc_iface>
00017
00018 namespace L4 {
00019
00024 class L4_EXPORT Arm_smccc : public Kobject_0t<Arm_smccc, L4_PROTO_SMCCC>
00025 {
00026 public:
00063   L4_INLINE_RPC(l4_msgtag_t, call,
00064                 (l4_umword_t func, l4_umword_t in0, l4_umword_t in1,
00065                  l4_umword_t in2, l4_umword_t in3, l4_umword_t in4,
00066                  l4_umword_t in5, l4_umword_t *out0, l4_umword_t *out1,
00067                  l4_umword_t *out2, l4_umword_t *out3,
00068                  l4_umword_t client_id));
00069
00070   typedef L4::Typeid::Rpc_nocode<call_t> Rpcs;
00071 };
00072
00073 }
```

16.269 l4/sys/arm_smccc.h File Reference

ARM secure monitor call functions.

```
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for arm_smccc.h:

Functions

- l4_msgtag_t l4_arm_smccc_call (l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0, l4_umword_t in1, l4_umword_t in2, l4_umword_t in3, l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0, l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3, l4_umword_t client_id) L4_NOTHROW

  C interface for calling the ARM secure monitor, see L4::Arm_smccc::call() for the C++ interface.

16.269.1 Detailed Description

ARM secure monitor call functions.

Definition in file arm_smccc.h.

16.269.2 Function Documentation

16.269.2.1 l4_arm_smccc_call()

```
l4_msgtag_t l4_arm_smccc_call (
            l4_cap_idx_t pfc,
            l4_umword_t func,
            l4_umword_t in0,
            l4_umword_t in1,
            l4_umword_t in2,
            l4_umword_t in3,
```

```
l4_umword_t in4,
l4_umword_t in5,
l4_umword_t * out0,
l4_umword_t * out1,
l4_umword_t * out2,
l4_umword_t * out3,
l4_umword_t client_id )  [inline]
```

C interface for calling the ARM secure monitor, see L4::Arm_smccc::call() for the C++ interface.

Parameters

	pfc	Capability of the SMC kernel object. The input parameters consist of a function identifier, 6 arguments and a client id. Results are returned in 4 output parameters.
	func	Function identifier. • Bit 31 has to be set: This marks the call as *Fast Call*. *Yielding Calls* (bit 31 unset) are rejected by the kernel. • Bit 30 defines the calling convention: • Bit 30 == 1: 64-bit calling convention. • Bit 30 == 0: 32-bit calling convention. • Bits 24..29 determine the service call ID. Only service IDs >= 0x30000000 (*Trusted Application Calls* and *Trusted OS Calls*) are allowed.
in	*in0*	First input parameter.
in	*in1*	Second input parameter.
in	*in2*	Third input parameter.
in	*in3*	Fourth input parameter.
in	*in4*	Fifth input parameter.
in	*in5*	Sixth input parameter.
out	*out0*	First output parameter.
out	*out1*	Second output parameter.
out	*out2*	Third output parameter.
out	*out3*	Fourth output parameter.
in	*client↩ _id*	Client ID. According to the specification, this value might be ignored by certain functions.

Return values

-L4_ENOSYS	Either bit 31 of the function call not set or service ID < 0x30000000.
-L4_EINVAL	Invalid number of parameters.
<0	Other L4 error.
0	Success.

Definition at line 43 of file arm_smccc.h.

16.270 arm_smccc.h

```
00001 /*
00002  * Copyright (C) 2018 Kernkonzept GmbH.
```

```
 * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
 *
 * This file is distributed under the terms of the GNU General Public
 * License, version 2.  Please see the COPYING-GPL-2 file for details.
 */
#pragma once

#include <l4/sys/types.h>
#include <l4/sys/utcb.h>

L4_INLINE l4_msgtag_t
l4_arm_smccc_call(l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0,
                  l4_umword_t in1, l4_umword_t in2, l4_umword_t in3,
                  l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0,
                  l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3,
                  l4_umword_t client_id) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_arm_smccc_call_u(l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0,
                    l4_umword_t in1, l4_umword_t in2, l4_umword_t in3,
                    l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0,
                    l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3,
                    l4_umword_t client_id, l4_utcb_t *utcb) L4_NOTHROW;

/* IMPLEMENTATION ----------------------------------------------------------*/

#include <l4/sys/ipc.h>

L4_INLINE l4_msgtag_t
l4_arm_smccc_call(l4_cap_idx_t pfc, l4_umword_t func,
                  l4_umword_t in0, l4_umword_t in1,
                  l4_umword_t in2, l4_umword_t in3,
                  l4_umword_t in4, l4_umword_t in5,
                  l4_umword_t *out0, l4_umword_t *out1,
                  l4_umword_t *out2, l4_umword_t *out3,
                  l4_umword_t client_id) L4_NOTHROW
{
  return l4_arm_smccc_call_u(pfc, func, in0, in1, in2, in3, in4, in5,
                             out0, out1, out2, out3, client_id, l4_utcb());
}


L4_INLINE l4_msgtag_t
l4_arm_smccc_call_u(l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0,
                    l4_umword_t in1, l4_umword_t in2, l4_umword_t in3,
                    l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0,
                    l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3,
                    l4_umword_t client_id, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msgtag_t ret;
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[0] = func;
  v->mr[1] = in0;
  v->mr[2] = in1;
  v->mr[3] = in2;
  v->mr[4] = in3;
  v->mr[5] = in4;
  v->mr[6] = in5;
  v->mr[7] = client_id;

  ret = l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_SMCCC, 8, 0, 0),
                    L4_IPC_NEVER);

  if (l4_error(ret) >= 0)
    {
      *out0 = v->mr[0];
      *out1 = v->mr[1];
      *out2 = v->mr[2];
      *out3 = v->mr[3];
    }

  return ret;
}
```

16.271 l4/sys/cache.h File Reference

Cache-consistency functions.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for cache.h:

Functions

- int l4_cache_clean_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache clean a range in D-cache.
- int l4_cache_flush_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache flush a range.
- int l4_cache_inv_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache invalidate a range.
- int l4_cache_coherent (unsigned long start, unsigned long end) L4_NOTHROW

  Make memory coherent between I-cache and D-cache.
- int l4_cache_dma_coherent (unsigned long start, unsigned long end) L4_NOTHROW

  Make memory coherent for use with external memory.
- int l4_cache_dma_coherent_full (void) L4_NOTHROW

  Make memory coherent for use with external memory.

16.271.1 Detailed Description

Cache-consistency functions.

Date

2007-11

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file cache.h.

16.272 cache.h

```
00001
00011 /*
00012  * (c) 2007-2009 Author(s)
00013  *     economic rights: Technische Universität Dresden (Germany)
00014  *
00015  * This file is part of TUD:OS and distributed under the terms of the
00016  * GNU General Public License 2.
00017  * Please see the COPYING-GPL-2 file for details.
00018  *
00019  * As a special exception, you may use this file as part of a free software
00020  * library without restriction.  Specifically, if other files instantiate
00021  * templates or use macros or inline functions from this file, or you compile
00022  * this file and link it with other files to produce an executable, this
00023  * file does not by itself cause the resulting executable to be covered by
00024  * the GNU General Public License.  This exception does not however
00025  * invalidate any other reasons why the executable file might be covered by
00026  * the GNU General Public License.
00027  */
00028
00029 #ifndef __L4SYS__INCLUDE__CACHE_H__
00030 #define __L4SYS__INCLUDE__CACHE_H__
00031
00032 #include <l4/sys/compiler.h>
00033
00042 EXTERN_C_BEGIN
00043
00058 L4_INLINE int
00059 l4_cache_clean_data(unsigned long start,
00060                     unsigned long end) L4_NOTHROW;
00061
00076 L4_INLINE int
00077 l4_cache_flush_data(unsigned long start,
00078                     unsigned long end) L4_NOTHROW;
00079
00098 L4_INLINE int
00099 l4_cache_inv_data(unsigned long start,
00100                   unsigned long end) L4_NOTHROW;
00101
00113 L4_INLINE int
00114 l4_cache_coherent(unsigned long start,
00115                   unsigned long end) L4_NOTHROW;
00116
00128 L4_INLINE int
00129 l4_cache_dma_coherent(unsigned long start,
00130                       unsigned long end) L4_NOTHROW;
00131
00136 L4_INLINE int
00137 l4_cache_dma_coherent_full(void) L4_NOTHROW;
00138
00139 EXTERN_C_END
00140
00141 #endif /* ! __L4SYS__INCLUDE__CACHE_H__ */
```

16.273 arm/l4/sys/cache.h File Reference

Cache functions.

```
#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>
```

Include dependency graph for cache.h:

Functions

- int l4_cache_clean_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache clean a range in D-cache.
- int l4_cache_flush_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache flush a range.
- int l4_cache_inv_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache invalidate a range.
- int l4_cache_coherent (unsigned long start, unsigned long end) L4_NOTHROW

  Make memory coherent between I-cache and D-cache.
- int l4_cache_dma_coherent (unsigned long start, unsigned long end) L4_NOTHROW

  Make memory coherent for use with external memory.
- int l4_cache_dma_coherent_full (void) L4_NOTHROW

  Make memory coherent for use with external memory.

16.273.1 Detailed Description

Cache functions.

Date

2007-11

Author

Adam Lackorzynski `adam@os.inf.tu-dresden.de`

Definition in file cache.h.

16.274 cache.h

```
/*
 * (c) 2007-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#ifndef __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
#define __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__

#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>

#include_next <l4/sys/cache.h>

L4_INLINE void
l4_cache_op_arm_call(unsigned long op,
                     unsigned long start,
                     unsigned long end);

L4_INLINE void
l4_cache_op_arm_call(unsigned long op,
                     unsigned long start,
                     unsigned long end)
{
  register unsigned long _op    __asm__ ("r0") = op;
  register unsigned long _start __asm__ ("r1") = start;
  register unsigned long _end   __asm__ ("r2") = end;

  __asm__ __volatile__
    ("@ l4_cache_op_arm_call(start) \n\t"
     "mov     lr, pc              \n\t"
     "mov     pc, %[sc]             \n\t"
     "@ l4_cache_op_arm_call(end)   \n\t"
       :
  "=r" (_op),
  "=r" (_start),
  "=r" (_end)
       :
       [sc] "i" (L4_SYSCALL_MEM_OP),
  "0" (_op),
  "1" (_start),
  "2" (_end)
       :
  "cc", "memory", "lr"
       );
}

enum L4_mem_cache_ops
{
  L4_MEM_CACHE_OP_CLEAN_DATA        = 0,
  L4_MEM_CACHE_OP_FLUSH_DATA        = 1,
  L4_MEM_CACHE_OP_INV_DATA          = 2,
  L4_MEM_CACHE_OP_COHERENT          = 3,
  L4_MEM_CACHE_OP_DMA_COHERENT      = 4,
  L4_MEM_CACHE_OP_DMA_COHERENT_FULL = 5,
};

L4_INLINE int
l4_cache_clean_data(unsigned long start,
                    unsigned long end) L4_NOTHROW
{
  l4_cache_op_arm_call(L4_MEM_CACHE_OP_CLEAN_DATA, start, end);
  return 0;
}

L4_INLINE int
l4_cache_flush_data(unsigned long start,
                    unsigned long end) L4_NOTHROW
{
  l4_cache_op_arm_call(L4_MEM_CACHE_OP_FLUSH_DATA, start, end);
  return 0;
}
```

```
L4_INLINE int
l4_cache_inv_data(unsigned long start,
                  unsigned long end) L4_NOTHROW
{
  l4_cache_op_arm_call(L4_MEM_CACHE_OP_INV_DATA, start, end);
  return 0;
}

L4_INLINE int
l4_cache_coherent(unsigned long start,
                  unsigned long end) L4_NOTHROW
{
  l4_cache_op_arm_call(L4_MEM_CACHE_OP_COHERENT, start, end);
  return 0;
}

L4_INLINE int
l4_cache_dma_coherent(unsigned long start,
                      unsigned long end) L4_NOTHROW
{
  l4_cache_op_arm_call(L4_MEM_CACHE_OP_DMA_COHERENT, start, end);
  return 0;
}

L4_INLINE int
l4_cache_dma_coherent_full(void) L4_NOTHROW
{
  l4_cache_op_arm_call(L4_MEM_CACHE_OP_DMA_COHERENT_FULL, 0, 0);
  return 0;
}

#endif /* ! __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__ */
```

16.275 amd64/l4/sys/cache.h File Reference

Cache functions.

Functions

- int l4_cache_clean_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache clean a range in D-cache.
- int l4_cache_flush_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache flush a range.
- int l4_cache_inv_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache invalidate a range.
- int l4_cache_coherent (unsigned long start, unsigned long end) L4_NOTHROW

  Make memory coherent between I-cache and D-cache.
- int l4_cache_dma_coherent (unsigned long start, unsigned long end) L4_NOTHROW

  Make memory coherent for use with external memory.
- int l4_cache_dma_coherent_full (void) L4_NOTHROW

  Make memory coherent for use with external memory.

16.275.1 Detailed Description

Cache functions.

Definition in file cache.h.

16.276 cache.h

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#ifndef __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
#define __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__

#include_next <l4/sys/cache.h>

L4_INLINE int
l4_cache_clean_data(unsigned long start,
                    unsigned long end) L4_NOTHROW
{
  (void)start; (void)end;
  return 0;
}

L4_INLINE int
l4_cache_flush_data(unsigned long start,
                    unsigned long end) L4_NOTHROW
{
  (void)start; (void)end;
  return 0;
}

L4_INLINE int
l4_cache_inv_data(unsigned long start,
                  unsigned long end) L4_NOTHROW
{
  (void)start; (void)end;
  return 0;
}

L4_INLINE int
l4_cache_coherent(unsigned long start,
                  unsigned long end) L4_NOTHROW
{
  (void)start; (void)end;
  return 0;
}

L4_INLINE int
l4_cache_dma_coherent(unsigned long start,
                      unsigned long end) L4_NOTHROW
{
  (void)start; (void)end;
  return 0;
}

L4_INLINE int
l4_cache_dma_coherent_full(void) L4_NOTHROW
{
  return 0;
}

#endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__ */
```

16.277 x86/l4/sys/cache.h File Reference

Cache functions.

Functions

- int l4_cache_clean_data (unsigned long start, unsigned long end) L4_NOTHROW

Cache clean a range in D-cache.

- int l4_cache_flush_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache flush a range.

- int l4_cache_inv_data (unsigned long start, unsigned long end) L4_NOTHROW

  Cache invalidate a range.

- int l4_cache_coherent (unsigned long start, unsigned long end) L4_NOTHROW

  Make memory coherent between I-cache and D-cache.

- int l4_cache_dma_coherent (unsigned long start, unsigned long end) L4_NOTHROW

  Make memory coherent for use with external memory.

- int l4_cache_dma_coherent_full (void) L4_NOTHROW

  Make memory coherent for use with external memory.

16.277.1 Detailed Description

Cache functions.

Definition in file cache.h.

16.278 cache.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #ifndef __L4SYS__INCLUDE__ARCH_X86__CACHE_H__
00023 #define __L4SYS__INCLUDE__ARCH_X86__CACHE_H__
00024
00025 #include_next <l4/sys/cache.h>
00026
00027 L4_INLINE int
00028 l4_cache_clean_data(unsigned long start,
00029                     unsigned long end) L4_NOTHROW
00030 {
00031   (void)start; (void)end;
00032   return 0;
00033 }
00034
00035 L4_INLINE int
00036 l4_cache_flush_data(unsigned long start,
00037                     unsigned long end) L4_NOTHROW
00038 {
00039   (void)start; (void)end;
00040   return 0;
00041 }
00042
00043 L4_INLINE int
00044 l4_cache_inv_data(unsigned long start,
00045                   unsigned long end) L4_NOTHROW
00046 {
00047   (void)start; (void)end;
00048   return 0;
00049 }
00050
00051 L4_INLINE int
00052 l4_cache_coherent(unsigned long start,
00053                   unsigned long end) L4_NOTHROW
```

```
{
  (void)start; (void)end;
  return 0;
}

L4_INLINE int
l4_cache_dma_coherent(unsigned long start,
                      unsigned long end) L4_NOTHROW
{
  (void)start; (void)end;
  return 0;
}

L4_INLINE int
l4_cache_dma_coherent_full(void) L4_NOTHROW
{
  return 0;
}

#endif /* ! __L4SYS__INCLUDE__ARCH_X86__CACHE_H__ */
```

16.279 l4/sys/capability File Reference

L4::Cap related definitions.

```
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kobject>
#include <l4/sys/task.h>
#include <l4/sys/meta>
```

Include dependency graph for capability:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4

  L4 low-level kernel interface.

Macros

- #define L4_DISABLE_COPY(_class)

  Disable copy of a class.

Functions

- template<typename T , typename F >
  Cap< T > L4::cap_dynamic_cast (Cap< F > const &c) noexcept

  `dynamic_cast` for capabilities.

16.279.1 Detailed Description

L4::Cap related definitions.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file capability.

16.279.2 Macro Definition Documentation

16.279.2.1 L4_DISABLE_COPY

```
#define L4_DISABLE_COPY(
            _class )
```

Value:

```
  public:                                          \
    _class(_class const &) = delete;               \
    _class operator = (_class const &) = delete; \
  private:
```

Disable copy of a class.

Parameters

_class	Name of the class that shall not have value copy semantics.

The typical use of this is:

```
class Non_value
{
  L4_DISABLE_COPY(Non_value)
  ...
}
```

Definition at line 61 of file capability.

16.280 capability

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00009 /*
00010  * (c) 2008-2009,2015 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction.  Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License.  This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027
00028 #include <l4/sys/consts.h>
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/kobject>
00031 #include <l4/sys/task.h>
00032
00033 namespace L4
00034 {
00035
00036 /* Forward declarations for our kernel object classes. */
00037 class Task;
00038 class Thread;
00039 class Factory;
00040 class Irq;
00041 class Log;
00042 class Vm;
00043 class Kobject;
00044
00060 #if __cplusplus >= 201103L
00061 #  define L4_DISABLE_COPY(_class)                \
00062   public:                                        \
00063     _class(_class const &) = delete;             \
00064     _class operator = (_class const &) = delete; \
00065   private:
00066 #else
00067 #  define L4_DISABLE_COPY(_class)        \
00068   private:                               \
00069     _class(_class const &);              \
00070     _class operator = (_class const &);
00071 #endif
00072
00073
00074 #define L4_KOBJECT_DISABLE_COPY(_class) \
00075   protected:                            \
00076     _class();                           \
00077   L4_DISABLE_COPY(_class)
00078
00079
00080 #define L4_KOBJECT(_class) L4_KOBJECT_DISABLE_COPY(_class)
00081
00082 inline l4_msgtag_t
00083 Cap_base::validate(Cap<Task> task, l4_utcb_t *u) const noexcept
00084 {
00085   return is_valid() ? l4_task_cap_valid_u(task.cap(), _c, u)
```

```
                    : l4_msgtag(0, 0, 0, 0);
}

inline l4_msgtag_t
Cap_base::validate(l4_utcb_t *u) const noexcept
{
  return is_valid() ? l4_task_cap_valid_u(L4_BASE_TASK_CAP, _c, u)
                    : l4_msgtag(0, 0, 0, 0);
}

}; // namespace L4

#include <l4/sys/meta>

namespace L4 {

template< typename T, typename F >
inline
Cap<T> cap_dynamic_cast(Cap<F> const &c) noexcept
{
  if (!c.is_valid())
    return Cap<T>::Invalid;

  Cap<Meta> mc = cap_reinterpret_cast<Meta>(c);
  Type_info const *m = kobject_typeid<T>();
  if (m->proto() && l4_error(mc->supports(m->proto())) > 0)
    return Cap<T>(c.cap());

  // FIXME: use generic checker
#if 0
  if (l4_error(mc->supports(T::kobject_proto())) > 0)
    return Cap<T>(c.cap());
#endif

  return Cap<T>::Invalid;
}

}
```

16.281 l4/sys/compiler.h File Reference

L4 compiler related defines.

`#include <l4/sys/linkage.h>`

Include dependency graph for compiler.h:

This graph shows which files directly or indirectly include this file:

Macros

- #define L4_ALWAYS_INLINE

  L4 Inline function attribute.
- #define L4_NOTHROW

  Mark a function declaration and definition as never throwing an exception.
- #define EXTERN_C_BEGIN

  Start section with C types and functions.
- #define EXTERN_C_END

  End section with C types and functions.
- #define EXTERN_C

  Mark C types and functions.
- #define __END_DECLS

  End section with C types and functions.
- #define L4_NORETURN

  Noreturn function attribute.
- #define L4_NOINSTRUMENT

  No instrumentation function attribute.
- #define L4_HIDDEN

  Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.
- #define L4_LIKELY(x)

  Expression is likely to execute.
- #define L4_UNLIKELY(x)

  Expression is unlikely to execute.
- #define L4_STICKY(x)

  Mark symbol sticky (even not there)
- #define L4_DEPRECATED(s)

  Mark symbol deprecated.
- #define L4_stringify_helper(x)

  stringify helper.
- #define L4_stringify(x)

  stringify.

Functions

- unsigned long l4_align_stack_for_direct_fncall (unsigned long stack)

  Specify the desired alignment of the stack pointer.
- void l4_barrier (void)

  Memory barrier.
- void l4_mb (void)

  Memory barrier.
- void l4_wmb (void)

  Write memory barrier.

16.281.1 Detailed Description

L4 compiler related defines.

Definition in file compiler.h.

16.282 compiler.h

```
/*****************************************************************************/
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Frank Mehnert <fm3@os.inf.tu-dresden.de>,
 *               Jork Löser <jork@os.inf.tu-dresden.de>,
 *               Ronald Aigner <ra3@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
/*****************************************************************************/
#ifndef __L4_COMPILER_H__
#define __L4_COMPILER_H__

#if !defined(__ASSEMBLY__) && !defined(__ASSEMBLER__)

#ifndef L4_INLINE
#ifndef __cplusplus
#  ifdef __OPTIMIZE__
#    define L4_INLINE_STATIC static __inline__
#    define L4_INLINE_EXTERN extern __inline__
#    ifdef __GNUC_STDC_INLINE__
#      define L4_INLINE L4_INLINE_STATIC
#    else
#      define L4_INLINE L4_INLINE_EXTERN
#    endif
#  else /* ! __OPTIMIZE__ */
#    define L4_INLINE static
#  endif /* ! __OPTIMIZE__ */
#else /* __cplusplus */
#  define L4_INLINE inline
#endif  /* __cplusplus */
#endif  /* L4_INLINE */

#define L4_ALWAYS_INLINE L4_INLINE __attribute__((__always_inline__))


#define L4_DECLARE_CONSTRUCTOR(func, prio) \
  static inline __attribute__((constructor(prio))) void func ## _ctor_func(void) { func(); }


#ifndef __cplusplus
#  define L4_NOTHROW__A       __attribute__((nothrow))
#  define L4_NOTHROW
#  define EXTERN_C_BEGIN
#  define EXTERN_C_END
#  define EXTERN_C
#  ifndef __BEGIN_DECLS
#    define __BEGIN_DECLS
#  endif
#  ifndef __END_DECLS
#    define __END_DECLS
#  endif
#  define L4_DEFAULT_PARAM(x)
#else /* __cplusplus */
#  if __cplusplus >= 201103L
#    define L4_NOTHROW noexcept
#  else /* C++ < 11 */
#    define L4_NOTHROW throw()
#  endif
#  define EXTERN_C_BEGIN extern "C" {
#  define EXTERN_C_END }
#  define EXTERN_C extern "C"
#  ifndef __BEGIN_DECLS
#    define __BEGIN_DECLS extern "C" {
#  endif
#  ifndef __END_DECLS
#    define __END_DECLS }
#  endif
#  define L4_DEFAULT_PARAM(x) = x
#endif /* __cplusplus */

#define L4_NORETURN __attribute__((noreturn))
```

```
#define L4_PURE __attribute__((pure))

#define L4_NOINSTRUMENT __attribute__((no_instrument_function))
#ifndef L4_HIDDEN
#  define L4_HIDDEN __attribute__((visibility("hidden")))
#endif
#ifndef L4_EXPORT
#  define L4_EXPORT __attribute__((visibility("default")))
#endif
#ifndef L4_EXPORT_TYPE
#  ifdef __cplusplus
#    define L4_EXPORT_TYPE __attribute__((visibility("default")))
#  else
#    define L4_EXPORT_TYPE
#  endif
#endif
#define L4_STRONG_ALIAS(name, aliasname) L4__STRONG_ALIAS(name, aliasname)
#define L4__STRONG_ALIAS(name, aliasname) \
  extern __typeof (name) aliasname __attribute__ ((alias (#name)));

#if defined(ARCH_x86) || defined(ARCH_amd64) || \
  defined(ARCH_arm) || defined(ARCH_arm64) || \
  defined(ARCH_mips) || defined(ARCH_ppc32) || defined(ARCH_sparc)
# define L4_STACK_ALIGN         __BIGGEST_ALIGNMENT__
#else
# error Define L4_STACK_ALIGN for this target!
#endif

#if defined(ARCH_x86) || defined(ARCH_amd64)
L4_INLINE unsigned long l4_align_stack_for_direct_fncall(unsigned long stack)
{
  if ((stack & (L4_STACK_ALIGN - 1)) == (L4_STACK_ALIGN - sizeof(unsigned long)))
    return stack;
  return (stack & ~(L4_STACK_ALIGN)) - sizeof(unsigned long);
}
#else
L4_INLINE unsigned long l4_align_stack_for_direct_fncall(unsigned long stack)
{
  return stack & ~(L4_STACK_ALIGN);
}
#endif

#endif /* !__ASSEMBLY__ */

#include <l4/sys/linkage.h>

#define L4_LIKELY(x)  __builtin_expect((x),1)
#define L4_UNLIKELY(x)  __builtin_expect((x),0)

/* Make sure that the function is not removed by optimization. Without the
 * "used" attribute, unreferenced static functions are removed. */
#define L4_STICKY(x)     __attribute__((used)) x
#define L4_DEPRECATED(s) __attribute__((deprecated(s)))

#ifndef static_assert
# if !defined(__cplusplus)
#  define static_assert(x, y) _Static_assert(x, y)
# elif __cplusplus < 201103L
#  define static_assert(x, y) \
   extern int l4_static_assert[-(!(x))] __attribute__((unused))
# endif
#endif

#define L4_stringify_helper(x) #x
#define L4_stringify(x)        L4_stringify_helper(x)

#ifndef __ASSEMBLER__
L4_INLINE void l4_barrier(void);

L4_INLINE void l4_mb(void);

L4_INLINE void l4_wmb(void);


/* Implementations */
L4_INLINE void l4_barrier(void)
{
  __asm__ __volatile__ ("" : : : "memory");
}

L4_INLINE void l4_mb(void)
{
  __asm__ __volatile__ ("" : : : "memory");
}

L4_INLINE void l4_wmb(void)
```

```
{
  __asm__ __volatile__ ("" : : : "memory");
}
#endif

#endif /* !__L4_COMPILER_H__ */
```

16.283 l4/sys/consts.h File Reference

Common constants.

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
```

Include dependency graph for consts.h:

This graph shows which files directly or indirectly include this file:

Macros

- #define L4_PAGESIZE

  Minimal page size (in bytes).
- #define L4_PAGEMASK

  Mask for the page number.
- #define L4_LOG2_PAGESIZE

  Number of bits used for page offset.
- #define L4_SUPERPAGESIZE

  Size of a large page.

- #define L4_SUPERPAGEMASK

  Mask for the number of a large page.
- #define L4_LOG2_SUPERPAGESIZE

  Number of bits used as offset for a large page.
- #define L4_INVALID_PTR ((void ∗)L4_INVALID_ADDR)

  Invalid address as pointer type.

Enumerations

- enum l4_syscall_flags_t { L4_SYSF_NONE , L4_SYSF_SEND , L4_SYSF_RECV , L4_SYSF_OPEN_WAIT , L4_SYSF_REPLY , L4_SYSF_CALL , L4_SYSF_WAIT , L4_SYSF_SEND_AND_WAIT , L4_SYSF_REPLY_AND_WAIT }

  Capability selector flags.
- enum l4_cap_consts_t { L4_CAP_SHIFT , L4_CAP_SIZE , **L4_CAP_OFFSET** = 1UL << L4_CAP_SHIFT , L4_CAP_MASK , L4_INVALID_CAP , **L4_INVALID_CAP_BIT** = 1UL << (L4_CAP_SHIFT - 1) }

  Constants related to capability selectors.
- enum l4_unmap_flags_t { L4_FP_ALL_SPACES , L4_FP_DELETE_OBJ , L4_FP_OTHER_SPACES }

  Flags for the unmap operation.
- enum l4_msg_item_consts_t { L4_ITEM_MAP = 8 , L4_ITEM_CONT = 1 , L4_MAP_ITEM_GRANT = 2 , L4_MAP_ITEM_MAP = 0 , L4_RCV_ITEM_SINGLE_CAP = L4_ITEM_MAP | 2 , L4_RCV_ITEM_LOCAL_ID = 4 }

  Constants for message items.
- enum l4_buffer_desc_consts_t { L4_BDR_MEM_SHIFT = 0 , L4_BDR_IO_SHIFT = 5 , L4_BDR_OBJ_SHIFT = 10 , **L4_BDR_OFFSET_MASK** = (1UL << 20) - 1 }

  Constants for buffer descriptors.
- enum l4_default_caps_t { L4_BASE_TASK_CAP , L4_BASE_FACTORY_CAP , L4_BASE_THREAD_CAP , L4_BASE_PAGER_CAP , L4_BASE_LOG_CAP , L4_BASE_ICU_CAP , L4_BASE_SCHEDULER_CAP , L4_BASE_IOMMU_CAP , L4_BASE_DEBUGGER_CAP , L4_BASE_ARM_SMCCC_CAP , **L4_BASE_CAPS_LAST_P1** , L4_BASE_CAPS_LAST = L4_BASE_CAPS_LAST_P1 - 1 }

  Default capabilities setup for the initial tasks.
- enum l4_addr_consts_t { L4_INVALID_ADDR = ∼0UL }

  Address related constants.

Functions

- l4_addr_t l4_trunc_page (l4_addr_t address) L4_NOTHROW

  Round an address down to the next lower page boundary.
- l4_addr_t l4_trunc_size (l4_addr_t address, unsigned char bits) L4_NOTHROW

  Round an address down to the next lower flex page with size bits.
- l4_addr_t l4_round_page (l4_addr_t address) L4_NOTHROW

  Round address up to the next page.
- l4_addr_t l4_round_size (l4_addr_t value, unsigned char bits) L4_NOTHROW

  Round value up to the next alignment with bits size.
- unsigned l4_bytes_to_mwords (unsigned size) L4_NOTHROW

  Determine how many machine words (l4_umword_t) are required to store a buffer of 'size' bytes.

16.283.1 Detailed Description

Common constants.

Definition in file consts.h.

16.284 consts.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction.  Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License.  This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #ifndef __L4_SYS__INCLUDE__CONSTS_H__
00027 #define __L4_SYS__INCLUDE__CONSTS_H__
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/l4int.h>
00031
00039 enum l4_syscall_flags_t
00040 {
00048   L4_SYSF_NONE      = 0x00,
00049
00058   L4_SYSF_SEND      = 0x01,
00059
00069   L4_SYSF_RECV      = 0x02,
00070
00080   L4_SYSF_OPEN_WAIT = 0x04,
00081
00089   L4_SYSF_REPLY     = 0x08,
00090
00097   L4_SYSF_CALL           = L4_SYSF_SEND | L4_SYSF_RECV,
00098
00105   L4_SYSF_WAIT           = L4_SYSF_OPEN_WAIT | L4_SYSF_RECV,
00106
00113   L4_SYSF_SEND_AND_WAIT  = L4_SYSF_OPEN_WAIT | L4_SYSF_CALL,
00114
00121   L4_SYSF_REPLY_AND_WAIT = L4_SYSF_WAIT | L4_SYSF_SEND | L4_SYSF_REPLY
00122 };
00123
00128 enum l4_cap_consts_t
00129 {
00131   L4_CAP_SHIFT   = 12UL,
00133   L4_CAP_SIZE    = 1UL « L4_CAP_SHIFT,
00134   L4_CAP_OFFSET  = 1UL « L4_CAP_SHIFT,
00139   L4_CAP_MASK    = ~0UL « (L4_CAP_SHIFT - 1),
00141   L4_INVALID_CAP = ~0UL « (L4_CAP_SHIFT - 1),
00142
00143   L4_INVALID_CAP_BIT = 1UL « (L4_CAP_SHIFT - 1),
00144 };
00145
00146 enum l4_sched_consts_t
00147 {
00148   L4_SCHED_MIN_PRIO = 0,
00149   L4_SCHED_MAX_PRIO = 255,
00150 };
00151
00157 enum l4_unmap_flags_t
00158 {
00171   L4_FP_ALL_SPACES   = 0x80000000UL,
00172
00179   L4_FP_DELETE_OBJ   = 0xc0000000UL,
00180
00186   L4_FP_OTHER_SPACES = 0x0UL
```

```
};

enum l4_msg_item_consts_t
{
  L4_ITEM_MAP       = 8,

  L4_ITEM_CONT      = 1,

  // send
  L4_MAP_ITEM_GRANT = 2,
  L4_MAP_ITEM_MAP   = 0,

  // receive
  L4_RCV_ITEM_SINGLE_CAP = L4_ITEM_MAP | 2,

  L4_RCV_ITEM_LOCAL_ID   = 4,
};

enum l4_buffer_desc_consts_t
{
  L4_BDR_MEM_SHIFT   = 0,
  L4_BDR_IO_SHIFT    = 5,
  L4_BDR_OBJ_SHIFT   = 10,
  L4_BDR_OFFSET_MASK = (1UL « 20) - 1,
};

enum l4_default_caps_t
{
  L4_BASE_TASK_CAP      = 1UL « L4_CAP_SHIFT,
  L4_BASE_FACTORY_CAP   = 2UL « L4_CAP_SHIFT,
  L4_BASE_THREAD_CAP    = 3UL « L4_CAP_SHIFT,
  L4_BASE_PAGER_CAP     = 4UL « L4_CAP_SHIFT,
  L4_BASE_LOG_CAP       = 5UL « L4_CAP_SHIFT,
  L4_BASE_ICU_CAP       = 6UL « L4_CAP_SHIFT,
  L4_BASE_SCHEDULER_CAP = 7UL « L4_CAP_SHIFT,
  L4_BASE_IOMMU_CAP     = 8UL « L4_CAP_SHIFT,
  L4_BASE_DEBUGGER_CAP  = 10UL « L4_CAP_SHIFT,
  L4_BASE_ARM_SMCCC_CAP = 11UL « L4_CAP_SHIFT,

  L4_BASE_CAPS_LAST_P1,
  L4_BASE_CAPS_LAST = L4_BASE_CAPS_LAST_P1 - 1
};

#define L4_PAGESIZE   (1UL « L4_PAGESHIFT)

#define L4_PAGEMASK   (~(L4_PAGESIZE - 1))

#define L4_LOG2_PAGESIZE  L4_PAGESHIFT

#define L4_SUPERPAGESIZE  (1UL « L4_SUPERPAGESHIFT)

#define L4_SUPERPAGEMASK  (~(L4_SUPERPAGESIZE - 1))

#define L4_LOG2_SUPERPAGESIZE L4_SUPERPAGESHIFT

L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address) L4_NOTHROW;
L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address) L4_NOTHROW
{ return address & L4_PAGEMASK; }

L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits) L4_NOTHROW;
L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits) L4_NOTHROW
{ return address & (~0UL « bits); }

L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW;
L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW
{ return (address + L4_PAGESIZE - 1) & L4_PAGEMASK; }

L4_INLINE l4_addr_t l4_round_size(l4_addr_t value, unsigned char bits) L4_NOTHROW;
L4_INLINE l4_addr_t l4_round_size(l4_addr_t value, unsigned char bits) L4_NOTHROW
{ return (value + (1UL « bits) - 1) & (~0UL « bits); }

L4_INLINE unsigned l4_bytes_to_mwords(unsigned size) L4_NOTHROW;
L4_INLINE unsigned l4_bytes_to_mwords(unsigned size) L4_NOTHROW
{ return (size + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t); }

enum l4_addr_consts_t {
  L4_INVALID_ADDR = ~0UL
};

#define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)

#ifndef NULL
#ifndef __cplusplus
# define NULL ((void *)0)
#else
# define NULL 0
#endif
```

```
#endif

#endif /* ! __L4_SYS__INCLUDE__CONSTS_H__ */
```

16.285 arm/l4/sys/consts.h File Reference

Common L4 constants, arm version.

`#include <l4/sys/l4int.h>`
Include dependency graph for consts.h:

Macros

- #define L4_PAGESHIFT 12

  Size of a page, log2-based.
- #define L4_SUPERPAGESHIFT 21

  Size of a large page, log2-based.

16.285.1 Detailed Description

Common L4 constants, arm version.

Definition in file consts.h.

16.286 consts.h

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
```

```
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#ifndef _L4_SYS_CONSTS_H
#define _L4_SYS_CONSTS_H

/* L4 includes */
#include <l4/sys/l4int.h>
#define L4_PAGESHIFT    12

#define L4_SUPERPAGESHIFT 21

#include_next <l4/sys/consts.h>

#endif /* !_L4_SYS_CONSTS_H */
```

16.287 amd64/l4/sys/consts.h File Reference

Common L4 constants, amd64 version.

Macros

- #define L4_PAGESHIFT 12

  Size of a page, log2-based.
- #define L4_SUPERPAGESHIFT 21

  Size of a large page, log2-based.

16.287.1 Detailed Description

Common L4 constants, amd64 version.

Definition in file consts.h.

16.288 consts.h

```
/*****************************************************************************/
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
 *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
/*****************************************************************************/
#ifndef __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
#define __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__

#define L4_PAGESHIFT    12

#define L4_SUPERPAGESHIFT 21

#include_next <l4/sys/consts.h>

#endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__ */
```

16.289 x86/l4/sys/consts.h File Reference

Common L4 constants, x86 version.

Macros

- #define L4_PAGESHIFT 12

  Size of a page log2-based.
- #define L4_SUPERPAGESHIFT 22

  Size of a large page log2-based.

16.289.1 Detailed Description

Common L4 constants, x86 version.

Definition in file consts.h.

16.290 consts.h

```
00001 /*****************************************************************************/
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011  *               Lars Reuther <reuther@os.inf.tu-dresden.de>
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU General Public License 2.
00016  * Please see the COPYING-GPL-2 file for details.
00017  *
00018  * As a special exception, you may use this file as part of a free software
00019  * library without restriction.  Specifically, if other files instantiate
00020  * templates or use macros or inline functions from this file, or you compile
00021  * this file and link it with other files to produce an executable, this
00022  * file does not by itself cause the resulting executable to be covered by
00023  * the GNU General Public License.  This exception does not however
00024  * invalidate any other reasons why the executable file might be covered by
00025  * the GNU General Public License.
00026  */
00027 /*****************************************************************************/
00028 #ifndef __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00029 #define __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00030
00035 #define L4_PAGESHIFT    12
00036
00041 #define L4_SUPERPAGESHIFT 22
00042
00043 #include_next <l4/sys/consts.h>
00044
00045 #endif /* ! __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__ */
```

16.291 l4/re/consts.h File Reference

Constants.

```
#include <l4/sys/consts.h>
```

Include dependency graph for consts.h:

This graph shows which files directly or indirectly include this file:

Enumerations

- enum

  Defaults for local thread priorities.

16.291.1 Detailed Description

Constants.

Definition in file consts.h.

16.291.2 Enumeration Type Documentation

16.291.2.1 anonymous enum

```
anonymous enum
```

Defaults for local thread priorities.

Priorities are to be seen as local. These are used by the loader and libpthread. The are to be understood as 'local', which means the actual priority of the thread (as seen by the kernel) is the base priority as defined by the scheduler plus the local priority.

Definition at line 39 of file consts.h.

16.292 consts.h

```
00001
00005 /*
00006  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <l4/sys/consts.h>
00025
00026 enum
00027 {
00028   L4RE_THIS_TASK_CAP = 1UL « L4_CAP_SHIFT,
00029 };
00030
00039 enum
00040 {
00041   L4RE_MAIN_THREAD_PRIO = 2, /* Priority of the main thread */
00042 };
00043
```

16.293 l4/re/consts File Reference

Constants.

```
#include <l4/sys/capability>
#include <l4/sys/consts.h>
```

```
#include <l4/re/env.h>
```

Include dependency graph for consts:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4Re

  L4Re C++ Interfaces.

16.293.1 Detailed Description

Constants.

Definition in file consts.

16.294 consts

```
// -*- Mode: C++ -*-
// vim:ft=cpp
/*
 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/sys/capability>
#include <l4/sys/consts.h>
#include <l4/re/env.h>

namespace L4Re {
  static L4::Cap<L4::Task>::Cap_type const This_task
    = (L4::Cap<L4::Task>::Cap_type)(L4RE_THIS_TASK_CAP);
}
```

16.295 l4/sys/cxx/ipc_client File Reference

```
#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/__typeinfo.h>
#include <l4/sys/err.h>
```

Include dependency graph for ipc_client:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4

L4 low-level kernel interface.

- L4::Ipc

  IPC related functionality.

- L4::Ipc::Msg

  IPC Message related functionality.

Macros

- #define L4_RPC_DEF(name)

  Generate the definition of an RPC stub.

16.295.1 Macro Definition Documentation

16.295.1.1 L4_RPC_DEF

```
#define L4_RPC_DEF(
                name )
```

Value:

```
template struct  L4::Ipc::Msg::Rpc_call \
  <name##_t, name##_t::class_type, name##_t::ipc_type, name##_t::flags_type>
```

Generate the definition of an RPC stub.

Parameters

name	The fully qualified method name to be implemented, this means `class::method`.

This macro generates the definition (implementation) for the given RPC interface method.

Definition at line 43 of file ipc_client.

16.296 ipc_client

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction.  Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License.  This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019 #pragma GCC system_header
00020
```

```
#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/__typeinfo.h>
#include <l4/sys/err.h>

namespace L4 { namespace Ipc { namespace Msg {
//-----------------------------------------------------------------------

#define L4_RPC_DEF(name) \
  template struct  L4::Ipc::Msg::Rpc_call \
    <name##_t, name##_t::class_type, name##_t::ipc_type, name##_t::flags_type>


//---------------------------------------------------
//Implementation of the RPC call
template<typename OP, typename C, typename FLAGS, typename R, typename ...ARGS>
R L4_EXPORT
Rpc_call<OP, C, R (ARGS...), FLAGS>::
  call(L4::Cap<C> cap, typename _Elem<ARGS>::arg_type ...a, l4_utcb_t *utcb) noexcept
{
  return Rpc_inline_call<OP, C, R (ARGS...), FLAGS>::call(cap, a..., utcb);
}

} // namespace Msg
} // namespace Ipc
} // namespace L4

```

16.297 l4/sys/cxx/ipc_iface File Reference

Interface Definition Language.

```
#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/__typeinfo.h>
```

Include dependency graph for ipc_iface:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct L4::Ipc::Call

  RPC attribute for a standard RPC call.
- struct L4::Ipc::Call_zero_send_timeout

  RPC attribute for an RPC call, with zero send timeout.
- struct L4::Ipc::Call_t< RIGHTS >

  RPC attribute for an RPC call with required rights.
- struct L4::Ipc::Send_only

  RPC attribute for a send-only RPC.

Namespaces

- L4

  L4 low-level kernel interface.
- L4::Ipc

  IPC related functionality.
- L4::Ipc::Msg

  IPC Message related functionality.

Macros

- #define L4_INLINE_RPC_NF(res, name, args...)

  Define an inline RPC call type (the type only, no callable).
- #define L4_INLINE_RPC_NF_OP(op, res, name, args...)

  Define an inline RPC call type with specific opcode (the type only, no callable).
- #define L4_INLINE_RPC(res, name, args, attr...) res name args

  Define an inline RPC call (type and callable).
- #define L4_INLINE_RPC_OP(op, res, name, args, attr...) res name args

  Define an inline RPC call with specific opcode (type and callable).
- #define L4_RPC_NF(res, name, args...)

  Define an RPC call type (the type only, no callable).
- #define L4_RPC_NF_OP(op, res, name, args...)

  Define an RPC call type with specific opcode (the type only, no callable).
- #define L4_RPC(res, name, args, attr...) res name args

  Define an RPC call (type and callable).
- #define L4_RPC_OP(op, res, name, args, attr...) res name args

  Define an RPC call with specific opcode (type and callable).

16.297.1 Detailed Description

Interface Definition Language.

See also

L4_RPC, L4_INLINE_RPC, L4::Ipc::Call L4::Ipc::Send_only, L4::Ipc::Msg::Rpc_call, L4::Ipc::Msg::Rpc_↩ inline_call

Definition in file ipc_iface.

16.297.2 Macro Definition Documentation

16.297.2.1 L4_INLINE_RPC

```
#define L4_INLINE_RPC(
            res,
            name,
            args,
            attr...  ) res name args
```

Define an inline RPC call (type and callable).

Parameters

res	The result type of the RPC call
name	The name of the function (`name_t` is used for the type.)
args	The argument list of the RPC function.
attr	Optional RPC attributes (L4::Ipc::Call, L4::Ipc::Call_t etc.).

Definition at line 469 of file ipc_iface.

16.297.2.2 L4_INLINE_RPC_NF

```
#define L4_INLINE_RPC_NF(
                res,
                name,
                args...  )
```

Value:

```
  struct name##_t : L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> \
  {                                                                           \
    typedef L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> type;    \
    L4_INLINE_RPC_SRV_FORWARD(name);                                          \
  }
```

Define an inline RPC call type (the type only, no callable).

Parameters

res	The result type of the RPC call
name	The name of the function (`name_t` is used for the type.)
args	The argument list of the RPC function, and RPC attributes (L4::Ipc::Call, L4::Ipc::Call_t etc.).

Stubs generated by this macro can be used explicitly in custom wrapper methods that need to use the underlying RPC code and provide some higher level abstraction, for example with default arguments or extra argument conversion.

Definition at line 440 of file ipc_iface.

16.297.2.3 L4_INLINE_RPC_NF_OP

```
#define L4_INLINE_RPC_NF_OP(
                op,
                res,
                name,
                args...  )
```

Value:

```
  struct name##_t : L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> \
  {                                                                           \
    typedef L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> type;    \
    enum { Opcode = (op) };                                                   \
    L4_INLINE_RPC_SRV_FORWARD(name);                                          \
  }
```

Define an inline RPC call type with specific opcode (the type only, no callable).

Parameters

op	The opcode number for this function
res	The result type of the RPC call
name	The name of the function (`name_t` is used for the type.)
args	The argument list of the RPC function, and RPC attributes (L4::Ipc::Call, L4::Ipc::Call_t etc.).

Stubs generated by this macro can be used explicitly in custom wrapper methods that need to use the underlying RPC code and provide some higher level abstraction, for example with default arguments or extra argument conversion.

Definition at line 453 of file ipc_iface.

16.297.2.4 L4_INLINE_RPC_OP

```
#define L4_INLINE_RPC_OP(
              op,
              res,
              name,
              args,
              attr...  ) res name args
```

Define an inline RPC call with specific opcode (type and callable).

Parameters

op	The opcode number for this function
res	The result type of the RPC call
name	The name of the function (`name_t` is used for the type.)
args	The argument list of the RPC function.
attr	Optional RPC attributes (L4::Ipc::Call, L4::Ipc::Call_t etc.).

Definition at line 484 of file ipc_iface.

16.297.2.5 L4_RPC

```
#define L4_RPC(
              res,
              name,
              args,
              attr...  ) res name args
```

Define an RPC call (type and callable).

Parameters

res	The result type of the RPC call
name	The name of the function (`name_t` is used for the type.)
args	The argument list of the RPC function.
attr	Optional RPC attributes (L4::Ipc::Call, L4::Ipc::Call_t etc.).

Definition at line 528 of file ipc_iface.

16.297.2.6 L4_RPC_NF

```
#define L4_RPC_NF(
                res,
                name,
                args...  )
```

Value:

```
  struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>        \
  {                                                                          \
    typedef L4::Ipc::Msg::Rpc_call<name##_t, Class, res args> type;          \
    L4_INLINE_RPC_SRV_FORWARD(name);                                         \
  }
```

Define an RPC call type (the type only, no callable).

Parameters

res	The result type of the RPC call
name	The name of the function (`name_t` is used for the type.)
args	The argument list of the RPC function, and RPC attributes (L4::Ipc::Call, L4::Ipc::Call_t etc.).

Definition at line 497 of file ipc_iface.

16.297.2.7 L4_RPC_NF_OP

```
#define L4_RPC_NF_OP(
                op,
                res,
                name,
                args...  )
```

Value:

```
  struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>        \
  {                                                                          \
    typedef L4::Ipc::Msg::Rpc_call<name##_t, Class, res args> type;          \
    enum { Opcode = (op) };                                                  \
    L4_INLINE_RPC_SRV_FORWARD(name);                                         \
  }
```

Define an RPC call type with specific opcode (the type only, no callable).

Parameters

op	The opcode number for this function
res	The result type of the RPC call
name	The name of the function (`name_t` is used for the type.)
args	The argument list of the RPC function, and RPC attributes (L4::Ipc::Call, L4::Ipc::Call_t etc.).

Definition at line 512 of file ipc_iface.

16.297.2.8 L4_RPC_OP

```
#define L4_RPC_OP(
            op,
            res,
            name,
            args,
            attr...  ) res name args
```

Define an RPC call with specific opcode (type and callable).

Parameters

op	The opcode number for this function
res	The result type of the RPC call
name	The name of the function (`name_t` is used for the type.)
args	The argument list of the RPC function.
attr	Optional RPC attributes (L4::Ipc::Call, L4::Ipc::Call_t etc.).

Definition at line 543 of file ipc_iface.

16.298 ipc_iface

```
// vi:set ft=cpp: -*- Mode: C++ -*-
/*
 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once
#pragma GCC system_header

#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/__typeinfo.h>

```

```
// TODO: add some more documentation
namespace L4 { namespace Ipc {

struct L4_EXPORT Call
{
  enum { Is_call = true };
  enum { Rights = 0 };
  static l4_timeout_t timeout() { return L4_IPC_NEVER; }
};

struct L4_EXPORT Call_zero_send_timeout : Call
{
  static l4_timeout_t timeout() { return L4_IPC_SEND_TIMEOUT_0; }
};

template<unsigned RIGHTS>
struct L4_EXPORT Call_t : Call
{
  enum { Rights = RIGHTS };
};

struct L4_EXPORT Send_only
{
  enum { Is_call = false };
  enum { Rights = 0 };
  static l4_timeout_t timeout() { return L4_IPC_NEVER; }
};

namespace Msg {

template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
struct L4_EXPORT Rpc_inline_call;

template<typename OP, typename CLASS, typename FLAGS, typename R,
         typename ...ARGS>
struct L4_EXPORT Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
{
  template<typename T> struct Result { typedef T result_type; };
  enum
  {
    Return_tag = L4::Types::Same<R, l4_msgtag_t>::value
  };

  typedef Rpc_inline_call type;
  typedef OP op_type;
  typedef CLASS class_type;
  typedef typename Result<R>::result_type result_type;
  typedef R ipc_type (ARGS...);
  typedef result_type func_type (typename _Elem<ARGS>::arg_type...);

  typedef FLAGS flags_type;

  template<typename RES>
  static typename L4::Types::Enable_if< Return_tag, RES >::type
  return_err(long err) noexcept { return l4_msgtag(err, 0, 0, 0); }

  template<typename RES>
  static typename L4::Types::Enable_if< Return_tag, RES >::type
  return_ipc_err(l4_msgtag_t tag, l4_utcb_t const *) noexcept { return tag; }

  template<typename RES>
  static typename L4::Types::Enable_if< Return_tag, RES >::type
  return_code(l4_msgtag_t tag) noexcept { return tag; }

  template<typename RES>
  static typename L4::Types::Enable_if< !Return_tag, RES >::type
  return_err(long err) noexcept { return err; }

  template<typename RES>
  static typename L4::Types::Enable_if< !Return_tag, RES >::type
  return_ipc_err(l4_msgtag_t, l4_utcb_t *utcb) noexcept
  { return l4_ipc_to_errno(l4_ipc_error_code(utcb)); }

  template<typename RES>
  static typename L4::Types::Enable_if< !Return_tag, RES >::type
  return_code(l4_msgtag_t tag) noexcept { return tag.label(); }

  static R call(L4::Cap<class_type> cap,
                typename _Elem<ARGS>::arg_type ...a,
                l4_utcb_t *utcb = l4_utcb()) noexcept;
};

template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
struct L4_EXPORT Rpc_call;

template<typename IPC, typename SIG> struct _Call;

```

```
template<typename IPC, typename R, typename ...ARGS>
struct _Call<IPC, R (ARGS...)>
{
public:
  typedef typename IPC::class_type class_type;
  typedef typename IPC::result_type result_type;

private:
  L4::Cap<class_type> cap() const noexcept
  {
    return L4::Cap<class_type>(reinterpret_cast<l4_cap_idx_t>(this)
                               & L4_CAP_MASK);
  }

public:
  result_type operator () (ARGS ...a, l4_utcb_t *utcb = l4_utcb()) const noexcept
  { return IPC::call(cap(), a..., utcb); }
};

template<typename IPC> struct Call : _Call<IPC, typename IPC::func_type> {};

template<typename OP,
         typename CLASS,
         typename FLAGS,
         typename R,
         typename ...ARGS>
struct L4_EXPORT Rpc_call<OP, CLASS, R (ARGS...), FLAGS> :
  Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
{
  static R call(L4::Cap<CLASS> cap,
                typename _Elem<ARGS>::arg_type ...a,
                l4_utcb_t *utcb = l4_utcb()) noexcept;
};

#define L4_INLINE_RPC_SRV_FORWARD(name)                                  \
  template<typename OBJ> struct fwd                                      \
  {                                                                      \
    OBJ *o;                                                              \
    fwd(OBJ *o) noexcept : o(o) {}                                       \
    template<typename ...ARGS> long call(ARGS ...a) noexcept(noexcept(o->op_##name(a...)))   \
    { return o->op_##name(a...); }                                       \
  }


#define L4_INLINE_RPC_NF(res, name, args...)                             \
  struct name##_t : L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> \
  {                                                                      \
    typedef L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> type;   \
    L4_INLINE_RPC_SRV_FORWARD(name);                                     \
  }

#define L4_INLINE_RPC_NF_OP(op, res, name, args...)                      \
  struct name##_t : L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> \
  {                                                                      \
    typedef L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> type;   \
    enum { Opcode = (op) };                                              \
    L4_INLINE_RPC_SRV_FORWARD(name);                                     \
  }

#ifdef DOXYGEN
#define L4_INLINE_RPC(res, name, args, attr...) res name args
#else
#define L4_INLINE_RPC(res, name, args...)                                \
  L4_INLINE_RPC_NF(res, name, args); L4::Ipc::Msg::Call<name##_t> name
#endif

#ifdef DOXYGEN
#define L4_INLINE_RPC_OP(op, res, name, args, attr...) res name args
#else
#define L4_INLINE_RPC_OP(op, res, name, args...)                         \
  L4_INLINE_RPC_NF_OP(op, res, name, args); L4::Ipc::Msg::Call<name##_t> name
#endif

#define L4_RPC_NF(res, name, args...)                                    \
  struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>    \
  {                                                                      \
    typedef L4::Ipc::Msg::Rpc_call<name##_t, Class, res args> type;      \
    L4_INLINE_RPC_SRV_FORWARD(name);                                     \
  }

#define L4_RPC_NF_OP(op, res, name, args...)                             \
  struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>    \
  {                                                                      \
    typedef L4::Ipc::Msg::Rpc_call<name##_t, Class, res args> type;      \
    enum { Opcode = (op) };                                              \
    L4_INLINE_RPC_SRV_FORWARD(name);                                     \
  }
```

```
#ifdef DOXYGEN
#define L4_RPC(res, name, args, attr...) res name args
#else
#define L4_RPC(res, name, args...)                                          \
  L4_RPC_NF(res, name, args); L4::Ipc::Msg::Call<name##_t> name
#endif

#ifdef DOXYGEN
#define L4_RPC_OP(op, res, name, args, attr...) res name args
#else
#define L4_RPC_OP(op, res, name, args...)                                   \
  L4_RPC_NF_OP(op, res, name, args); L4::Ipc::Msg::Call<name##_t> name
#endif


namespace Detail {

template<typename ...ARGS>
struct Buf
{
public:
  template<typename DIR>
  static constexpr int write(char *, int offset, int) noexcept
  { return offset; }

  template<typename DIR>
  static constexpr int read(char *, int offset, int, long) noexcept
  { return offset; }

  typedef void Base;
};

template<typename A, typename ...M>
struct Buf<A, M...> : Buf<M...>
{
  typedef Buf<M...> Base;

  typedef Clnt_xmit<A> xmit;
  typedef typename _Elem<A>::arg_type arg_type;
  typedef Detail::_Plain<arg_type> plain;

  template<typename DIR>
  static int
  write(char *base, int offset, int limit,
        arg_type a, typename _Elem<M>::arg_type ...m) noexcept
  {
    offset = xmit::to_msg(base, offset, limit, plain::deref(a),
                          typename DIR::dir(), typename DIR::cls());
    return Base::template write<DIR>(base, offset, limit, m...);
  }

  template<typename DIR>
  static int
  read(char *base, int offset, int limit, long ret,
       arg_type a, typename _Elem<M>::arg_type ...m) noexcept
  {
    int r = xmit::from_msg(base, offset, limit, ret, plain::deref(a),
                           typename DIR::dir(), typename DIR::cls());
    if (L4_LIKELY(r >= 0))
      return Base::template read<DIR>(base, r, limit, ret, m...);

    if (_Elem<A>::Is_optional)
      return Base::template read<DIR>(base, offset, limit, ret, m...);

    return r;
  }
};

template <typename ...ARGS> struct _Part
{
  typedef Buf<ARGS...> Data;

  template<typename DIR>
  static int write(void *b, int offset, int limit,
                       typename _Elem<ARGS>::arg_type ...m) noexcept
  {
    int r = Data::template write<DIR>((char *)b, offset, limit, m...);
    if (L4_LIKELY(r >= offset))
      return r - offset;
    return r;
  }

  template<typename DIR>
  static int read(void *b, int offset, int limit, long ret,
                      typename _Elem<ARGS>::arg_type ...m) noexcept
  {
```

```
    int r = Data::template read<DIR>((char *)b, offset, limit, ret, m...);
    if (L4_LIKELY(r >= offset))
      return r - offset;
    return r;
  }
};

template<typename IPC_TYPE, typename OPCODE = void>
struct Part;

// The version without an op-code
template<typename R, typename ...ARGS>
struct Part<R (ARGS...), void> : _Part<ARGS...>
{
  typedef Buf<ARGS...> Data;

  // write arguments, skipping the dummy opcode
  template<typename DIR>
  static int write_op(void *b, int offset, int limit,
                      int /*placeholder for op*/,
                      typename _Elem<ARGS>::arg_type ...m) noexcept
  {
    int r = Data::template write<DIR>((char *)b, offset, limit, m...);
    if (L4_LIKELY(r >= offset))
      return r - offset;
    return r;
  }
};

// Message part with additional opcode
template<typename OPCODE, typename R, typename ...ARGS>
struct Part<R (ARGS...), OPCODE> : _Part<ARGS...>
{
  typedef OPCODE opcode_type;
  typedef Buf<opcode_type, ARGS...> Data;

  // write arguments, including the opcode
  template<typename DIR>
  static int write_op(void *b, int offset, int limit,
                      opcode_type op, typename _Elem<ARGS>::arg_type ...m) noexcept
  {
    int r = Data::template write<DIR>((char *)b, offset, limit, op, m...);
    if (L4_LIKELY(r >= offset))
      return r - offset;
    return r;
  }
};


} // namespace Detail

//-----------------------------------------------------
// Implementation of the RPC call
// TODO: Add support for timeout via special RPC argument
// TODO: Add support for passing the UTCB pointer as argument
//
template<typename OP, typename CLASS, typename FLAGS, typename R,
         typename ...ARGS>
inline R
Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>::
  call(L4::Cap<CLASS> cap,
       typename _Elem<ARGS>::arg_type ...a,
       l4_utcb_t *utcb) noexcept
{
  using namespace Ipc::Msg;

  typedef typename Kobject_typeid<CLASS>::Iface::Rpcs Rpcs;
  typedef typename Rpcs::template Rpc<OP> Opt;
  typedef Detail::Part<ipc_type, typename Rpcs::opcode_type> Args;

  l4_msg_regs_t *mrs = l4_utcb_mr_u(utcb);

  // handle in-data part of the arguments
  int send_bytes =
    Args::template write_op<Do_in_data>(mrs->mr, 0, Mr_bytes,
                                        Opt::Opcode, a...);

  if (L4_UNLIKELY(send_bytes < 0))
    return return_err<R>(send_bytes);

  send_bytes = align_to<l4_umword_t>(send_bytes);
  int const send_words = send_bytes / Word_bytes;
  // write the in-items part of the message if there is one
  int item_bytes =
    Args::template write<Do_in_items>(&mrs->mr[send_words], 0,
                                      Mr_bytes - send_bytes, a...);

```

```
  if (L4_UNLIKELY(item_bytes < 0))
    return return_err<R>(item_bytes);

  int send_items = item_bytes / Item_bytes;

    {
      // setup the receive buffers for the RPC call
      l4_buf_regs_t *brs = l4_utcb_br_u(utcb);
      // XXX: we currently support only one type of receive buffers per call
      brs->bdr = 0; // we always start at br[0]

      // the limit leaves us at least one register for the zero terminator
      // add the buffers given as arguments to the buffer registers
      int bytes =
        Args::template write<Do_rcv_buffers>(brs->br, 0, Br_bytes - Word_bytes,
                                             a...);

      if (L4_UNLIKELY(bytes < 0))
        return return_err<R>(bytes);

      brs->br[bytes / Word_bytes] = 0;
    }


  // here we do the actual IPC --------------------------------
  l4_msgtag_t t;
  t = l4_msgtag(CLASS::Protocol, send_words, send_items, 0);
  // do the call (Q: do we need support for timeouts?)
  if (flags_type::Is_call)
    t = l4_ipc_call(cap.cap(), utcb, t, flags_type::timeout());
  else
    {
      t = l4_ipc_send(cap.cap(), utcb, t, flags_type::timeout());
      if (L4_UNLIKELY(t.has_error()))
        return return_ipc_err<R>(t, utcb);

      return return_code<R>(l4_msgtag(0, 0, 0, t.flags()));
    }

  // unmarshalling starts here --------------------------------

  // bail out early in the case of an IPC error
  if (L4_UNLIKELY(t.has_error()))
    return return_ipc_err<R>(t, utcb);

  // take the label as return value
  long r = t.label();

  // bail out on negative error codes too
  if (L4_UNLIKELY(r < 0))
    return return_err<R>(r);

  int const rcv_bytes = t.words() * Word_bytes;

  // read the static out-data values to the arguments
  int err = Args::template read<Do_out_data>(mrs->mr, 0, rcv_bytes, r, a...);

  int const item_limit = t.items() * Item_bytes;

  if (L4_UNLIKELY(err < 0 || item_limit > Mr_bytes))
    return return_err<R>(-L4_EMSGTOOSHORT);

  // read the static out-items to the arguments
  err = Args::template read<Do_out_items>(&mrs->mr[t.words()], 0, item_limit,
                                          r, a...);

  if (L4_UNLIKELY(err < 0))
    return return_err<R>(-L4_EMSGTOOSHORT);

  return return_code<R>(t);
}

} // namespace Msg
} // namespace Ipc
} // namespace L4
```

16.299 l4/cxx/ipc_server File Reference

IPC server loop.

```
#include <l4/sys/capability>
#include <l4/sys/typeinfo_svr>
#include <l4/sys/err.h>
#include <l4/cxx/ipc_stream>
#include <l4/sys/cxx/ipc_epiface>
#include <l4/sys/cxx/ipc_server_loop>
#include <l4/cxx/type_traits>
#include <l4/cxx/exceptions>
```

Include dependency graph for ipc_server:

Data Structures

- class L4::Server_object

  Abstract server object to be used with L4::Server and L4::Basic_registry.
- struct L4::Server_object_t< IFACE, BASE >

  Base class (template) for server implementing server objects.
- struct L4::Server_object_x< Derived, IFACE, BASE >

  Helper class to implement p_dispatch based server objects.
- struct L4::Irq_handler_object

  Server object base class for handling IRQ messages.

Namespaces

- L4

  L4 low-level kernel interface.

16.299.1 Detailed Description

IPC server loop.

Definition in file ipc_server.

16.300 ipc_server

```
// vi:set ft=cpp: -*- Mode: C++ -*-
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/sys/capability>
#include <l4/sys/typeinfo_svr>
#include <l4/sys/err.h>
#include <l4/cxx/ipc_stream>
#include <l4/sys/cxx/ipc_epiface>
#include <l4/sys/cxx/ipc_server_loop>
#include <l4/cxx/type_traits>
#include <l4/cxx/exceptions>

namespace L4 {

class Server_object : public Epiface
{
public:
  virtual int dispatch(unsigned long rights, Ipc::Iostream &ios) = 0;

  l4_msgtag_t dispatch(l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb)
  {
    L4::Ipc::Iostream ios(utcb);
    ios.tag() = tag;
    int r = dispatch(rights, ios);
    return ios.prepare_ipc(r);
  }

  Cap<Kobject> obj_cap() const
  { return cap_cast<Kobject>(Epiface::obj_cap()); }
};

template<typename IFACE, typename BASE = L4::Server_object>
struct Server_object_t : BASE
{
  typedef IFACE Interface;

  typename BASE::Demand get_buffer_demand() const
  { return typename L4::Kobject_typeid<IFACE>::Demand(); }

  int dispatch_meta_request(L4::Ipc::Iostream &ios)
  { return L4::Util::handle_meta_request<IFACE>(ios); }

  template<typename THIS>
  static int proto_dispatch(THIS *self, l4_umword_t rights, L4::Ipc::Iostream &ios)
  {
    l4_msgtag_t t;
    ios » t;
    return Kobject_typeid<IFACE>::proto_dispatch(self, t.label(), rights, ios);
  }
};

template<typename Derived, typename IFACE, typename BASE = L4::Server_object>
struct Server_object_x : Server_object_t<IFACE, BASE>
{
  int dispatch(l4_umword_t r, L4::Ipc::Iostream &ios)
  {
    return Server_object_t<IFACE, BASE>::proto_dispatch(static_cast<Derived *>(this),
                                                        r, ios);
  }
};

struct Irq_handler_object : Server_object_t<Kobject> {};

}
```

16.301 l4/sys/cxx/ipc_types File Reference

```
#include "capability.h"
#include "types"
#include "ipc_basics"
```

Include dependency graph for ipc_types:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct L4::Ipc::In_out< T >

  Mark an argument as in-out argument.
- struct L4::Ipc::As_value< T >

  Pass the argument as plain data value.
- struct L4::Ipc::Opt< T >

  Attribute for defining an optional RPC argument.
- class L4::Ipc::Small_buf

  A receive item for receiving a single capability.

- class L4::Ipc::Snd_item

  RPC wrapper for a send item.
- class L4::Ipc::Buf_item

  RPC warpper for a receive item.
- class L4::Ipc::Gen_fpage< T >

  Generic RPC wrapper for L4 flex-pages.
- class L4::Ipc::Cap< T >

  Capability type for RPC interfaces (see `L4::Cap<T>`).

Namespaces

- L4

  L4 low-level kernel interface.
- L4::Ipc

  IPC related functionality.
- L4::Ipc::Msg

  IPC Message related functionality.

Typedefs

- typedef Gen_fpage< Snd_item > L4::Ipc::Snd_fpage

  Send flex-page.
- typedef Gen_fpage< Buf_item > L4::Ipc::Rcv_fpage

  Rcv flex-page.

Functions

- template<typename T >
  Cap< T > L4::Ipc::make_cap (L4::Cap< T > cap, unsigned rights) noexcept

  Make an L4::Ipc::Cap<T> for the given capability and rights.
- template<typename T >
  Cap< T > L4::Ipc::make_cap_rw (L4::Cap< T > cap) noexcept

  Make an L4::Ipc::Cap<T> for the given capability with L4_CAP_FPAGE_RW rights.
- template<typename T >
  Cap< T > L4::Ipc::make_cap_rws (L4::Cap< T > cap) noexcept

  Make an L4::Ipc::Cap<T> for the given capability with L4_CAP_FPAGE_RWS rights.
- template<typename T >
  Cap< T > L4::Ipc::make_cap_full (L4::Cap< T > cap) noexcept

  Make an L4::IPC::Cap<T> for the given capability with full fpage and object-specific rights.

16.302 ipc_types

```
// vi:set ft=cpp: -*- Mode: C++ -*-
/*
 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include "capability.h"
#include "types"
#include "ipc_basics"
namespace L4 {

typedef int Opcode;

namespace Ipc {

template<typename T> struct L4_EXPORT Out;


template<typename T> struct  L4_EXPORT In_out
{
  T v;
  In_out() {}
  In_out(T v) : v(v) {}
  operator T () const { return v; }
  operator T & () { return v; }
};

namespace Msg {
template<typename A> struct Elem< In_out<A *> > : Elem<A *> {};

template<typename A>
struct Svr_xmit< In_out<A *> > : Svr_xmit<A *>, Svr_xmit<A const *>
{
  using Svr_xmit<A *>::from_svr;
  using Svr_xmit<A const *>::to_svr;
};

template<typename A>
struct Clnt_xmit< In_out<A *> > : Clnt_xmit<A *>, Clnt_xmit<A const *>
{
  using Clnt_xmit<A *>::from_msg;
  using Clnt_xmit<A const *>::to_msg;
};

template<typename A>
struct Is_valid_rpc_type< In_out<A *> > : Is_valid_rpc_type<A *> {};
template<typename A>
struct Is_valid_rpc_type< In_out<A const *> > : L4::Types::False {};

#ifdef CONFIG_ALLOW_REFS
template<typename A> struct Elem< In_out<A &> > : Elem<A &> {};

template<typename A>
struct Svr_xmit< In_out<A &> > : Svr_xmit<A &>, Svr_xmit<A const &>
{
  using Svr_xmit<A &>::from_svr;
  using Svr_xmit<A const &>::to_svr;
};

template<typename A>
struct Clnt_xmit< In_out<A &> > : Clnt_xmit<A &>, Clnt_xmit<A const &>
{
  using Clnt_xmit<A &>::from_msg;
  using Clnt_xmit<A const &>::to_msg;
};

template<typename A>
struct Is_valid_rpc_type< In_out<A &> > : Is_valid_rpc_type<A &> {};
template<typename A>
struct Is_valid_rpc_type< In_out<A const &> > : L4::Types::False {};

#else
```

```
template<typename A>
struct Is_valid_rpc_type< In_out<A &> > : L4::Types::False {};

#endif

// Value types don't make sense for output.
template<typename A>
struct Is_valid_rpc_type< In_out<A> > : L4::Types::False {};

}


template<typename T> struct L4_EXPORT As_value
{
  typedef T value_type;
  T v;
  As_value() noexcept {}
  As_value(T v) noexcept : v(v) {}
  operator T () const noexcept { return v; }
  operator T & () noexcept { return v; }
};

namespace Msg {
template<typename T> struct Class< As_value<T> > : Cls_data {};
template<typename T> struct Elem< As_value<T> > : Elem<T> {};
template<typename T> struct Elem< As_value<T> *> : Elem<T *> {};
}


template<typename T> struct L4_EXPORT Opt
{
  T _value;
  bool _valid;

  Opt() noexcept : _valid(false) {}

  Opt(T value) noexcept : _value(value), _valid(true) {}

  Opt &operator = (T value) noexcept
  {
    this->_value = value;
    this->_valid = true;
    return *this;
  }

  void set_valid(bool valid = true) noexcept { _valid = valid; }

  T *operator -> () noexcept { return &this->_value; }
  T const *operator -> () const noexcept { return &this->_value; }
  T value() const noexcept { return this->_value; }
  T &value() noexcept { return this->_value; }
  bool is_valid() const noexcept { return this->_valid; }
};

namespace Msg {
template<typename T> struct Elem< Opt<T &> > : Elem<T &>
{
  enum { Is_optional = true };
  typedef Opt<typename Elem<T &>::svr_type> &svr_arg_type;
  typedef Opt<typename Elem<T &>::svr_type> svr_type;
};

template<typename T> struct Elem< Opt<T *> > : Elem<T *>
{
  enum { Is_optional = true };
  typedef Opt<typename Elem<T *>::svr_type> &svr_arg_type;
  typedef Opt<typename Elem<T *>::svr_type> svr_type;
};



template<typename T, typename CLASS>
struct Svr_val_ops<Opt<T>, Dir_out, CLASS> : Svr_noops< Opt<T> >
{
  typedef Opt<T> svr_type;
  typedef Svr_val_ops<T, Dir_out, CLASS> Native;

  using Svr_noops< Opt<T> >::to_svr;
  static int to_svr(char *msg, unsigned offset, unsigned limit,
                    Opt<T> &arg, Dir_out, CLASS) noexcept
  {
    return Native::to_svr(msg, offset, limit, arg.value(), Dir_out(), CLASS());
  }

  using Svr_noops< Opt<T> >::from_svr;
  static int from_svr(char *msg, unsigned offset, unsigned limit, long ret,
```

```
                      svr_type &arg, Dir_out, CLASS) noexcept
  {
    if (arg.is_valid())
      return Native::from_svr(msg, offset, limit, ret, arg.value(),
                              Dir_out(), CLASS());
    return offset;
  }
};

template<typename T> struct Elem< Opt<T> > : Elem<T>
{
  enum { Is_optional = true };
  typedef Opt<T> arg_type;
};

template<typename T> struct Elem< Opt<T const *> > : Elem<T const *>
{
  enum { Is_optional = true };
  typedef Opt<T const *> arg_type;
};

template<typename T>
struct Is_valid_rpc_type< Opt<T const &> > : L4::Types::False {};

template<typename T, typename CLASS>
struct Clnt_val_ops<Opt<T>, Dir_in, CLASS> : Clnt_noops< Opt<T> >
{
  typedef Opt<T> arg_type;
  typedef Detail::_Clnt_val_ops<typename Elem<T>::arg_type, Dir_in, CLASS> Native;

  using Clnt_noops< Opt<T> >::to_msg;
  static int to_msg(char *msg, unsigned offset, unsigned limit,
                    arg_type arg, Dir_in, CLASS) noexcept
  {
    if (arg.is_valid())
      return Native::to_msg(msg, offset, limit,
                            Detail::_Plain<T>::deref(arg.value()),
                            Dir_in(), CLASS());
    return offset;
  }
};

template<typename T> struct Class< Opt<T> > :
  Class< typename Detail::_Plain<T>::type > {};
template<typename T> struct Direction< Opt<T> > : Direction<T> {};
}

class  L4_EXPORT Small_buf
{
public:
  explicit Small_buf(L4::Cap<void> cap, unsigned long flags = 0) noexcept
  : _data(cap.cap() | L4_RCV_ITEM_SINGLE_CAP | flags) {}

  explicit Small_buf(l4_cap_idx_t cap, unsigned long flags = 0) noexcept
  : _data(cap | L4_RCV_ITEM_SINGLE_CAP | flags) {}

  l4_umword_t raw() const noexcept { return _data; }
private:
  l4_umword_t _data;
};

class Snd_item
{
public:
  Snd_item(l4_umword_t base, l4_umword_t data) noexcept : _base(base), _data(data) {}

protected:
  l4_umword_t _base;
  l4_umword_t _data;
};

class Buf_item
{
public:
  Buf_item(l4_umword_t base, l4_umword_t data) noexcept : _base(base), _data(data) {}

protected:
  l4_umword_t _base;
  l4_umword_t _data;
};

template< typename T >
class  L4_EXPORT Gen_fpage : public T
{
public:
  enum Type
  {
```

```
    Special = L4_FPAGE_SPECIAL « 4,
    Memory  = L4_FPAGE_MEMORY « 4,
    Io      = L4_FPAGE_IO « 4,
    Obj     = L4_FPAGE_OBJ « 4
  };

  enum Map_type
  {
    Map   = L4_MAP_ITEM_MAP,
    Grant = L4_MAP_ITEM_GRANT,
  };

  enum Cacheopt
  {
    None     = 0,
    Cached   = L4_FPAGE_CACHEABLE « 4,
    Buffered = L4_FPAGE_BUFFERABLE « 4,
    Uncached = L4_FPAGE_UNCACHEABLE « 4
  };

  enum Continue
  {
    Single   = 0,
    Last     = 0,
    More     = L4_ITEM_CONT,
    Compound = L4_ITEM_CONT,
  };

private:
  Gen_fpage(l4_umword_t d, l4_umword_t fp) noexcept : T(d, fp) {}

  Gen_fpage(Type type, l4_addr_t base, int order,
            unsigned char rights,
            l4_addr_t snd_base,
            Map_type map_type,
            Cacheopt cache, Continue cont) noexcept
  : T(L4_ITEM_MAP | (snd_base & (~0UL « 10)) | l4_umword_t(map_type) | l4_umword_t(cache)
      | l4_umword_t(cont),
      base | l4_umword_t(type) | rights | (l4_umword_t(order) « 6))
  {}

public:
  Gen_fpage() noexcept : T(0, 0) {}
  Gen_fpage(l4_fpage_t const &fp, l4_addr_t snd_base = 0,
            Map_type map_type = Map,
            Cacheopt cache = None, Continue cont = Last) noexcept
  : T(L4_ITEM_MAP | (snd_base & (~0UL « 10)) | l4_umword_t(map_type) | l4_umword_t(cache)
      | l4_umword_t(cont),
      fp.raw)
  {}

  Gen_fpage(L4::Cap<void> cap, unsigned rights, Map_type map_type = Map) noexcept
  : T(L4_ITEM_MAP | l4_umword_t(map_type) | (rights & 0xf0),
      cap.fpage(rights).raw)
  {}

  static Gen_fpage<T> obj(l4_addr_t base, int order,
                          unsigned char rights,
                          l4_addr_t snd_base = 0,
                          Map_type map_type = Map,
                          Continue cont = Last) noexcept
  {
    return Gen_fpage<T>(Obj, base « 12, order, rights, snd_base, map_type, None, cont);
  }

  static Gen_fpage<T> mem(l4_addr_t base, int order,
                          unsigned char rights,
                          l4_addr_t snd_base = 0,
                          Map_type map_type = Map,
                          Cacheopt cache = None, Continue cont = Last) noexcept
  {
    return Gen_fpage<T>(Memory, base, order, rights, snd_base,
                        map_type, cache, cont);
  }

  static Gen_fpage<T> rmem(l4_addr_t base, int order, l4_addr_t snd_base,
                           unsigned char rights, unsigned cap_br) noexcept
  {
    return Gen_fpage<T>(
        L4_ITEM_MAP | (snd_base & (~0UL « 10)) | l4_umword_t(Map)
        | l4_umword_t(None) | l4_umword_t(Compound) | (cap_br « 8),

        base | l4_umword_t(Memory) | rights | (l4_umword_t(order) « 6));
  }

  static Gen_fpage<T> io(l4_addr_t base, int order,
                         unsigned char rights,
```

```
                          l4_addr_t snd_base = 0,
                          Map_type map_type = Map,
                          Continue cont = Last) noexcept
  {
    return Gen_fpage<T>(Io, base « 12, order, rights, snd_base, map_type, None, cont);
  }

  unsigned order() const noexcept { return (T::_data » 6) & 0x3f; }
  unsigned snd_order() const noexcept { return (T::_data » 6) & 0x3f; }
  unsigned rcv_order() const noexcept { return (T::_base » 6) & 0x3f; }
  l4_addr_t base() const noexcept { return T::_data & (~0UL « 12); }
  l4_addr_t snd_base() const noexcept { return T::_base & (~0UL « 10); }
  void snd_base(l4_addr_t b) noexcept { T::_base = (T::_base & ~(~0UL « 10)) | (b & (~0UL « 10)); }

  bool is_valid() const noexcept { return T::_base & L4_ITEM_MAP; }
  bool cap_received() const noexcept { return (T::_base & 0x3e) == 0x38; }
  bool id_received() const noexcept { return (T::_base & 0x3e) == 0x3c; }
  bool local_id_received() const noexcept { return (T::_base & 0x3e) == 0x3e; }

  bool is_compound() const noexcept { return T::_base & 1; }
  l4_umword_t data() const noexcept { return T::_data; }
  l4_umword_t base_x() const noexcept { return T::_base; }
};


typedef Gen_fpage<Snd_item> Snd_fpage;
typedef Gen_fpage<Buf_item> Rcv_fpage;

#ifdef L4_CXX_IPC_SUPPORT_STRINGS
template <typename T, typename B>
class Gen_string : public T
{
public:
  Gen_string() noexcept : T(0, 0) {}
  Gen_string(B buf, unsigned long size) noexcept
  : T(size « 10, l4_umword_t(buf))
  {}

  unsigned long len() const noexcept { return T::_base » 10; }
};

typedef Gen_string<Snd_item, void const *> Snd_string;
typedef Gen_string<Buf_item, void *> Rcv_string;
#endif


namespace Msg {

// Snd_fpage are out items
template<> struct Class<L4::Ipc::Snd_fpage> : Cls_item {};

// Rcv_fpage are buffer items
template<> struct Class<L4::Ipc::Rcv_fpage> : Cls_buffer {};

// Remove receive buffers from server-side arguments
template<> struct Elem<L4::Ipc::Rcv_fpage>
{
  typedef L4::Ipc::Rcv_fpage arg_type;
  typedef void svr_type;
  typedef void svr_arg_type;
  enum { Is_optional = false };
};

// Rcv_fpage are buffer items
template<> struct Class<L4::Ipc::Small_buf> : Cls_buffer {};

// Remove receive buffers from server-side arguments
template<> struct Elem<L4::Ipc::Small_buf>
{
  typedef L4::Ipc::Small_buf arg_type;
  typedef void svr_type;
  typedef void svr_arg_type;
  enum { Is_optional = false };
};
} // namespace Msg

// L4::Cap<> handling

template<typename T> class Cap
{
  template<typename O> friend class Cap;
  l4_umword_t _cap_n_rights;

public:
  enum
  {
    Rights_mask = 0xff,
```

```
00555
00560     Cap_mask    = L4_CAP_MASK
00561   };
00562
00564   template<typename O>
00565   Cap(Cap<O> const &o) noexcept : _cap_n_rights(o._cap_n_rights)
00566   { T *x = (O*)1; (void)x; }
00567
00569   Cap(L4::Cap<T> cap) noexcept
00570   : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00571   {}
00572
00574   template<typename O>
00575   Cap(L4::Cap<O> cap) noexcept
00576   : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00577   { T *x = (O*)1; (void)x; }
00578
00580   Cap() noexcept : _cap_n_rights(L4_INVALID_CAP) {}
00581
00589   Cap(L4::Cap<T> cap, unsigned char rights) noexcept
00590   : _cap_n_rights((cap.cap() & Cap_mask) | (rights & Rights_mask)) {}
00591
00597   static Cap from_ci(l4_cap_idx_t c) noexcept
00598   { return Cap(L4::Cap<T>(c & Cap_mask), c & Rights_mask); }
00599
00601   L4::Cap<T> cap() const noexcept
00602   { return L4::Cap<T>(_cap_n_rights & Cap_mask); }
00603
00605   unsigned rights() const noexcept
00606   { return _cap_n_rights & Rights_mask; }
00607
00609   L4::Ipc::Snd_fpage fpage() const noexcept
00610   { return L4::Ipc::Snd_fpage(cap(), rights()); }
00611
00613   bool is_valid() const noexcept
00614   { return !(_cap_n_rights & L4_INVALID_CAP_BIT); }
00615 };
00616
00623 template<typename T>
00624 Cap<T> make_cap(L4::Cap<T> cap, unsigned rights) noexcept
00625 { return Cap<T>(cap, rights); }
00626
00633 template<typename T>
00634 Cap<T> make_cap_rw(L4::Cap<T> cap) noexcept
00635 { return Cap<T>(cap, L4_CAP_FPAGE_RW); }
00636
00643 template<typename T>
00644 Cap<T> make_cap_rws(L4::Cap<T> cap) noexcept
00645 { return Cap<T>(cap, L4_CAP_FPAGE_RWS); }
00646
00661 template<typename T>
00662 Cap<T> make_cap_full(L4::Cap<T> cap) noexcept
00663 { return Cap<T>(cap, L4_CAP_FPAGE_RWSD | L4_FPAGE_C_OBJ_RIGHTS); }
00664
00665 // caps are special the have an invalid representation
00666 template<typename T> struct L4_EXPORT Opt< Cap<T> >
00667 {
00668   Cap<T> _value;
00669   Opt() noexcept {}
00670   Opt(Cap<T> value) noexcept : _value(value) {}
00671   Opt(L4::Cap<T> value) noexcept : _value(value) {}
00672   Opt &operator = (Cap<T> value) noexcept
00673   { this->_value = value; }
00674   Opt &operator = (L4::Cap<T> value) noexcept
00675   { this->_value = value; }
00676
00677   Cap<T> value() const noexcept { return this->_value; }
00678   bool is_valid() const noexcept { return this->_value.is_valid(); }
00679 };
00680
00681
00682 namespace Msg {
00683 // prohibit L4::Cap as argument
00684 template<typename A>
00685 struct Is_valid_rpc_type< L4::Cap<A> > : L4::Types::False {};
00686
00687 template<typename A> struct Class< Cap<A> > : Cls_item {};
00688 template<typename A> struct Elem< Cap<A> >
00689 {
00690   enum { Is_optional = false };
00691   typedef Cap<A> arg_type;
00692   typedef L4::Ipc::Snd_fpage svr_type;
00693   typedef L4::Ipc::Snd_fpage svr_arg_type;
00694 };
00695
00696
00697 template<typename A, typename CLASS>
```

```
struct Svr_val_ops<Cap<A>, Dir_in, CLASS> :
  Svr_val_ops<L4::Ipc::Snd_fpage, Dir_in, CLASS>
{};

template<typename A, typename CLASS>
struct Clnt_val_ops<Cap<A>, Dir_in, CLASS> :
  Clnt_noops< Cap<A> >
{
  using Clnt_noops< Cap<A> >::to_msg;

  static int to_msg(char *msg, unsigned offset, unsigned limit,
                    Cap<A> arg, Dir_in, Cls_item) noexcept
  {
    // passing an invalid cap as mandatory argument is an error
    // XXX: This checks for a client calling error, we could
    //      also just ignore this for performance reasons and
    //      let the client fail badly (Alex: I'd prefer this)
    if (L4_UNLIKELY(!arg.is_valid()))
      return -L4_EMSGMISSARG;

    return msg_add(msg, offset, limit, arg.fpage());
  }
};

template<typename A>
struct Elem<Out<L4::Cap<A> > >
{
  enum { Is_optional = false };
  typedef L4::Cap<A> arg_type;
  typedef Ipc::Cap<A> svr_type;
  typedef svr_type &svr_arg_type;
};

template<typename A> struct Direction< Out< L4::Cap<A> > > : Dir_out {};
template<typename A> struct Class< Out< L4::Cap<A> > > : Cls_item {};

template<typename A>
struct Clnt_val_ops< L4::Cap<A>, Dir_out, Cls_item > :
  Clnt_noops< L4::Cap<A> >
{
  using Clnt_noops< L4::Cap<A> >::to_msg;
  static int to_msg(char *msg, unsigned offset, unsigned limit,
                    L4::Cap<A> arg, Dir_in, Cls_buffer) noexcept
  {
    if (L4_UNLIKELY(!arg.is_valid()))
      return -L4_EMSGMISSARG; // no buffer inserted
    return msg_add(msg, offset, limit, Small_buf(arg));
  }
};

template<typename A>
struct Svr_val_ops< L4::Ipc::Cap<A>, Dir_out, Cls_item > :
  Svr_noops<Cap<A> &>
{
  using Svr_noops<Cap<A> &>::from_svr;
  static int from_svr(char *msg, unsigned offset, unsigned limit, long,
                      Cap<A> arg, Dir_out, Cls_item) noexcept
  {
    if (L4_UNLIKELY(!arg.is_valid()))
      // do not map anything
      return msg_add(msg, offset, limit, L4::Ipc::Snd_fpage(arg.cap(), 0));

    return msg_add(msg, offset, limit, arg.fpage());
  }
};

// prohibit a UTCB pointer as normal RPC argument
template<> struct Is_valid_rpc_type<l4_utcb_t *> : L4::Types::False {};

} // namespace Msg
} // namespace Ipc
} // namespace L4
```

16.303 l4/sys/cxx/smart_capability_1x File Reference

```
#include <l4/sys/capability>
```

Include dependency graph for smart_capability_1x:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4

  L4 low-level kernel interface.

16.304 smart_capability_1x

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
```

```
/*
 * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */

#pragma once

#include <l4/sys/capability>

namespace L4 { namespace Detail {

template< typename T, typename IMPL >
class Smart_cap_base : public Cap_base, protected IMPL
{
protected:
  template<typename X>
  static IMPL &impl(Smart_cap_base<X, IMPL> &o) { return o; }

  template<typename X>
  static IMPL const &impl(Smart_cap_base<X, IMPL> const &o) { return o; }

public:
  template<typename X, typename I>
  friend class ::L4::Detail::Smart_cap_base;

  Smart_cap_base(Smart_cap_base const &) = delete;
  Smart_cap_base &operator = (Smart_cap_base const &) = delete;

  Smart_cap_base() noexcept : Cap_base(Invalid) {}

  explicit Smart_cap_base(Cap_base::Cap_type t) noexcept
  : Cap_base(t)
  {}

  template<typename O>
  explicit constexpr Smart_cap_base(Cap<O> c) noexcept
  : Cap_base(c.cap())
  {}

  template<typename O>
  explicit constexpr Smart_cap_base(Cap<O> c, IMPL const &impl) noexcept
  : Cap_base(c.cap()), IMPL(impl)
  {}

  Cap<T> release() noexcept
  {
    l4_cap_idx_t c = this->cap();
    IMPL::invalidate(*this);
    return Cap<T>(c);
  }

  void reset()
  { IMPL::free(*this); }

  Cap<T> operator -> () const noexcept { return Cap<T>(this->cap()); }
  Cap<T> get() const noexcept { return Cap<T>(this->cap()); }
  ~Smart_cap_base() noexcept { IMPL::free(*this); }
};


template< typename T, typename IMPL >
class Unique_cap_impl final :  public Smart_cap_base<T, IMPL>
{
private:
  typedef Smart_cap_base<T, IMPL> Base;

public:
  using Base::Base;
  Unique_cap_impl() noexcept = default;

  Unique_cap_impl(Unique_cap_impl &&o) noexcept
  : Base(o.release(), Base::impl(o))
  {}

  template<typename O>
```

```
  Unique_cap_impl(Unique_cap_impl<O, IMPL> &&o) noexcept
  : Base(o.release(), Base::impl(o))
  { T* __t = ((O*)100); (void)__t; }

  Unique_cap_impl &operator = (Unique_cap_impl &&o) noexcept
  {
    if (&o == this)
      return *this;

    IMPL::free(*this);
    this->_c = o.release().cap();
    this->IMPL::operator = (Base::impl(o));
    return *this;
  }

  template<typename O>
  Unique_cap_impl &operator = (Unique_cap_impl<O, IMPL> &&o) noexcept
  {
    T* __t = ((O*)100); (void)__t;

    IMPL::free(*this);
    this->_c = o.release().cap();
    this->IMPL::operator = (Base::impl(o));
    return *this;
  }
};

template<typename T, typename IMPL>
class Shared_cap_impl final : public Smart_cap_base<T, IMPL>
{
private:
  typedef Smart_cap_base<T, IMPL> Base;

public:
  using Base::Base;
  Shared_cap_impl() noexcept = default;

  Shared_cap_impl(Shared_cap_impl &&o) noexcept
  : Base(o.release())
  {}

  template<typename O>
  Shared_cap_impl(Shared_cap_impl<O, IMPL> &&o) noexcept
  : Base(o.release())
  { T* __t = ((O*)100); (void)__t; }

  Shared_cap_impl &operator = (Shared_cap_impl &&o) noexcept
  {
    if (&o == this)
      return *this;

    IMPL::free(*this);
    this->_c = o.release().cap();
    this->IMPL::operator = (Base::impl(o));
    return *this;
  }

  template<typename O>
  Shared_cap_impl &operator = (Shared_cap_impl<O, IMPL> &&o) noexcept
  {
    T* __t = ((O*)100); (void)__t;

    IMPL::free(*this);
    this->_c = o.release().cap();
    this->IMPL::operator = (Base::impl(o));
    return *this;
  }

  Shared_cap_impl(Shared_cap_impl const &o) noexcept
  : Base(L4::Cap<T>(IMPL::copy(o).cap()))
  {}

  template<typename O>
  Shared_cap_impl(Shared_cap_impl<O, IMPL> const &o) noexcept
  : Base(IMPL::copy(o))
  { T* __t = ((O*)100); (void)__t; }

  Shared_cap_impl &operator = (Shared_cap_impl const &o) noexcept
  {
    if (&o == this)
      return *this;

    IMPL::free(*this);
    this->IMPL::operator = (static_cast<IMPL const &>(o));
    this->_c = this->IMPL::copy(o).cap();
    return *this;
  }
```

```
  template<typename O>
  Shared_cap_impl &operator = (Shared_cap_impl<O, IMPL> const &o) noexcept
  {
    T* __t = ((O*)100); (void)__t;
    IMPL::free(*this);
    this->IMPL::operator = (static_cast<IMPL const &>(o));
    this->_c = this->IMPL::copy(o).cap();
    return *this;
  }
};

}} // L4::Detail
```

16.305 l4/sys/cxx/types File Reference

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Types::Flags< BITS_ENUM, UNDERLYING >

  Template for defining typical Flags bitmaps.
- struct L4::Types::Int_for_type< T >

  Metafunction to get an integral type of the same size as `T`.
- struct L4::Types::Flags_ops_t< DT >

  Mixin class to define a set of friend bitwise operators on `DT`.
- struct L4::Types::Flags_t< DT, T >

  Template type to define a flags type with bitwise operations.
- struct L4::Types::Bool< V >

  Boolean meta type.
- struct L4::Types::False

  False meta value.
- struct L4::Types::True

  True meta value.
- struct L4::Types::Same< A, B >

  Compare two data types for equality.

Namespaces

- L4

  L4 low-level kernel interface.
- L4::Types

  L4 basic type helpers for C++.

Macros

- #define L4_TYPES_FLAGS_OPS_DEF(T)

  Helper macro to define a set of bitwise operators on an enum type.

16.305.1 Macro Definition Documentation

16.305.1.1 L4_TYPES_FLAGS_OPS_DEF

```
#define L4_TYPES_FLAGS_OPS_DEF(
                T )
```

Value:

```
    friend constexpr T operator ~ (T f)                                 \
    {                                                                   \
      return T(~((typename L4::Types::Int_for_type<T>::type)f));        \
    }                                                                   \
                                                                        \
    friend constexpr T operator | (T l, T r)                            \
    {                                                                   \
      return T(((typename L4::Types::Int_for_type<T>::type)l)           \
               | ((typename L4::Types::Int_for_type<T>::type)r));       \
    }                                                                   \
                                                                        \
    friend constexpr T operator & (T l, T r)                            \
    {                                                                   \
      return T(((typename L4::Types::Int_for_type<T>::type)l)           \
               & ((typename L4::Types::Int_for_type<T>::type)r));       \
    }
```

Helper macro to define a set of bitwise operators on an enum type.

This allows to use the enum type as bitmask type with '&', '|', and '~' operators that keep the enum type as result.

Definition at line 207 of file types.

16.306 types

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction.  Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License.  This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018
00020
00021 #pragma once
00022
00023 // very simple type traits for basic L4 functions, for a more complete set
00024 // use <l4/cxx/type_traits> or the standard <type_traits>.
00025
00026 namespace L4 {
00027
00031 namespace Types {
00032
```

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
class Flags
{
public:
  typedef UNDERLYING value_type;
  typedef BITS_ENUM bits_enum_type;
  typedef Flags<BITS_ENUM, UNDERLYING> type;

private:
  struct Private_bool;
  value_type _v;
  explicit Flags(value_type v) : _v(v) {}

public:
  enum None_type { None  };

  Flags(None_type) : _v(0) {}

  Flags() : _v(0) {}

  Flags(BITS_ENUM e) : _v(((value_type)1) « e) {}

  static type from_raw(value_type v) { return type(v); }

  operator Private_bool * () const
  { return _v != 0 ? (Private_bool *)1 : 0; }

  bool operator ! () const { return _v == 0; }

  friend type operator | (type lhs, type rhs)
  { return type(lhs._v | rhs._v); }

  friend type operator | (type lhs, bits_enum_type rhs)
  { return lhs | type(rhs); }

  friend type operator & (type lhs, type rhs)
  { return type(lhs._v & rhs._v); }

  friend type operator & (type lhs, bits_enum_type rhs)
  { return lhs & type(rhs); }

  type &operator |= (type rhs) { _v |= rhs._v; return *this; }
  type &operator |= (bits_enum_type rhs) { return operator |= (type(rhs)); }

  type &operator &= (type rhs) { _v &= rhs._v; return *this; }
  type &operator &= (bits_enum_type rhs) { return operator &= (type(rhs)); }

  type operator ~ () const { return type(~_v); }

  type &clear(bits_enum_type flag) { return operator &= (~type(flag)); }

  value_type as_value() const { return _v; }
};

template<unsigned SIZE, bool = true> struct Int_for_size;

template<> struct Int_for_size<sizeof(unsigned char), true>
{ typedef unsigned char type; };

template<> struct Int_for_size<sizeof(unsigned short),
                               (sizeof(unsigned short) > sizeof(unsigned char))>
{ typedef unsigned short type; };

template<> struct Int_for_size<sizeof(unsigned),
                               (sizeof(unsigned) > sizeof(unsigned short))>
{ typedef unsigned type; };

template<> struct Int_for_size<sizeof(unsigned long),
                               (sizeof(unsigned long) > sizeof(unsigned))>
{ typedef unsigned long type; };

template<> struct Int_for_size<sizeof(unsigned long long),
                               (sizeof(unsigned long long) > sizeof(unsigned long))>
{ typedef unsigned long long type; };

template<typename T> struct Int_for_type
{
  typedef typename Int_for_size<sizeof(T)>::type type;
};

#define L4_TYPES_FLAGS_OPS_DEF(T)                                   \
  friend constexpr T operator ~ (T f)                               \
  {                                                                 \
    return T(~((typename L4::Types::Int_for_type<T>::type)f));      \
  }                                                                 \
                                                                    \
  friend constexpr T operator | (T l, T r)                          \
```

```
    {                                                                 \
      return T(((typename L4::Types::Int_for_type<T>::type)l)      \
               | ((typename L4::Types::Int_for_type<T>::type)r)); \
    }                                                                 \
                                                                      \
    friend constexpr T operator & (T l, T r)                          \
    {                                                                 \
      return T(((typename L4::Types::Int_for_type<T>::type)l)      \
               & ((typename L4::Types::Int_for_type<T>::type)r)); \
    }

  template<typename DT>
  struct Flags_ops_t
  {
    friend constexpr DT operator | (DT l, DT r)
    { return DT(l.raw | r.raw); }

    friend constexpr DT operator & (DT l, DT r)
    { return DT(l.raw & r.raw); }

    friend constexpr bool operator == (DT l, DT r)
    { return l.raw == r.raw; }

    friend constexpr bool operator != (DT l, DT r)
    { return l.raw != r.raw; }

    DT operator |= (DT r)
    {
      static_cast<DT *>(this)->raw |= r.raw;
      return *static_cast<DT *>(this);
    }

    DT operator &= (DT r)
    {
      static_cast<DT *>(this)->raw &= r.raw;
      return *static_cast<DT *>(this);
    }

    explicit constexpr operator bool () const
    {
      return static_cast<DT const *>(this)->raw != 0;
    }

    constexpr DT operator ~ () const
    { return DT(~static_cast<DT const *>(this)->raw); }
  };

  template<typename DT, typename T>
  struct Flags_t : Flags_ops_t<Flags_t<DT, T»
  {
    T raw;
    Flags_t() = default;
    explicit constexpr Flags_t(T f) : raw(f) {}
  };


  template< bool V > struct Bool
  {
    typedef Bool<V> type;
    enum { value = V };
  };

  struct False : Bool<false> {};

  struct True : Bool<true> {};

  /*********************/
  template<typename A, typename B>
  struct Same : False {};

  template<typename A>
  struct Same<A, A> : True {};

  template<bool EXP, typename T = void> struct Enable_if {};
  template<typename T> struct Enable_if<true, T> { typedef T type; };

  template<typename T1, typename T2, typename T = void>
  struct Enable_if_same : Enable_if<Same<T1, T2>::value, T> {};

  template<typename T> struct Remove_const { typedef T type; };
  template<typename T> struct Remove_const<T const> { typedef T type; };
  template<typename T> struct Remove_volatile { typedef T type; };
  template<typename T> struct Remove_volatile<T volatile> { typedef T type; };
  template<typename T> struct Remove_cv
  { typedef typename Remove_const<typename Remove_volatile<T>::type>::type type; };

  template<typename T> struct Remove_pointer { typedef T type; };
```

```
  template<typename T> struct Remove_pointer<T*> { typedef T type; };
  template<typename T> struct Remove_reference { typedef T type; };
  template<typename T> struct Remove_reference<T&> { typedef T type; };
  template<typename T> struct Remove_pr { typedef T type; };
  template<typename T> struct Remove_pr<T&> { typedef T type; };
  template<typename T> struct Remove_pr<T*> { typedef T type; };
} // Types
} // L4
```

16.307 l4/sys/debugger File Reference

The debugger interface specifies common debugging related definitions.

```
#include <l4/sys/debugger.h>
#include <l4/sys/kobject>
```

Include dependency graph for debugger:

Data Structures

- class L4::Debugger

  C++ kernel debugger API.

Namespaces

- L4

  L4 low-level kernel interface.

16.307.1 Detailed Description

The debugger interface specifies common debugging related definitions.

Definition in file debugger.

16.308 debugger

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2010-2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/debugger.h>
00027 #include <l4/sys/kobject>
00028
00029 namespace L4 {
00030
00053 class Debugger : public Kobject_t<Debugger, Kobject, L4_PROTO_DEBUGGER>
00054 {
00055 public:
00056   enum
00057   {
00058     Switch_log_on  = L4_DEBUGGER_SWITCH_LOG_ON,
00059     Switch_log_off = L4_DEBUGGER_SWITCH_LOG_OFF,
00060   };
00061
00070   l4_msgtag_t set_object_name(const char *name,
00071                               l4_utcb_t *utcb = l4_utcb()) noexcept
00072   { return l4_debugger_set_object_name_u(cap(), name, utcb); }
00073
00082   unsigned long global_id(l4_utcb_t *utcb = l4_utcb()) noexcept
00083   { return l4_debugger_global_id_u(cap(), utcb); }
00084
00094   unsigned long kobj_to_id(l4_addr_t kobjp,
00095                            l4_utcb_t *utcb = l4_utcb()) noexcept
00096   { return l4_debugger_kobj_to_id_u(cap(), kobjp, utcb); }
00097
00108   long query_log_typeid(const char *name, unsigned idx,
00109                         l4_utcb_t *utcb = l4_utcb()) noexcept
00110   { return l4_debugger_query_log_typeid_u(cap(), name, idx, utcb); }
00111
00127   long query_log_name(unsigned idx,
00128                       char *name, unsigned namelen,
00129                       char *shortname, unsigned shortnamelen,
00130                       l4_utcb_t *utcb = l4_utcb()) noexcept
00131   {
00132     return l4_debugger_query_log_name_u(cap(), idx, name, namelen,
00133                                         shortname, shortnamelen, utcb);
00134   }
00135
00144   l4_msgtag_t switch_log(const char *name, unsigned on_off,
00145                          l4_utcb_t *utcb = l4_utcb()) noexcept
00146   { return l4_debugger_switch_log_u(cap(), name, on_off, utcb); }
00147
00159   l4_msgtag_t get_object_name(unsigned id, char *name, unsigned size,
00160                               l4_utcb_t *utcb = l4_utcb()) noexcept
00161   { return l4_debugger_get_object_name_u(cap(), id, name, size, utcb); }
00162 };
00163 }
```

16.309 l4/sys/debugger.h File Reference

Debugger related definitions.

```
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
#include <l4/sys/kernel_object.h>
```

Include dependency graph for debugger.h:

This graph shows which files directly or indirectly include this file:

Functions

- l4_msgtag_t l4_debugger_set_object_name (l4_cap_idx_t cap, const char *name) L4_NOTHROW

Set the name of a kernel object.

- l4_msgtag_t l4_debugger_get_object_name (l4_cap_idx_t cap, unsigned id, char *name, unsigned size) L4_NOTHROW

  Get name of the kernel object with Id `id`.

- unsigned long l4_debugger_global_id (l4_cap_idx_t cap) L4_NOTHROW

  Get the globally unique ID of the object behind a capability.

- unsigned long l4_debugger_kobj_to_id (l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW

  Get the globally unique ID of the object behind the kobject pointer.

- long l4_debugger_query_log_typeid (l4_cap_idx_t cap, const char *name, unsigned idx) L4_NOTHROW

  Query the log-id for a log type.

- long l4_debugger_query_log_name (l4_cap_idx_t cap, unsigned idx, char *name, unsigned namelen, char *shortname, unsigned shortnamelen) L4_NOTHROW

  Query the name of a log type given the ID.

- l4_msgtag_t l4_debugger_switch_log (l4_cap_idx_t cap, const char *name, int on_off) L4_NOTHROW

  Set or unset log.

- unsigned __strcpy_maxlen (char *dst, char const *src, unsigned maxlen)

  *Copy a number of characters from the C string **src** to the C string **dst**.*

16.309.1 Detailed Description

Debugger related definitions.

Definition in file debugger.h.

16.309.2 Function Documentation

16.309.2.1 __strcpy_maxlen()

```
unsigned __strcpy_maxlen (
            char * dst,
            char const * src,
            unsigned maxlen )  [inline]
```

Copy a number of characters from the C string **src** to the C string **dst**.

The resulting string **dst** is always '\0'-terminated unless **maxlen** is 0. If the C string in **src** is shorter than the buffer **dst** then the remaining bytes in **dst** are NOT initialized.

Parameters

dst	Target buffer.
src	Source buffer.
maxlen	Maximum number of bytes written to the target buffer.

Returns

The number of bytes written (including the terminating '\0'.

Definition at line 231 of file debugger.h.

16.309.2.2 l4_debugger_get_object_name()

```
l4_msgtag_t l4_debugger_get_object_name (
            l4_cap_idx_t cap,
            unsigned id,
            char * name,
            unsigned size )  [inline]
```

Get name of the kernel object with Id `id`.

Parameters

	cap	Capability of the debugger object.
	id	Global id of the object whose name is asked.
`out`	*name*	Buffer to copy the name into. The buffer must be allocated by the caller.
	size	Length of the `name` buffer.

Returns

Syscall return tag

Definition at line 381 of file debugger.h.

16.309.2.3 l4_debugger_query_log_name()

```
long l4_debugger_query_log_name (
            l4_cap_idx_t cap,
            unsigned idx,
            char * name,
            unsigned namelen,
            char * shortname,
            unsigned shortnamelen )  [inline]
```

Query the name of a log type given the ID.

Parameters

cap	Debugger capability.
idx	ID to query.
name	Buffer to copy name to.
namelen	Buffer length of name.
shortname	Buffer to copy `shortname` to.
shortnamelen	Buffer length of `shortname`.

Return values

0	Success
<0	Error

This is a debugging facility, the call might be invalid.

Definition at line 365 of file debugger.h.

16.309.2.4 l4_debugger_query_log_typeid()

```
long l4_debugger_query_log_typeid (
            l4_cap_idx_t cap,
            const char * name,
            unsigned idx )  [inline]
```

Query the log-id for a log type.

Parameters

cap	Debugger capability
name	Name to query for.
idx	Idx to start searching, start with 0

Returns

positive ID, or negative error code

This is a debugging facility, the call might be invalid.

Definition at line 358 of file debugger.h.

16.309.2.5 l4_debugger_switch_log()

```
l4_msgtag_t l4_debugger_switch_log (
            l4_cap_idx_t cap,
            const char * name,
            int on_off )  [inline]
```

Set or unset log.

Parameters

cap	Debugger object.
name	Name of the log type.
on_off	1: turn log on, 0: turn log off

Returns

Syscall return tag

Definition at line 374 of file debugger.h.

16.310 debugger.h

```
#pragma once
/*
 * (c) 2008-2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */

#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

L4_INLINE l4_msgtag_t
l4_debugger_set_object_name(l4_cap_idx_t cap, const char *name) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_debugger_set_object_name_u(l4_cap_idx_t cap, const char *name, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
                            char *name, unsigned size) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
                              char *name, unsigned size,
                              l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE unsigned long
l4_debugger_global_id(l4_cap_idx_t cap) L4_NOTHROW;

L4_INLINE unsigned long
l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE unsigned long
l4_debugger_kobj_to_id(l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW;

L4_INLINE unsigned long
l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE long
l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
                             unsigned idx) L4_NOTHROW;

L4_INLINE long
l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
                               unsigned idx, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE long
l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
                           char *name, unsigned namelen,
                           char *shortname, unsigned shortnamelen) L4_NOTHROW;

L4_INLINE long
l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
                             char *name, unsigned namelen,
                             char *shortname, unsigned shortnamelen,
                             l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
                       int on_off) L4_NOTHROW;
```

```
L4_INLINE l4_msgtag_t
l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
                         l4_utcb_t *utcb) L4_NOTHROW;

enum
{
  L4_DEBUGGER_NAME_SET_OP         = 0UL,
  L4_DEBUGGER_GLOBAL_ID_OP        = 1UL,
  L4_DEBUGGER_KOBJ_TO_ID_OP       = 2UL,
  L4_DEBUGGER_QUERY_LOG_TYPEID_OP = 3UL,
  L4_DEBUGGER_SWITCH_LOG_OP       = 4UL,
  L4_DEBUGGER_NAME_GET_OP         = 5UL,
  L4_DEBUGGER_QUERY_LOG_NAME_OP   = 6UL,
};

enum
{
  L4_DEBUGGER_SWITCH_LOG_ON  = 1,
  L4_DEBUGGER_SWITCH_LOG_OFF = 0,
};

/* IMPLEMENTATION ----------------------------------------------------------*/

#include <l4/sys/kernel_object.h>

L4_INLINE unsigned
__strcpy_maxlen(char *dst, char const *src, unsigned maxlen)
{
  unsigned i;
  if (!maxlen)
    return 0;

  for (i = 0; i < maxlen - 1 && src[i]; ++i)
    dst[i] = src[i];
  dst[i] = '\0';

  return i + 1;
}

L4_INLINE l4_msgtag_t
l4_debugger_set_object_name_u(l4_cap_idx_t cap,
                              const char *name, l4_utcb_t *utcb) L4_NOTHROW
{
  unsigned i;
  l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_SET_OP;
  i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[1], name,
                      (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t));
  i = l4_bytes_to_mwords(i);
  return l4_invoke_debugger(cap, l4_msgtag(0, 1 + i, 0, 0), utcb);
}

L4_INLINE unsigned long
l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_GLOBAL_ID_OP;
  if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 1, 0, 0), utcb), utcb))
    return ~0UL;
  return l4_utcb_mr_u(utcb)->mr[0];
}

L4_INLINE unsigned long
l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_KOBJ_TO_ID_OP;
  l4_utcb_mr_u(utcb)->mr[1] = kobjp;
  if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb))
    return ~0UL;
  return l4_utcb_mr_u(utcb)->mr[0];
}

L4_INLINE long
l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
                               unsigned idx,
                               l4_utcb_t *utcb) L4_NOTHROW
{
  unsigned i;
  long e;
  l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_TYPEID_OP;
  l4_utcb_mr_u(utcb)->mr[1] = idx;
  i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 32);
  i = l4_bytes_to_mwords(i);
  e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb), utcb);
  if (e < 0)
    return e;
  return l4_utcb_mr_u(utcb)->mr[0];
}
```

```
L4_INLINE long
l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
                             char *name, unsigned namelen,
                             char *shortname, unsigned shortnamelen,
                             l4_utcb_t *utcb) L4_NOTHROW
{
  long e;
  char const *n;
  l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_NAME_OP;
  l4_utcb_mr_u(utcb)->mr[1] = idx;
  e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb);
  if (e < 0)
    return e;
  n = (char const *)&l4_utcb_mr_u(utcb)->mr[0];
  __strcpy_maxlen(name, n, namelen);
  __strcpy_maxlen(shortname, n + __builtin_strlen(n) + 1, shortnamelen);
  return 0;
}


L4_INLINE l4_msgtag_t
l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
                         l4_utcb_t *utcb) L4_NOTHROW
{
  unsigned i;
  l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_SWITCH_LOG_OP;
  l4_utcb_mr_u(utcb)->mr[1] = on_off;
  i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 32);
  i = l4_bytes_to_mwords(i);
  return l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb);
}

L4_INLINE l4_msgtag_t
l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
                              char *name, unsigned size,
                              l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msgtag_t t;
  l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_GET_OP;
  l4_utcb_mr_u(utcb)->mr[1] = id;
  t = l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb);
  __strcpy_maxlen(name, (char const *)&l4_utcb_mr_u(utcb)->mr[0], size);
  return t;
}


L4_INLINE l4_msgtag_t
l4_debugger_set_object_name(l4_cap_idx_t cap,
                            const char *name) L4_NOTHROW
{
  return l4_debugger_set_object_name_u(cap, name, l4_utcb());
}

L4_INLINE unsigned long
l4_debugger_global_id(l4_cap_idx_t cap) L4_NOTHROW
{
  return l4_debugger_global_id_u(cap, l4_utcb());
}

L4_INLINE unsigned long
l4_debugger_kobj_to_id(l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW
{
  return l4_debugger_kobj_to_id_u(cap, kobjp, l4_utcb());
}

L4_INLINE long
l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
                             unsigned idx) L4_NOTHROW
{
  return l4_debugger_query_log_typeid_u(cap, name, idx, l4_utcb());
}

L4_INLINE long
l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
                           char *name, unsigned namelen,
                           char *shortname, unsigned shortnamelen) L4_NOTHROW
{
  return l4_debugger_query_log_name_u(cap, idx, name, namelen,
                                      shortname, shortnamelen, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
                       int on_off) L4_NOTHROW
{
  return l4_debugger_switch_log_u(cap, name, on_off, l4_utcb());
```

```
}

L4_INLINE l4_msgtag_t
l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
                            char *name, unsigned size) L4_NOTHROW
{
  return l4_debugger_get_object_name_u(cap, id, name, size, l4_utcb());
}
```

16.311 l4/sys/err.h File Reference

Error codes.

`#include <l4/sys/compiler.h>`

Include dependency graph for err.h:

This graph shows which files directly or indirectly include this file:

Enumerations

- enum l4_error_code_t {
  L4_EOK = 0 , L4_EPERM = 1 , L4_ENOENT = 2 , L4_EIO = 5 ,
  L4_ENXIO = 6 , L4_E2BIG = 7 , L4_EAGAIN = 11 , L4_ENOMEM = 12 ,
  L4_EACCESS = 13 , L4_EFAULT = 14 , L4_EBUSY = 16 , L4_EEXIST = 17 ,
  L4_ENODEV = 19 , L4_EINVAL = 22 , L4_ENOSPC = 28 , L4_ERANGE = 34 ,
  L4_ENAMETOOLONG = 36 , L4_ENOSYS = 38 , L4_EBADPROTO = 39 , L4_EADDRNOTAVAIL = 99 ,
  L4_ERRNOMAX = 100 , L4_ENOREPLY = 1000 , L4_EMSGTOOSHORT = 1001 , L4_EMSGTOOLONG = 1002 ,
  L4_EMSGMISSARG = 1003 , L4_EIPC_LO = 2000 , L4_EIPC_HI = 2000 + 0x1f }

  L4 error codes.

16.311.1 Detailed Description

Error codes.

Definition in file err.h.

16.312 err.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/sys/compiler.h>
00026
00041 enum l4_error_code_t
00042 {
00043   L4_EOK           =  0,
00044   L4_EPERM         =  1,
00045   L4_ENOENT        =  2,
00046   L4_EIO           =  5,
00047   L4_ENXIO         =  6,
00048   L4_E2BIG         =  7,
00049   L4_EAGAIN        = 11,
00050   L4_ENOMEM        = 12,
00051   L4_EACCESS       = 13,
00052   L4_EFAULT        = 14,
00053   L4_EBUSY         = 16,
00054   L4_EEXIST        = 17,
00055   L4_ENODEV        = 19,
00056   L4_EINVAL        = 22,
00057   L4_ENOSPC        = 28,
00058   L4_ERANGE        = 34,
00059   L4_ENAMETOOLONG  = 36,
00060   L4_ENOSYS        = 38,
00061   L4_EBADPROTO     = 39,
00062   L4_EADDRNOTAVAIL = 99,
00063   L4_ERRNOMAX      = 100,
00065   L4_ENOREPLY      = 1000,
00066   L4_EMSGTOOSHORT  = 1001,
00067   L4_EMSGTOOLONG   = 1002,
00068   L4_EMSGMISSARG   = 1003,
00070   L4_EIPC_LO       = 2000,
00071   L4_EIPC_HI       = 2000 + 0x1f,
00072 };
00073
00074 __BEGIN_DECLS
00075 L4_CV char const *l4sys_errtostr(long err) L4_NOTHROW;
00076 __END_DECLS
00077
00078
```

16.313 l4/sys/exception File Reference

Exception C++ interface.

```
#include <l4/sys/capability>
#include <l4/sys/types.h>
```

```
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for exception:

Data Structures

- class L4::Exception

  Exception interface.

Namespaces

- L4

  L4 low-level kernel interface.

16.313.1 Detailed Description

Exception C++ interface.

Definition in file exception.

16.314 exception

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/sys/capability>
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/cxx/ipc_types>
00028 #include <l4/sys/cxx/ipc_iface>
00029
00030 namespace L4 {
00031
00042 class L4_EXPORT Exception :
00043   public Kobject_0t<Exception, L4_PROTO_EXCEPTION>
00044 {
00045 public:
00046   // TODO: pass a reference/pointer to the UTCB not copy the regs
00057   L4_INLINE_RPC(
00058       l4_msgtag_t, exception, (L4::Ipc::In_out<l4_exc_regs_t *> regs,
00059                                L4::Ipc::Rcv_fpage rwin,
00060                                L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00061
00062   typedef L4::Typeid::Rpc_nocode<exception_t> Rpcs;
00063 };
00064
00065 }
```

16.315 l4/sys/factory File Reference

Common factory related definitions.

```
#include <l4/sys/factory.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_varg>
```

Include dependency graph for factory:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Factory

  C++ Factory interface, see Factory for the C interface.
- struct L4::Factory::Nil

  Special type to add a void argument into the factory create stream.
- struct L4::Factory::Lstr

  Special type to add a pascal string into the factory create stream.
- class L4::Factory::S

  Stream class for the create() argument stream.

Namespaces

- L4

  L4 low-level kernel interface.

16.315.1 Detailed Description

Common factory related definitions.

Definition in file factory.

16.316 factory

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/factory.h>
00028 #include <l4/sys/capability>
00029 #include <l4/sys/cxx/ipc_iface>
00030 #include <l4/sys/cxx/ipc_varg>
00031
00032 namespace L4 {
00033
00048 class Factory : public Kobject_t<Factory, Kobject, L4_PROTO_FACTORY>
00049 {
00050 public:
00051
00052   typedef l4_mword_t Proto;
00053
00057   struct Nil {};
00058
00064   struct Lstr
00065   {
00069     char const *s;
00070
00074     unsigned len;
00075
00080     Lstr(char const *s, unsigned len) noexcept : s(s), len(len) {}
00081   };
00082
00089   class S
00090   {
00091   private:
00092     l4_utcb_t *u;
00093     l4_msgtag_t t;
00094     l4_cap_idx_t f;
00095
00096     template<typename T>
00097     static T &&_move(T &c) { return static_cast<T &&>(c); }
00098
00099   public:
00100     S(S const &) = delete;
00101     S &operator = (S const &) & = delete;
00102
00108     S(S &&o) noexcept
00109     : u(o.u), t(o.t), f(o.f)
00110     { o.t.raw = 0; }
00111
00112     S &operator = (S &&o) & noexcept
00113     {
00114       u = o.u;
00115       t = o.t;
00116       f = o.f;
00117       o.t.raw = 0;
00118       return *this;
00119     }
```

```
00120
00132     S(l4_cap_idx_t f, long obj, L4::Cap<void> target,
00133       l4_utcb_t *utcb) noexcept
00134     : u(utcb), t(l4_factory_create_start_u(obj, target.cap(), u)), f(f)
00135     {}
00136
00141     ~S() noexcept
00142     {
00143       if (t.raw)
00144   l4_factory_create_commit_u(f, t, u);
00145     }
00146
00152     operator l4_msgtag_t () noexcept
00153     {
00154       l4_msgtag_t r = l4_factory_create_commit_u(f, t, u);
00155       t.raw = 0;
00156       return r;
00157     }
00158
00166     void put(l4_mword_t i) noexcept
00167     {
00168       l4_factory_create_add_int_u(i, &t, u);
00169     }
00170
00178     void put(l4_umword_t i) noexcept
00179     {
00180       l4_factory_create_add_uint_u(i, &t, u);
00181     }
00182
00192     void put(char const *s) & noexcept
00193     {
00194       l4_factory_create_add_str_u(s, &t, u);
00195     }
00196
00208     void put(Lstr const &s) & noexcept
00209     {
00210       l4_factory_create_add_lstr_u(s.s, s.len, &t, u);
00211     }
00212
00218     void put(Nil) & noexcept
00219     {
00220       l4_factory_create_add_nil_u(&t, u);
00221     }
00222
00230     void put(l4_fpage_t d) & noexcept
00231     {
00232       l4_factory_create_add_fpage_u(d, &t, u);
00233     }
00234
00235     template<typename T>
00236     S &operator « (T const &d) & noexcept
00237     {
00238       put(d);
00239       return *this;
00240     }
00241
00242     template<typename T>
00243     S &&operator « (T const &d) && noexcept
00244     {
00245       put(d);
00246       return _move(*this);
00247     }
00248   };
00249
00250
00251 public:
00252
00277   S create(Cap<void> target, long obj, l4_utcb_t *utcb = l4_utcb()) noexcept
00278   {
00279     return S(cap(), obj, target, utcb);
00280   }
00281
00307   template<typename OBJ>
00308   S create(Cap<OBJ> target, l4_utcb_t *utcb = l4_utcb()) noexcept
00309   {
00310     return S(cap(), OBJ::Protocol, target, utcb);
00311   }
00312
00313   L4_INLINE_RPC_NF(
00314       l4_msgtag_t, create, (L4::Ipc::Out<L4::Cap<void> > target, l4_mword_t obj,
00315                             L4::Ipc::Varg const *args),
00316       L4::Ipc::Call_t<L4_CAP_FPAGE_S>);
00317
00341   l4_msgtag_t create_task(Cap<Task> const & target_cap,
00342                           l4_fpage_t const &utcb_area,
00343                           l4_utcb_t *utcb = l4_utcb()) noexcept
00344   { return l4_factory_create_task_u(cap(), target_cap.cap(), utcb_area, utcb); }
```

```
00345
00368   l4_msgtag_t create_factory(Cap<Factory> const &target_cap,
00369                              unsigned long limit,
00370                              l4_utcb_t *utcb = l4_utcb()) noexcept
00371   { return l4_factory_create_factory_u(cap(), target_cap.cap(), limit, utcb); }
00372
00402   l4_msgtag_t create_gate(Cap<void> const &target_cap,
00403                           Cap<Thread> const &thread_cap, l4_umword_t label,
00404                           l4_utcb_t *utcb = l4_utcb()) noexcept
00405   { return l4_factory_create_gate_u(cap(), target_cap.cap(), thread_cap.cap(), label, utcb); }
00406
00407   typedef L4::Typeid::Rpc_nocode<create_t> Rpcs;
00408 };
00409
00410 }
```

16.317 l4/sys/factory.h File Reference

Common factory related definitions.

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for factory.h:

This graph shows which files directly or indirectly include this file:

Functions

- l4_msgtag_t l4_factory_create_task (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_fpage_t utcb_area) L4_NOTHROW

  Create a new task.
- l4_msgtag_t l4_factory_create_thread (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW

  Create a new thread.
- l4_msgtag_t l4_factory_create_factory (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW

  Create a new factory.
- l4_msgtag_t l4_factory_create_gate (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW

  Create a new IPC gate.
- l4_msgtag_t l4_factory_create_irq (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW

  Create a new IRQ sender.
- l4_msgtag_t l4_factory_create_vm (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW

  Create a new virtual machine.
- l4_msgtag_t l4_factory_create (l4_cap_idx_t factory, long obj, l4_cap_idx_t target) L4_NOTHROW

  Create a new kernel object.

16.317.1 Detailed Description

Common factory related definitions.

Definition in file factory.h.

16.318 factory.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>,
00011  *               Henning Schild <hschild@os.inf.tu-dresden.de>
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU General Public License 2.
00016  * Please see the COPYING-GPL-2 file for details.
00017  *
00018  * As a special exception, you may use this file as part of a free software
00019  * library without restriction.  Specifically, if other files instantiate
00020  * templates or use macros or inline functions from this file, or you compile
00021  * this file and link it with other files to produce an executable, this
00022  * file does not by itself cause the resulting executable to be covered by
00023  * the GNU General Public License.  This exception does not however
00024  * invalidate any other reasons why the executable file might be covered by
00025  * the GNU General Public License.
00026  */
00027 #pragma once
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/utcb.h>
00032
00087 L4_INLINE l4_msgtag_t
00088 l4_factory_create_task(l4_cap_idx_t factory,
00089                        l4_cap_idx_t target_cap, l4_fpage_t utcb_area) L4_NOTHROW;
00090
00095 L4_INLINE l4_msgtag_t
00096 l4_factory_create_task_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00097                          l4_fpage_t utcb_area, l4_utcb_t *utcb) L4_NOTHROW;
00098
00111 L4_INLINE l4_msgtag_t
```

```
l4_factory_create_thread(l4_cap_idx_t factory,
                         l4_cap_idx_t target_cap) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_thread_u(l4_cap_idx_t factory,
                           l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_factory(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
                          unsigned long limit) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_factory_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
                            unsigned long limit, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_gate(l4_cap_idx_t factory,
                       l4_cap_idx_t target_cap,
                       l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_gate_u(l4_cap_idx_t factory,
                         l4_cap_idx_t target_cap,
                         l4_cap_idx_t thread_cap, l4_umword_t label,
                         l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_irq(l4_cap_idx_t factory,
                      l4_cap_idx_t target_cap) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_irq_u(l4_cap_idx_t factory,
                        l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_vm(l4_cap_idx_t factory,
                     l4_cap_idx_t target_cap) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_vm_u(l4_cap_idx_t factory,
                       l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_start_u(long obj, l4_cap_idx_t target,
                          l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE int
l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
                              l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE int
l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
                            l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE int
l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
                             l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE int
l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
                            l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE int
l4_factory_create_add_lstr_u(char const *s, unsigned len, l4_msgtag_t *tag,
                             l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE int
l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
                           l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
                    l4_utcb_t *utcb) L4_NOTHROW;


L4_INLINE l4_msgtag_t
l4_factory_create(l4_cap_idx_t factory, long obj,
                  l4_cap_idx_t target) L4_NOTHROW;

/* IMPLEMENTATION ---------------------------------------------------------*/

#include <l4/sys/ipc.h>

L4_INLINE l4_msgtag_t
```

```
l4_factory_create_task_u(l4_cap_idx_t factory,
                         l4_cap_idx_t target_cap, l4_fpage_t utcb_area,
                         l4_utcb_t *u) L4_NOTHROW
{
  l4_msgtag_t t;
  t = l4_factory_create_start_u(L4_PROTO_TASK, target_cap, u);
  l4_factory_create_add_fpage_u(utcb_area, &t, u);
  return l4_factory_create_commit_u(factory, t, u);
}

L4_INLINE l4_msgtag_t
l4_factory_create_thread_u(l4_cap_idx_t factory,
                           l4_cap_idx_t target_cap, l4_utcb_t *u) L4_NOTHROW
{
  return l4_factory_create_u(factory, L4_PROTO_THREAD, target_cap, u);
}

L4_INLINE l4_msgtag_t
l4_factory_create_factory_u(l4_cap_idx_t factory,
                            l4_cap_idx_t target_cap, unsigned long limit,
                            l4_utcb_t *u) L4_NOTHROW
{
  l4_msgtag_t t;
  t = l4_factory_create_start_u(L4_PROTO_FACTORY, target_cap, u);
  l4_factory_create_add_uint_u(limit, &t, u);
  return l4_factory_create_commit_u(factory, t, u);
}

L4_INLINE l4_msgtag_t
l4_factory_create_gate_u(l4_cap_idx_t factory,
                         l4_cap_idx_t target_cap,
                         l4_cap_idx_t thread_cap, l4_umword_t label,
                         l4_utcb_t *u) L4_NOTHROW
{
  l4_msgtag_t t;
  l4_msg_regs_t *v;
  int items = 0;
  t = l4_factory_create_start_u(0, target_cap, u);
  l4_factory_create_add_uint_u(label, &t, u);
  v = l4_utcb_mr_u(u);
  if (!(thread_cap & L4_INVALID_CAP_BIT))
    {
      items = 1;
      v->mr[3] = l4_map_obj_control(0,0);
      v->mr[4] = l4_obj_fpage(thread_cap, 0, L4_CAP_FPAGE_RWS).raw;
    }
  t = l4_msgtag(l4_msgtag_label(t), l4_msgtag_words(t), items, l4_msgtag_flags(t));
  return l4_factory_create_commit_u(factory, t, u);
}

L4_INLINE l4_msgtag_t
l4_factory_create_irq_u(l4_cap_idx_t factory,
                        l4_cap_idx_t target_cap, l4_utcb_t *u) L4_NOTHROW
{
  return l4_factory_create_u(factory, L4_PROTO_IRQ_SENDER, target_cap, u);
}

L4_INLINE l4_msgtag_t
l4_factory_create_vm_u(l4_cap_idx_t factory,
                       l4_cap_idx_t target_cap,
                       l4_utcb_t *u) L4_NOTHROW
{
  return l4_factory_create_u(factory, L4_PROTO_VM, target_cap, u);
}



L4_INLINE l4_msgtag_t
l4_factory_create_task(l4_cap_idx_t factory,
                       l4_cap_idx_t target_cap, l4_fpage_t utcb_area) L4_NOTHROW
{
  return l4_factory_create_task_u(factory, target_cap, utcb_area, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_factory_create_thread(l4_cap_idx_t factory,
                         l4_cap_idx_t target_cap) L4_NOTHROW
{
  return l4_factory_create_thread_u(factory, target_cap, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_factory_create_factory(l4_cap_idx_t factory,
                          l4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW

```

```
{
  return l4_factory_create_factory_u(factory, target_cap, limit, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_factory_create_gate(l4_cap_idx_t factory,
                       l4_cap_idx_t target_cap,
                       l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW
{
  return l4_factory_create_gate_u(factory, target_cap, thread_cap, label, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_factory_create_irq(l4_cap_idx_t factory,
                      l4_cap_idx_t target_cap) L4_NOTHROW
{
  return l4_factory_create_irq_u(factory, target_cap, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_factory_create_vm(l4_cap_idx_t factory,
                     l4_cap_idx_t target_cap) L4_NOTHROW
{
  return l4_factory_create_vm_u(factory, target_cap, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_factory_create_start_u(long obj, l4_cap_idx_t target_cap,
                          l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(u);
  l4_buf_regs_t *b = l4_utcb_br_u(u);
  v->mr[0] = obj;
  b->bdr = 0;
  b->br[0] = target_cap | L4_RCV_ITEM_SINGLE_CAP;
  return l4_msgtag(L4_PROTO_FACTORY, 1, 0, 0);
}

L4_INLINE int
l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
                              l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(u);
  int w = l4_msgtag_words(*tag);
  if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
    return 0;
  v->mr[w] = L4_VARG_TYPE_FPAGE | (sizeof(l4_fpage_t) « 16);
  v->mr[w + 1] = d.raw;
  w += 2;
  tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
  return 1;
}

L4_INLINE int
l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
                            l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(u);
  int w = l4_msgtag_words(*tag);
  if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
    return 0;
  v->mr[w] = L4_VARG_TYPE_MWORD | (sizeof(l4_mword_t) « 16);
  v->mr[w + 1] = d;
  w += 2;
  tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
  return 1;
}

L4_INLINE int
l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
                             l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(u);
  int w = l4_msgtag_words(*tag);
  if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
    return 0;
  v->mr[w] = L4_VARG_TYPE_UMWORD | (sizeof(l4_umword_t) « 16);
  v->mr[w + 1] = d;
  w += 2;
  tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
  return 1;
}

L4_INLINE int
l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
                            l4_utcb_t *u) L4_NOTHROW
{
```

```
  return l4_factory_create_add_lstr_u(s, __builtin_strlen(s) + 1, tag, u);
}

L4_INLINE int
l4_factory_create_add_lstr_u(char const *s, unsigned len, l4_msgtag_t *tag,
                             l4_utcb_t *u) L4_NOTHROW
{

  l4_msg_regs_t *v = l4_utcb_mr_u(u);
  unsigned w = l4_msgtag_words(*tag);
  char *c;
  unsigned i;

  if (w + 1 + l4_bytes_to_mwords(len) > L4_UTCB_GENERIC_DATA_SIZE)
    return 0;

  v->mr[w] = L4_VARG_TYPE_STRING | (len « 16);
  c = (char*)&v->mr[w + 1];
  for (i = 0; i < len; ++i)
    *c++ = *s++;

  w = w + 1 + l4_bytes_to_mwords(len);

  tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
  return 1;
}

L4_INLINE int
l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  int w = l4_msgtag_words(*tag);
  v->mr[w] = L4_VARG_TYPE_NIL;
  ++w;
  tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
  return 1;
}


L4_INLINE l4_msgtag_t
l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
                           l4_utcb_t *u) L4_NOTHROW
{
  return l4_ipc_call(factory, u, tag, L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
                    l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msgtag_t t = l4_factory_create_start_u(obj, target, utcb);
  return l4_factory_create_commit_u(factory, t, utcb);
}


L4_INLINE l4_msgtag_t
l4_factory_create(l4_cap_idx_t factory, long obj,
                  l4_cap_idx_t target) L4_NOTHROW
{
  return l4_factory_create_u(factory, obj, target, l4_utcb());
}
```

16.319 l4/sys/icu File Reference

Interrupt controller.

`#include <l4/sys/irq>`

Include dependency graph for icu:

This graph shows which files directly or indirectly include this file:

16.319.1 Detailed Description

Interrupt controller.

Definition in file icu.

16.320 icu

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
```

```
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/sys/irq>
```

16.321 l4/sys/icu.h File Reference

Interrupt controller.

```
#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for icu.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct l4_icu_info_t

  Info structure for an ICU.
- struct l4_icu_msi_info_t

  Info to use for a specific MSI.

Typedefs

- typedef struct l4_icu_info_t l4_icu_info_t

  Info structure for an ICU.
- typedef struct l4_icu_msi_info_t l4_icu_msi_info_t

  Info to use for a specific MSI.

Enumerations

- enum L4_icu_flags { L4_ICU_FLAG_MSI }

  Flags for IRQ numbers used for the ICU.
- enum L4_irq_mode {
  L4_IRQ_F_NONE = 0 , L4_IRQ_F_LEVEL = 0x2 , L4_IRQ_F_EDGE = 0x0 , L4_IRQ_F_POS = 0x0 , L4_IRQ_F_NEG = 0x4 , L4_IRQ_F_BOTH = 0x8 , L4_IRQ_F_LEVEL_HIGH = 0x3 , L4_IRQ_F_LEVEL_LOW = 0x7 ,
  L4_IRQ_F_POS_EDGE = 0x1 , L4_IRQ_F_NEG_EDGE = 0x5 , L4_IRQ_F_BOTH_EDGE = 0x9 , L4_IRQ_F_MASK = 0xf ,
  L4_IRQ_F_SET_WAKEUP = 0x10 , L4_IRQ_F_CLEAR_WAKEUP = 0x20 }

  Interrupt attributes.
- enum L4_icu_opcode {
  L4_ICU_OP_BIND , L4_ICU_OP_UNBIND , L4_ICU_OP_INFO , L4_ICU_OP_MSI_INFO , L4_ICU_OP_UNMASK , L4_ICU_OP_MASK , L4_ICU_OP_SET_MODE }

  Opcodes to the ICU interface.

Functions

- l4_msgtag_t l4_icu_bind (l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW

  Bind an interrupt line of an interrupt controller to an interrupt object.
- l4_msgtag_t l4_icu_bind_u (l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW

  Bind an interrupt line of an interrupt controller to an interrupt object.
- l4_msgtag_t l4_icu_unbind (l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW

  Remove binding of an interrupt line from the interrupt controller object.
- l4_msgtag_t l4_icu_unbind_u (l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW

  Remove binding of an interrupt line from the interrupt controller object.
- l4_msgtag_t l4_icu_set_mode (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW

  Set interrupt mode.
- l4_msgtag_t l4_icu_set_mode_u (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode, l4_utcb_t *utcb) L4_NOTHROW

  Set interrupt mode.
- l4_msgtag_t l4_icu_info (l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW

  Get information about the ICU features.
- l4_msgtag_t l4_icu_info_u (l4_cap_idx_t icu, l4_icu_info_t *info, l4_utcb_t *utcb) L4_NOTHROW

  Get information about the ICU features.
- l4_msgtag_t l4_icu_msi_info (l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source, l4_icu_msi_info_t *msi_info) L4_NOTHROW

  Get MSI info about IRQ.
- l4_msgtag_t l4_icu_msi_info_u (l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source, l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW

  Get MSI info about IRQ.
- l4_msgtag_t l4_icu_unmask (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to) L4_NOTHROW

  Unmask an IRQ line.
- l4_msgtag_t l4_icu_unmask_u (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW

  Unmask the given interrupt line.
- l4_msgtag_t l4_icu_mask (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to) L4_NOTHROW

  Mask an IRQ line.
- l4_msgtag_t l4_icu_mask_u (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW

  Mask an IRQ line.

16.321.1 Detailed Description

Interrupt controller.

Definition in file icu.h.

16.322 icu.h

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>

enum L4_icu_flags
{
  L4_ICU_FLAG_MSI = 0x80000000,
};


enum L4_irq_mode
{
  L4_IRQ_F_NONE         = 0,
  L4_IRQ_F_LEVEL        = 0x2,
  L4_IRQ_F_EDGE         = 0x0,
  L4_IRQ_F_POS          = 0x0,
  L4_IRQ_F_NEG          = 0x4,
  L4_IRQ_F_BOTH         = 0x8,
  L4_IRQ_F_LEVEL_HIGH   = 0x3,
  L4_IRQ_F_LEVEL_LOW    = 0x7,
  L4_IRQ_F_POS_EDGE     = 0x1,
  L4_IRQ_F_NEG_EDGE     = 0x5,
  L4_IRQ_F_BOTH_EDGE    = 0x9,
  L4_IRQ_F_MASK         = 0xf,
  L4_IRQ_F_SET_WAKEUP   = 0x10,
  L4_IRQ_F_CLEAR_WAKEUP = 0x20,
};


enum L4_icu_opcode
{
  L4_ICU_OP_BIND = 0,

  L4_ICU_OP_UNBIND = 1,

  L4_ICU_OP_INFO = 2,

  L4_ICU_OP_MSI_INFO = 3,

  L4_ICU_OP_UNMASK   = 4,

  L4_ICU_OP_MASK     = 5,

  L4_ICU_OP_SET_MODE = 6,
};

enum L4_icu_ctl_op
{
  L4_ICU_CTL_UNMASK = 0,
  L4_ICU_CTL_MASK   = 1
};


typedef struct l4_icu_info_t
{
  unsigned features;

  unsigned nr_irqs;

  unsigned nr_msis;
} l4_icu_info_t;

typedef struct l4_icu_msi_info_t
{
```

```
  l4_uint64_t msi_addr;
  l4_uint32_t msi_data;
} l4_icu_msi_info_t;

L4_INLINE l4_msgtag_t
l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
              l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
                l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode,
                  l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
              l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
                l4_icu_msi_info_t *msi_info) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
                  l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW;


L4_INLINE l4_msgtag_t
l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
              l4_timeout_t to) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
                l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
            l4_timeout_t to) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
              l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
                 l4_umword_t *label,l4_timeout_t to,
                 l4_utcb_t *utcb) L4_NOTHROW;


/*************************************************************************
 * Implementations
 */

L4_INLINE l4_msgtag_t
l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
              l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
  m->mr[0] = L4_ICU_OP_BIND;
  m->mr[1] = irqnum;
  m->mr[2] = l4_map_obj_control(0, 0);
  m->mr[3] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
  return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
                l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
  m->mr[0] = L4_ICU_OP_UNBIND;
  m->mr[1] = irqnum;
  m->mr[2] = l4_map_obj_control(0, 0);
```

```
  m->mr[3] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
  return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
              l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msgtag_t res;
  l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
  m->mr[0] = L4_ICU_OP_INFO;
  res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), L4_IPC_NEVER);
  info->features = m->mr[0];
  info->nr_irqs  = m->mr[1];
  info->nr_msis  = m->mr[2];
  return res;
}

L4_INLINE l4_msgtag_t
l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
                  l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msgtag_t res;
  l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
  m->mr[0] = L4_ICU_OP_MSI_INFO;
  m->mr[1] = irqnum;
  m->mr64[l4_utcb_mr64_idx(2)] = source;
  res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ,
                                         2 + 1 * sizeof(l4_uint64_t)
                                               / sizeof(l4_umword_t),
                                         0, 0), L4_IPC_NEVER);
  if (L4_UNLIKELY(l4_msgtag_has_error(res)))
    return res;

  if (L4_UNLIKELY(l4_msgtag_words(res) * sizeof(l4_umword_t) < sizeof(*msi_info)))
    return res;

  __builtin_memcpy(msi_info, &m->mr[0], sizeof(*msi_info));
  return res;
}

L4_INLINE l4_msgtag_t
l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode,
                  l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
  mr->mr[0] = L4_ICU_OP_SET_MODE;
  mr->mr[1] = irqnum;
  mr->mr[2] = mode;
  return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 3, 0, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
                 l4_umword_t *label, l4_timeout_t to,
                 l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
  m->mr[0] = L4_ICU_OP_UNMASK + op;
  m->mr[1] = irqnum;
  if (label)
    return l4_ipc_send_and_wait(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 0, 0),
                                label, to);
  else
    return l4_ipc_send(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 0, 0), to);
}

L4_INLINE l4_msgtag_t
l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
              l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
{ return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, utcb); }

L4_INLINE l4_msgtag_t
l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
                l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
{ return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, utcb); }



L4_INLINE l4_msgtag_t
l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW
{ return l4_icu_bind_u(icu, irqnum, irq, l4_utcb()); }

L4_INLINE l4_msgtag_t
l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW
{ return l4_icu_unbind_u(icu, irqnum, irq, l4_utcb()); }
```

```
L4_INLINE l4_msgtag_t
l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW
{ return l4_icu_info_u(icu, info, l4_utcb()); }

L4_INLINE l4_msgtag_t
l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
                l4_icu_msi_info_t *msi_info) L4_NOTHROW
{ return l4_icu_msi_info_u(icu, irqnum, source, msi_info, l4_utcb()); }

L4_INLINE l4_msgtag_t
l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
              l4_timeout_t to) L4_NOTHROW
{ return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, l4_utcb()); }

L4_INLINE l4_msgtag_t
l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
            l4_timeout_t to) L4_NOTHROW
{ return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, l4_utcb()); }

L4_INLINE l4_msgtag_t
l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW
{
  return l4_icu_set_mode_u(icu, irqnum, mode, l4_utcb());
}
```

16.323 l4/sys/ipc.h File Reference

Common IPC interface.

```
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/err.h>
```

Include dependency graph for ipc.h:

This graph shows which files directly or indirectly include this file:

Enumerations

- enum l4_ipc_tcr_error_t {
  L4_IPC_ERROR_MASK = 0x1F , L4_IPC_SND_ERR_MASK = 0x01 , L4_IPC_ENOT_EXISTENT = 0x04 , L4_IPC_RETIMEOUT = 0x03 ,
  L4_IPC_SETIMEOUT = 0x02 , L4_IPC_RECANCELED = 0x07 , L4_IPC_SECANCELED = 0x06 , L4_IPC_REMAPFAILED = 0x11 ,
  L4_IPC_SEMAPFAILED = 0x10 , L4_IPC_RESNDPFTO = 0x0b , L4_IPC_SESNDPFTO = 0x0a , L4_IPC_RERCVPFTO = 0x0d ,
  L4_IPC_SERCVPFTO = 0x0c , L4_IPC_REABORTED = 0x0f , L4_IPC_SEABORTED = 0x0e , L4_IPC_REMSGCUT = 0x09 ,
  L4_IPC_SEMSGCUT = 0x08 }

  Error codes in the error TCR.

Functions

- l4_umword_t l4_ipc_error (l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW

  Get the error code for an object invocation.
- long l4_error (l4_msgtag_t tag) L4_NOTHROW

  Return error code of a system call return message tag or the tag label.
- int l4_ipc_is_snd_error (l4_utcb_t *utcb) L4_NOTHROW

  Returns whether an error occurred in send phase of an invocation.
- int l4_ipc_is_rcv_error (l4_utcb_t *utcb) L4_NOTHROW

  Returns whether an error occurred in receive phase of an invocation.
- int l4_ipc_error_code (l4_utcb_t *utcb) L4_NOTHROW

  Get the error condition of the last invocation from the TCR.
- long l4_ipc_to_errno (unsigned long ipc_error_code) L4_NOTHROW

  Get a negative error code for the given IPC error code.
- l4_msgtag_t l4_ipc_send (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW

  *Send a message to an object (do **not** wait for a reply).*
- l4_msgtag_t l4_ipc_wait (l4_utcb_t *utcb, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW

  Wait for an incoming message from any possible sender.
- l4_msgtag_t l4_ipc_receive (l4_cap_idx_t object, l4_utcb_t *utcb, l4_timeout_t timeout) L4_NOTHROW

  Wait for a message from a specific source.
- l4_msgtag_t l4_ipc_call (l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW

  Object call (usual invocation).
- l4_msgtag_t l4_ipc_reply_and_wait (l4_utcb_t *utcb, l4_msgtag_t tag, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW

  Reply and wait operation (uses the reply capability).
- l4_msgtag_t l4_ipc_send_and_wait (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW

  Send a message and do an open wait.
- l4_msgtag_t l4_ipc (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_umword_t flags, l4_umword_t slabel, l4_msgtag_t tag, l4_umword_t *rlabel, l4_timeout_t timeout) L4_NOTHROW

  Generic L4 object invocation.
- l4_msgtag_t l4_ipc_sleep (l4_timeout_t timeout) L4_NOTHROW

  Sleep for an amount of time.
- int l4_sndfpage_add (l4_fpage_t const snd_fpage, unsigned long snd_base, l4_msgtag_t *tag) L4_NOTHROW

  Add a flex-page to be sent to the UTCB.

16.323.1 Detailed Description

Common IPC interface.

Definition in file ipc.h.

16.323.2 Function Documentation

16.323.2.1 l4_ipc_to_errno()

```
long l4_ipc_to_errno (
            unsigned long ipc_error_code )  [inline]
```

Get a negative error code for the given IPC error code.

Parameters

ipc_error_code	IPC error code as delivered by the kernel. (or returned by the l4_ipc_error_code() function).

Returns

negative error code in the range of L4_EIPC_LO to L4_EIPC_HI.

Definition at line 459 of file ipc.h.

References L4_EIPC_LO.

16.324 ipc.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction.  Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License.  This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #ifndef __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__
00027 #define __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__
00028
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/utcb.h>
00031 #include <l4/sys/err.h>
00032
```

```
/*****************************************************************************
 *** IPC result checking
 *****************************************************************************/

enum l4_ipc_tcr_error_t
{
  L4_IPC_ERROR_MASK       = 0x1F,
  L4_IPC_SND_ERR_MASK     = 0x01,
  L4_IPC_ENOT_EXISTENT    = 0x04,
  L4_IPC_RETIMEOUT        = 0x03,
  L4_IPC_SETIMEOUT        = 0x02,
  L4_IPC_RECANCELED       = 0x07,
  L4_IPC_SECANCELED       = 0x06,
  L4_IPC_REMAPFAILED      = 0x11,
  L4_IPC_SEMAPFAILED      = 0x10,
  L4_IPC_RESNDPFTO        = 0x0b,
  L4_IPC_SESNDPFTO        = 0x0a,
  L4_IPC_RERCVPFTO        = 0x0d,
  L4_IPC_SERCVPFTO        = 0x0c,
  L4_IPC_REABORTED        = 0x0f,
  L4_IPC_SEABORTED        = 0x0e,
  L4_IPC_REMSGCUT         = 0x09,
  L4_IPC_SEMSGCUT         = 0x08,
};


L4_INLINE l4_umword_t
l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;


L4_INLINE long
l4_error(l4_msgtag_t tag) L4_NOTHROW;

L4_INLINE long
l4_error_u(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;

/*****************************************************************************
 *** IPC results
 *****************************************************************************/

L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE int l4_ipc_error_code(l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code) L4_NOTHROW;


/*****************************************************************************
 *** IPC calls
 *****************************************************************************/

L4_INLINE l4_msgtag_t
l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
            l4_timeout_t timeout) L4_NOTHROW;


L4_INLINE l4_msgtag_t
l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *label,
            l4_timeout_t timeout) L4_NOTHROW;


L4_INLINE l4_msgtag_t
l4_ipc_receive(l4_cap_idx_t object, l4_utcb_t *utcb,
               l4_timeout_t timeout) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_ipc_call(l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag,
            l4_timeout_t timeout) L4_NOTHROW;


L4_INLINE l4_msgtag_t
l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
                      l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_ipc_send_and_wait(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
                     l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW;

#if 0
L4_INLINE l4_msgtag_t
l4_ipc_wait_next_period(l4_utcb_t *utcb,
                        l4_umword_t *label,
                        l4_timeout_t timeout);

#endif
```

```
L4_ALWAYS_INLINE l4_msgtag_t
l4_ipc(l4_cap_idx_t dest,
       l4_utcb_t *utcb,
       l4_umword_t flags,
       l4_umword_t slabel,
       l4_msgtag_t tag,
       l4_umword_t *rlabel,
       l4_timeout_t timeout) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW;

L4_INLINE int
l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
                l4_msgtag_t *tag) L4_NOTHROW;

/*
 * \internal
 * \ingroup l4_ipc_api
 */
L4_INLINE int
l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
                  l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW;


/***************************************************************************
 * Implementations
 **********************/

L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code) L4_NOTHROW
{ return -(L4_EIPC_LO + ipc_error_code); }

L4_INLINE l4_msgtag_t
l4_ipc_call(l4_cap_idx_t dest, l4_utcb_t *utcb,
            l4_msgtag_t tag,
            l4_timeout_t timeout) L4_NOTHROW
{
  return l4_ipc(dest, utcb, L4_SYSF_CALL, 0, tag, 0, timeout);
}

L4_INLINE l4_msgtag_t
l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
                      l4_umword_t *label,
                      l4_timeout_t timeout) L4_NOTHROW
{
  return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_REPLY_AND_WAIT, 0, tag, label, timeout);
}

L4_INLINE l4_msgtag_t
l4_ipc_send_and_wait(l4_cap_idx_t dest, l4_utcb_t *utcb,
                     l4_msgtag_t tag,
                     l4_umword_t *src,
                     l4_timeout_t timeout) L4_NOTHROW
{
  return l4_ipc(dest, utcb, L4_SYSF_SEND_AND_WAIT, 0, tag, src, timeout);
}

L4_INLINE l4_msgtag_t
l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb,
            l4_msgtag_t tag,
            l4_timeout_t timeout) L4_NOTHROW
{
  return l4_ipc(dest, utcb, L4_SYSF_SEND, 0, tag, 0, timeout);
}

L4_INLINE l4_msgtag_t
l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *src,
            l4_timeout_t timeout) L4_NOTHROW
{
  l4_msgtag_t t;
  t.raw = 0;
  return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_WAIT, 0, t, src, timeout);
}

L4_INLINE l4_msgtag_t
l4_ipc_receive(l4_cap_idx_t src, l4_utcb_t *utcb,
               l4_timeout_t timeout) L4_NOTHROW
{
  l4_msgtag_t t;
  t.raw = 0;
  return l4_ipc(src, utcb, L4_SYSF_RECV, 0, t, 0, timeout);
}

L4_INLINE l4_msgtag_t
l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW
{ return l4_ipc_receive(L4_INVALID_CAP, NULL, timeout); }
```

```
L4_INLINE l4_umword_t
l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
{
  if (!l4_msgtag_has_error(tag))
    return 0;
  return l4_utcb_tcr_u(utcb)->error & L4_IPC_ERROR_MASK;
}

L4_INLINE long
l4_error_u(l4_msgtag_t tag, l4_utcb_t *u) L4_NOTHROW
{
  if (l4_msgtag_has_error(tag))
    return l4_ipc_to_errno(l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK);

  return l4_msgtag_label(tag);
}

L4_INLINE long
l4_error(l4_msgtag_t tag) L4_NOTHROW
{
  return l4_error_u(tag, l4_utcb());
}


L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *u) L4_NOTHROW
{ return (l4_utcb_tcr_u(u)->error & 1) == 0; }

L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *u) L4_NOTHROW
{ return l4_utcb_tcr_u(u)->error & 1; }

L4_INLINE int l4_ipc_error_code(l4_utcb_t *u) L4_NOTHROW
{ return l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK; }


/*
 * \internal
 * \ingroup l4_ipc_api
 */
L4_INLINE int
l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
                  l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  int i = l4_msgtag_words(*tag) + 2 * l4_msgtag_items(*tag);

  if (i >= L4_UTCB_GENERIC_DATA_SIZE - 1)
    return -L4_ENOMEM;

  v->mr[i]     = snd_base | L4_ITEM_MAP | L4_ITEM_CONT;
  v->mr[i + 1] = snd_fpage.raw;

  *tag = l4_msgtag(l4_msgtag_label(*tag), l4_msgtag_words(*tag),
                   l4_msgtag_items(*tag) + 1, l4_msgtag_flags(*tag));
  return 0;
}

L4_INLINE int
l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
                l4_msgtag_t *tag) L4_NOTHROW
{
  return l4_sndfpage_add_u(snd_fpage, snd_base, tag, l4_utcb());
}


#endif /* ! __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__ */
```

16.325 arm/l4f/l4/sys/ipc.h File Reference

L4 IPC System Calls, ARM.

```
#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>
```

Include dependency graph for ipc.h:

Functions

- l4_msgtag_t l4_ipc (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_umword_t flags, l4_umword_t slabel, l4_msgtag_t tag, l4_umword_t *rlabel, l4_timeout_t timeout) L4_NOTHROW

  Generic L4 object invocation.

16.325.1 Detailed Description

L4 IPC System Calls, ARM.

Definition in file ipc.h.

16.326 ipc.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include_next <l4/sys/ipc.h>
00027
00028 #ifdef __GNUC__
00029
00030 #include <l4/sys/compiler.h>
```

```
#include <l4/sys/syscall_defs.h>

L4_INLINE l4_msgtag_t
l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb,
       l4_umword_t flags,
       l4_umword_t slabel,
       l4_msgtag_t tag,
       l4_umword_t *rlabel,
       l4_timeout_t timeout) L4_NOTHROW
{
  register l4_umword_t _dest     __asm__("r2") = dest | flags;
  register l4_umword_t _timeout  __asm__("r3") = timeout.raw;
  register l4_mword_t _tag       __asm__("r0") = tag.raw;
  register l4_umword_t _label    __asm__("r4") = slabel;
  (void)utcb;

  __asm__ __volatile__
    ("mov lr, pc    \n"
     "mov pc, %[sc] \n"
     :
     "+r" (_dest),
     "+r" (_timeout),
     "+r" (_label),
     "+r" (_tag)
     :
     [sc] "i" (L4_SYSCALL_INVOKE)
     :
     "cc", "memory", "lr");

  if (rlabel)
    *rlabel = _label;
  tag.raw = _tag;

  return tag;
}

#endif //__GNUC__
```

16.327 x86/l4f/l4/sys/ipc.h File Reference

L4 IPC System Calls, x86.

```
#include <l4/sys/types.h>
#include <l4/sys/ipc-invoke.h>
#include "ipc-l42-gcc3-nopic.h"
```

Include dependency graph for ipc.h:

16.327.1 Detailed Description

L4 IPC System Calls, x86.

Definition in file ipc.h.

16.328 ipc.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Lars Reuther <reuther@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #ifndef __L4_IPC_H__
00026 #define __L4_IPC_H__
00027
00028 #include <l4/sys/types.h>
00029
00030 #include_next <l4/sys/ipc.h>
00031
00032 /******************************************************************************
00033  *** Implementation
00034  ******************************************************************************/
00035
00036 #include <l4/sys/ipc-invoke.h>
00037 #include "ipc-l42-gcc3-nopic.h"
00038
00039 #endif /* !__L4_IPC_H__ */
```

16.329 l4/sys/ipc_gate File Reference

The C++ IPC gate interface.

```
#include <l4/sys/ipc_gate.h>
#include <l4/sys/capability>
#include <l4/sys/rcv_endpoint>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for ipc_gate:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Ipc_gate

  The C++ IPC gate interface, see IPC-Gate API for the C interface.

Namespaces

- L4

  L4 low-level kernel interface.

16.329.1 Detailed Description

The C++ IPC gate interface.

Definition in file ipc_gate.

16.330 ipc_gate

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/ipc_gate.h>
00027 #include <l4/sys/capability>
00028 #include <l4/sys/rcv_endpoint>
00029 #include <l4/sys/cxx/ipc_iface>
00030
00031 namespace L4 {
00032
00033 class Thread;
00034
00089 class L4_EXPORT Ipc_gate :
00090   public Kobject_t<Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT,
00091                    Type_info::Demand_t<1> >
00092 {
00093 public:
00101   L4_INLINE_RPC_OP(L4_IPC_GATE_GET_INFO_OP,
00102       l4_msgtag_t, get_infos, (l4_umword_t *label));
00103
00104   typedef L4::Typeid::Rpcs_sys<bind_thread_t, get_infos_t> Rpcs;
00105 };
00106
00107 }
```

16.331 l4/sys/ipc_gate.h File Reference

The C IPC gate interface, see L4::Ipc_gate for the C++ interface.

```
#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
#include <l4/sys/rcv_endpoint.h>
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for ipc_gate.h:

This graph shows which files directly or indirectly include this file:

Enumerations

- enum L4_ipc_gate_ops { L4_IPC_GATE_BIND_OP = 0x10 , L4_IPC_GATE_GET_INFO_OP = 0x11 }

  Operations on the IPC-gate.

Functions

- l4_msgtag_t l4_ipc_gate_get_infos (l4_cap_idx_t gate, l4_umword_t *label)

  Get information about the IPC-gate.

16.331.1 Detailed Description

The C IPC gate interface, see L4::Ipc_gate for the C++ interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the Factory interface.

Depending on the permissions of the capability used, an IPC gate forwards IPC to the Thread that is *bound* to the IPC gate (cf. l4_ipc_gate_bind_thread()). If the capability has the L4_FPAGE_C_IPCGATE_SVR permission, only IPC using a protocol different from the L4_PROTO_KOBJECT protocol is forwarded. Without the L4_FPAGE_C_IPCGATE_SVR permission, all IPC is forwarded. The latter is the usual case for a client in a client/server scenario. When no thread is bound yet, the forwarded IPC blocks until a thread is bound or the IPC times out.

Forwarded IPC is always forwarded to the userland of the bound thread. That means, the Thread interface of the bound thread is not accessible via an IPC gate. The IPC-Gate API of an IPC gate is only accessible if the capability used has the L4_FPAGE_C_IPCGATE_SVR permission (cf. previous paragraph). Conversely that means, if the capability used lacks the L4_FPAGE_C_IPCGATE_SVR permission, IPC-Gate API calls are forwarded to the bound thread instead of being processed by the IPC gate itself. In a client/server scenario, a client should only get IPC gate capabilities without L4_FPAGE_C_IPCGATE_SVR permission so the client cannot tamper with the IPC gate.

When binding a thread to an IPC gate, a user-defined, kernel protected, machine-word sized payload called the IPC gate's *label* is assigned to the IPC gate (cf. l4_ipc_gate_bind_thread()). When a send-only IPC or call IPC is forwarded via an IPC gate, the label provided by the sender is ignored and replaced by the IPC gate's label where the two least significant bits are the result of bitwise disjunction of the corresponding label bits with the L4_CAP_FPAGE_S and L4_CAP_FPAGE_W permissions of the capability used. Hence, the label provided via l4_ipc_gate_bind_thread() should usually have its two least significant bits set to zero. The replaced label is only visible to the bound thread upon receive. However, the configured label of an IPC gate can also be queried via l4_ipc_gate_get_infos() if the capability used has the L4_FPAGE_C_IPCGATE_SVR permission.

When deleting an IPC gate or when unbinding it from a thread, the label of IPC already in flight won't be changed. To ensure that no IPC from this IPC gate is received by a thread with an unexpected label, l4_thread_modify_sender_start() shall be used to change the labels of every pending IPC to that gate. This is also required if the label of an already bound IPC gate is changed. It is not necessary after binding the IPC gate to a thread for the first time.

Include File

```
#include <l4/sys/ipc_gate.h>
```

For the C++ interface refer to the L4::Ipc_gate documentation.

See also

Object Invocation

Definition in file ipc_gate.h.

16.332 ipc_gate.h

```
/*
 * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */

#pragma once

#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
#include <l4/sys/rcv_endpoint.h>

L4_INLINE l4_msgtag_t
l4_ipc_gate_get_infos(l4_cap_idx_t gate, l4_umword_t *label);

L4_INLINE l4_msgtag_t
l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label, l4_utcb_t *utcb);

enum L4_ipc_gate_ops
{
  L4_IPC_GATE_BIND_OP     = 0x10,
  L4_IPC_GATE_GET_INFO_OP = 0x11,
};


/* IMPLEMENTATION ----------------------------------------------------------*/

#include <l4/sys/ipc.h>

L4_INLINE l4_msgtag_t
l4_ipc_gate_bind_thread_u(l4_cap_idx_t gate,
                          l4_cap_idx_t thread, l4_umword_t label,
                          l4_utcb_t *utcb)
{
  return l4_rcv_ep_bind_thread_u(gate, thread, label, utcb);
}

L4_INLINE l4_msgtag_t
l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label, l4_utcb_t *utcb)
{
  l4_msgtag_t tag;
  l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
  m->mr[0] = L4_IPC_GATE_GET_INFO_OP;
  tag = l4_ipc_call(gate, utcb, l4_msgtag(L4_PROTO_KOBJECT, 1, 0, 0),
                    L4_IPC_NEVER);
  if (!l4_msgtag_has_error(tag) && l4_msgtag_label(tag) >= 0)
    *label = m->mr[0];

  return tag;
}



L4_INLINE l4_msgtag_t
l4_ipc_gate_bind_thread(l4_cap_idx_t gate, l4_cap_idx_t thread,
                        l4_umword_t label)
{
  return l4_rcv_ep_bind_thread_u(gate, thread, label, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_ipc_gate_get_infos(l4_cap_idx_t gate, l4_umword_t *label)
{
  return l4_ipc_gate_get_infos_u(gate, label, l4_utcb());
}
```

16.333 l4/sys/irq File Reference

C++ Irq interface.

```
#include <l4/sys/icu.h>
#include <l4/sys/irq.h>
#include <l4/sys/capability>
#include <l4/sys/rcv_endpoint>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_types>
```

Include dependency graph for irq:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Irq_eoi

Interface for sending an unmask message to an object.

- struct L4::Triggerable

  Interface that allows an object to be triggered by some source.

- class L4::Irq

  C++ Irq interface, see IRQs for the C interface.

- struct L4::Irq_mux

  IRQ multiplexer for shared IRQs.

- class L4::Icu

  C++ Icu interface, see Interrupt controller for the C interface.

- class L4::Icu::Info

  This class encapsulates information about an ICU.

Namespaces

- L4

  L4 low-level kernel interface.

16.333.1 Detailed Description

C++ Irq interface.

Definition in file irq.

16.334 irq

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/icu.h>
00028 #include <l4/sys/irq.h>
00029 #include <l4/sys/capability>
00030 #include <l4/sys/rcv_endpoint>
00031 #include <l4/sys/cxx/ipc_iface>
00032 #include <l4/sys/cxx/ipc_types>
00033
00034 namespace L4 {
00035
00048 class Irq_eoi : public Kobject_0t<Irq_eoi, L4::PROTO_EMPTY>
00049 {
00050 public:
00076   l4_msgtag_t unmask(unsigned irqnum, l4_umword_t *label = 0,
00077                      l4_timeout_t to = L4_IPC_NEVER,
00078                      l4_utcb_t *utcb = l4_utcb()) noexcept
00079   {
00080     return l4_icu_control_u(cap(), irqnum, L4_ICU_CTL_UNMASK, label, to, utcb);
00081   }
```

```
00082 };
00083
00091 struct Triggerable : Kobject_t<Triggerable, Irq_eoi, L4_PROTO_IRQ>
00092 {
00106   l4_msgtag_t trigger(l4_utcb_t *utcb = l4_utcb()) noexcept
00107   { return l4_irq_trigger_u(cap(), utcb); }
00108 };
00109
00135 class Irq : public Kobject_2t<Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER>
00136 {
00137 public:
00138   using Triggerable::unmask;
00139
00147   l4_msgtag_t detach(l4_utcb_t *utcb = l4_utcb()) noexcept
00148   { return l4_irq_detach_u(cap(), utcb); }
00149
00150
00162   l4_msgtag_t receive(l4_timeout_t timeout = L4_IPC_NEVER,
00163                       l4_utcb_t *utcb = l4_utcb()) noexcept
00164   { return l4_irq_receive_u(cap(), timeout, utcb); }
00165
00175   l4_msgtag_t wait(l4_umword_t *label, l4_timeout_t timeout = L4_IPC_NEVER,
00176                    l4_utcb_t *utcb = l4_utcb()) noexcept
00177   { return unmask(-1, label, timeout, utcb); }
00178
00198   l4_msgtag_t unmask(l4_utcb_t *utcb = l4_utcb()) noexcept
00199   { return unmask(-1, 0, L4_IPC_NEVER, utcb); }
00200 };
00201
00217 struct Irq_mux : Kobject_t<Irq_mux, Triggerable, L4_PROTO_IRQ_MUX>
00218 {
00232   l4_msgtag_t chain(Cap<Triggerable> const &slave,
00233                     l4_utcb_t *utcb = l4_utcb()) noexcept
00234   { return l4_irq_mux_chain_u(cap(), slave.cap(), utcb); }
00235 };
00236
00237
00262 class Icu :
00263   public Kobject_t<Icu, Irq_eoi, L4_PROTO_IRQ,
00264                    Type_info::Demand_t<1> >
00265 {
00266 public:
00267   enum Mode
00268   {
00269     F_none         = L4_IRQ_F_NONE,
00270     F_level_high   = L4_IRQ_F_LEVEL_HIGH,
00271     F_level_low    = L4_IRQ_F_LEVEL_LOW,
00272     F_pos_edge     = L4_IRQ_F_POS_EDGE,
00273     F_neg_edge     = L4_IRQ_F_NEG_EDGE,
00274     F_both_edge    = L4_IRQ_F_BOTH_EDGE,
00275     F_mask         = L4_IRQ_F_MASK,
00276
00277     F_set_wakeup   = L4_IRQ_F_SET_WAKEUP,
00278     F_clear_wakeup = L4_IRQ_F_CLEAR_WAKEUP,
00279   };
00280
00281   enum Flags
00282   {
00283     F_msi = L4_ICU_FLAG_MSI
00284   };
00285
00289   class Info : public l4_icu_info_t
00290   {
00291   public:
00293     bool supports_msi() const noexcept { return features & F_msi; }
00294   };
00295
00321   l4_msgtag_t bind(unsigned irqnum, L4::Cap<Triggerable> irq,
00322                    l4_utcb_t *utcb = l4_utcb()) noexcept
00323   { return l4_icu_bind_u(cap(), irqnum, irq.cap(), utcb); }
00324
00325   L4_RPC_NF_OP(
00326     L4_ICU_OP_BIND,
00327     l4_msgtag_t, bind, (l4_umword_t irqnum, Ipc::Cap<Irq> irq)
00328   );
00329
00339   l4_msgtag_t unbind(unsigned irqnum, L4::Cap<Triggerable> irq,
00340                      l4_utcb_t *utcb = l4_utcb()) noexcept
00341   { return l4_icu_unbind_u(cap(), irqnum, irq.cap(), utcb); }
00342
00343   L4_RPC_NF_OP(
00344     L4_ICU_OP_UNBIND,
00345     l4_msgtag_t, unbind, (l4_umword_t irqnum, Ipc::Cap<Irq> irq)
00346   );
00347
00356   l4_msgtag_t info(l4_icu_info_t *info, l4_utcb_t *utcb = l4_utcb()) noexcept
00357   { return l4_icu_info_u(cap(), info, utcb); }
```

```
00358
00359   struct _Info { l4_umword_t features, nr_irqs, nr_msis; };
00360   L4_RPC_NF_OP(L4_ICU_OP_INFO, l4_msgtag_t, info, (_Info *info));
00361
00374   L4_INLINE_RPC_OP(L4_ICU_OP_MSI_INFO,
00375       l4_msgtag_t, msi_info, (l4_umword_t irqnum, l4_uint64_t source,
00376                               l4_icu_msi_info_t *msi_info));
00377
00381   l4_msgtag_t control(unsigned irqnum, unsigned op, l4_umword_t *label,
00382                       l4_timeout_t to, l4_utcb_t *utcb = l4_utcb()) noexcept
00383   { return l4_icu_control_u(cap(), irqnum, op, label, to, utcb); }
00384
00399   l4_msgtag_t mask(unsigned irqnum,
00400                    l4_umword_t *label = 0,
00401                    l4_timeout_t to = L4_IPC_NEVER,
00402                    l4_utcb_t *utcb = l4_utcb()) noexcept
00403   { return l4_icu_mask_u(cap(), irqnum, label, to, utcb); }
00404
00405   L4_RPC_NF_OP(
00406     L4_ICU_OP_MASK,
00407     l4_msgtag_t, mask, (l4_umword_t irqnum),
00408     L4::Ipc::Send_only
00409   );
00410
00411
00412   L4_RPC_NF_OP(
00413     L4_ICU_OP_UNMASK,
00414     l4_msgtag_t, unmask, (l4_umword_t irqnum),
00415     L4::Ipc::Send_only
00416   );
00417
00427   l4_msgtag_t set_mode(unsigned irqnum, l4_umword_t mode,
00428                        l4_utcb_t *utcb = l4_utcb()) noexcept
00429   { return l4_icu_set_mode_u(cap(), irqnum, mode, utcb); }
00430
00431   L4_RPC_NF_OP(
00432     L4_ICU_OP_SET_MODE,
00433     l4_msgtag_t, set_mode, (l4_umword_t irqnum, l4_umword_t mode)
00434   );
00435
00436   typedef L4::Typeid::Rpcs_sys<
00437     bind_t, unbind_t, info_t, msi_info_t, unmask_t, mask_t, set_mode_t
00438   > Rpcs;
00439 };
00440
00441 }
```

16.335 l4/sys/kernel_object.h File Reference

Kernel object system calls.

```
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/__kernel_object_impl.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for kernel_object.h:

This graph shows which files directly or indirectly include this file:

16.335.1 Detailed Description

Kernel object system calls.

Definition in file kernel_object.h.

16.336 kernel_object.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *               Björn Döbel <doebel@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
```

```
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#ifndef __L4SYS__KERNEL_OBJECT_H__
#define __L4SYS__KERNEL_OBJECT_H__

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>

L4_INLINE l4_msgtag_t
l4_invoke_debugger(l4_cap_idx_t obj, l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;


/*****************************************************************************
 * Implementation
 *****************************************************************************/

#include <l4/sys/__kernel_object_impl.h>
#include <l4/sys/ipc.h>

enum L4_kobject_op {
  L4_KOBJECT_OP_DEC_REFCNT = 0,
  L4_KOBJECT_OP_REGISTER_IRQ,
};

L4_INLINE l4_msgtag_t
l4_kobject_dec_refcnt_u(l4_cap_idx_t obj, l4_mword_t diff, l4_utcb_t *u) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_kobject_dec_refcnt_u(l4_cap_idx_t obj, l4_mword_t diff, l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *m = l4_utcb_mr_u(u);
  m->mr[0] = L4_KOBJECT_OP_DEC_REFCNT;
  m->mr[1] = diff;
  return l4_ipc_call(obj, u, l4_msgtag(L4_PROTO_KOBJECT, 2, 0, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff) L4_NOTHROW
{
  return l4_kobject_dec_refcnt_u(obj, diff, l4_utcb());
}

#endif /* ! __L4SYS__KERNEL_OBJECT_H__ */
```

16.337 l4/sys/kip File Reference

```
#include <l4/cxx/static_vector>
```

Include dependency graph for kip:

Data Structures

- class L4::Kip::Mem_desc

  Memory descriptors stored in the kernel interface page.

Namespaces

- L4

  L4 low-level kernel interface.

16.337.1 Detailed Description

L4::Kip class, memory descriptors.

Author

Alexander Warg `alexander.warg@os.inf.tu-dresden.de`

Definition in file kip.

16.338 kip

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00010 /*
00011  * (c) 2008-2009 Author(s)
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU General Public License 2.
00016  * Please see the COPYING-GPL-2 file for details.
00017  *
00018  * As a special exception, you may use this file as part of a free software
00019  * library without restriction.  Specifically, if other files instantiate
00020  * templates or use macros or inline functions from this file, or you compile
00021  * this file and link it with other files to produce an executable, this
00022  * file does not by itself cause the resulting executable to be covered by
00023  * the GNU General Public License.  This exception does not however
00024  * invalidate any other reasons why the executable file might be covered by
00025  * the GNU General Public License.
00026  */
00027 #ifndef L4_SYS_KIP_H__
00028 #define L4_SYS_KIP_H__
00029
00030 #include <l4/cxx/static_vector>
00031
00032 /* C++ version of memory descriptors */
00033
00043 namespace L4
00044 {
00045   namespace Kip
00046   {
00053     class Mem_desc
00054     {
00055     public:
00059       enum Mem_type
00060       {
00061         Undefined    = 0x0,
00062         Conventional = 0x1,
00063         Reserved     = 0x2,
00064         Dedicated    = 0x3,
00065         Shared       = 0x4,
00066
00067         Info         = 0xd,
00068         Bootloader   = 0xe,
00069         Arch         = 0xf
00070       };
00071
00075       enum Info_sub_type
00076       {
00077         Info_acpi_rsdp = 0
00078       };
00079
00080     private:
00081       unsigned long _l, _h;
00082
00083       static unsigned long &memory_info(void *kip) noexcept
00084       { return *((unsigned long *)kip + 21); }
00085
00086       static unsigned long memory_info(void const *kip) noexcept
00087       { return *((unsigned long const *)kip + 21); }
00088
00089     public:
00097       static Mem_desc *first(void *kip) noexcept
00098       {
00099         return (Mem_desc *)((char *)kip
00100                + (memory_info(kip) » ((sizeof(unsigned long) / 2) * 8)));
00101       }
00102
00103       static Mem_desc const *first(void const *kip) noexcept
00104       {
00105         return (Mem_desc const *)((char const *)kip
00106                + (memory_info(kip) » ((sizeof(unsigned long) / 2) * 8)));
00107       }
00108
00116       static unsigned long count(void const *kip) noexcept
00117       {
00118         return memory_info(kip)
00119                & ((1UL « ((sizeof(unsigned long) / 2) * 8)) - 1);
00120       }
00121
00128       static void count(void *kip, unsigned count) noexcept
00129       {
00130         unsigned long &mi = memory_info(kip);
00131         mi = (mi & ~((1UL « ((sizeof(unsigned long) / 2) * 8)) - 1)) | count;
00132       }
00133
00139       static inline cxx::static_vector<Mem_desc const> all(void const *kip)
```

```
00140       {
00141         return cxx::static_vector<Mem_desc const>(Mem_desc::first(kip),
00142                                                   Mem_desc::count(kip));
00143       }
00144
00150       static inline cxx::static_vector<Mem_desc> all(void *kip)
00151       {
00152         return cxx::static_vector<Mem_desc>(Mem_desc::first(kip),
00153                                             Mem_desc::count(kip));
00154       }
00155
00166       Mem_desc(unsigned long start, unsigned long end,
00167                Mem_type t, unsigned char st = 0, bool virt = false) noexcept
00168       : _l((start & ~0x3ffUL) | (t & 0x0f) | ((st « 4) & 0x0f0)
00169            | (virt ? 0x0200 : 0x0)), _h(end | 0x3ffUL)
00170       {}
00171
00177       unsigned long start() const noexcept { return _l & ~0x3ffUL; }
00178
00184       unsigned long end() const noexcept { return _h | 0x3ffUL; }
00185
00191       unsigned long size() const noexcept { return end() + 1 - start(); }
00192
00198       Mem_type type() const noexcept { return (Mem_type)(_l & 0x0f); }
00199
00205       unsigned char sub_type() const noexcept { return (_l » 4) & 0x0f; }
00206
00213       unsigned is_virtual() const noexcept { return _l & 0x200; }
00214
00224       void set(unsigned long start, unsigned long end,
00225                Mem_type t, unsigned char st = 0, bool virt = false) noexcept
00226       {
00227         _l = (start & ~0x3ffUL) | (t & 0x0f) | ((st « 4) & 0x0f0)
00228              | (virt?0x0200:0x0);
00229
00230         _h = end | 0x3ffUL;
00231       }
00232
00233     };
00234   };
00235 };
00236
00237 #endif
```

16.339 l4/sys/ktrace.h File Reference

L4 kernel event tracing.

```
#include <l4/sys/types.h>
#include <l4/sys/ktrace_events.h>
#include <l4/sys/__ktrace-impl.h>
```

Include dependency graph for ktrace.h:

Functions

- l4_umword_t fiasco_tbuf_log (const char *text)

  Create new trace-buffer entry with describing <text>.
- l4_umword_t fiasco_tbuf_log_3val (const char *text, l4_umword_t v1, l4_umword_t v2, l4_umword_t v3)

  Create new trace-buffer entry with describing <text> and three additional values.
- l4_umword_t fiasco_tbuf_log_binary (const unsigned char *data)

  Create new trace-buffer entry with binary data.
- void fiasco_tbuf_clear (void)

  Clear trace-buffer.
- void fiasco_tbuf_dump (void)

  Dump trace-buffer to kernel console.

16.339.1 Detailed Description

L4 kernel event tracing.

Definition in file ktrace.h.

16.340 ktrace.h

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
 *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
 *     2015      Adam Lackorzynski <adam@l4re.org>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
/*****************************************************************************/
#ifndef __L4_KTRACE_H__
#define __L4_KTRACE_H__

#include <l4/sys/types.h>
#include <l4/sys/ktrace_events.h>

L4_INLINE l4_umword_t
fiasco_tbuf_log(const char *text);

L4_INLINE l4_umword_t
fiasco_tbuf_log_3val(const char *text, l4_umword_t v1, l4_umword_t v2, l4_umword_t v3);

L4_INLINE l4_umword_t
fiasco_tbuf_log_binary(const unsigned char *data);

L4_INLINE void
fiasco_tbuf_clear(void);

L4_INLINE void
fiasco_tbuf_dump(void);

#include <l4/sys/__ktrace-impl.h>

#endif
```

16.341 l4/sys/l4int.h File Reference

Fixed sized integer types, generic version.

This graph shows which files directly or indirectly include this file:

Typedefs

- typedef signed char l4_int8_t

  Signed 8bit value.
- typedef unsigned char l4_uint8_t

  Unsigned 8bit value.
- typedef signed short int l4_int16_t

  Signed 16bit value.

- typedef unsigned short int l4_uint16_t

  Unsigned 16bit value.
- typedef signed int l4_int32_t

  Signed 32bit value.
- typedef unsigned int l4_uint32_t

  Unsigned 32bit value.
- typedef signed long long l4_int64_t

  Signed 64bit value.
- typedef unsigned long long l4_uint64_t

  Unsigned 64bit value.
- typedef unsigned long l4_addr_t

  Address type.
- typedef signed long l4_mword_t

  Signed machine word.
- typedef unsigned long l4_umword_t

  Unsigned machine word.
- typedef l4_uint64_t l4_cpu_time_t

  CPU clock type.
- typedef l4_uint64_t l4_kernel_clock_t

  Kernel clock type.

16.341.1 Detailed Description

Fixed sized integer types, generic version.

Definition in file l4int.h.

16.342 l4int.h

```
00001
00013 /*
00014  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00015  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00016  *     economic rights: Technische Universität Dresden (Germany)
00017  *
00018  * This file is part of TUD:OS and distributed under the terms of the
00019  * GNU General Public License 2.
00020  * Please see the COPYING-GPL-2 file for details.
00021  *
00022  * As a special exception, you may use this file as part of a free software
00023  * library without restriction.  Specifically, if other files instantiate
00024  * templates or use macros or inline functions from this file, or you compile
00025  * this file and link it with other files to produce an executable, this
00026  * file does not by itself cause the resulting executable to be covered by
00027  * the GNU General Public License.  This exception does not however
00028  * invalidate any other reasons why the executable file might be covered by
00029  * the GNU General Public License.
00030  */
00031 #ifndef __L4_SYS_L4INT_H__
00032 #define __L4_SYS_L4INT_H__
00033
00034 /* fixed sized data types */
00035 typedef signed char             l4_int8_t;
00036 typedef unsigned char           l4_uint8_t;
00037 typedef signed short int        l4_int16_t;
00038 typedef unsigned short int      l4_uint16_t;
00039 typedef signed int              l4_int32_t;
00040 typedef unsigned int            l4_uint32_t;
00041 typedef signed long long        l4_int64_t;
00042 typedef unsigned long long      l4_uint64_t;
00044 /* some common data types */
00045 typedef unsigned long           l4_addr_t;
00048 typedef signed long             l4_mword_t;
00051 typedef unsigned long           l4_umword_t;
00058 typedef l4_uint64_t l4_cpu_time_t;
00059
00064 typedef l4_uint64_t l4_kernel_clock_t;
00065
00066 #endif /* !__L4_SYS_L4INT_H__ */
```

16.343 arm/l4/sys/l4int.h File Reference

Fixed sized integer types, arm version.

Macros

- #define L4_MWORD_BITS 32

  Size of machine words in bits.

Typedefs

- typedef unsigned int l4_size_t

  Unsigned size type.
- typedef signed int l4_ssize_t

  Signed size type.

16.343.1 Detailed Description

Fixed sized integer types, arm version.

Definition in file l4int.h.

16.344 l4int.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include_next <l4/sys/l4int.h>
00027
00033 #define L4_MWORD_BITS           32
00035 typedef unsigned int            l4_size_t;
00036 typedef signed int              l4_ssize_t;
```

16.345 amd64/l4/sys/l4int.h File Reference

Fixed sized integer types, amd64 version.

Macros

- #define L4_MWORD_BITS 64

  Size of machine words in bits.

Typedefs

- typedef unsigned long l4_size_t

  Unsigned size type.
- typedef signed long l4_ssize_t

  Signed size type.

16.345.1 Detailed Description

Fixed sized integer types, amd64 version.

Definition in file l4int.h.

16.346 l4int.h

```
00001 /****************************************************************************/
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include_next <l4/sys/l4int.h>
00028
00034 #define L4_MWORD_BITS           64
00036 typedef unsigned long           l4_size_t;
00037 typedef signed long             l4_ssize_t;
```

16.347 x86/l4/sys/l4int.h File Reference

Fixed sized integer types, x86 version.

Macros

- #define L4_MWORD_BITS 32

  Size of machine words in bits.

Typedefs

- typedef unsigned int l4_size_t

  Unsigned size type.
- typedef signed int l4_ssize_t

  Signed size type.

16.347.1 Detailed Description

Fixed sized integer types, x86 version.

Definition in file l4int.h.

16.348 l4int.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include_next <l4/sys/l4int.h>
00027
00033 #define L4_MWORD_BITS           32
00035 typedef unsigned int            l4_size_t;
00036 typedef signed int              l4_ssize_t;
```

16.349 l4/sys/memdesc.h File Reference

Memory description functions.

```
#include <l4/sys/kip.h>
```

Include dependency graph for memdesc.h:

Data Structures

- struct l4_kernel_info_mem_desc_t

  Memory descriptor data structure.

Typedefs

- typedef struct l4_kernel_info_mem_desc_t l4_kernel_info_mem_desc_t

  Memory descriptor data structure.

Enumerations

- enum l4_mem_type_t {
  l4_mem_type_undefined = 0x0 , l4_mem_type_conventional = 0x1 , l4_mem_type_reserved = 0x2 , l4_mem_type_dedicated = 0x3 ,
  l4_mem_type_shared = 0x4 , l4_mem_type_info = 0xd , l4_mem_type_bootloader = 0xe , l4_mem_type_archspecific = 0xf }

  Type of a memory descriptor.
- enum l4_mem_info_sub_type_t { l4_mem_info_acpi_rsdp = 0 }

  Memory sub types for l4_mem_type_info descriptors.

Functions

- l4_kernel_info_mem_desc_t * l4_kernel_info_get_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW

  Get pointer to memory descriptors from KIP.
- unsigned l4_kernel_info_get_num_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW

  Get number of memory descriptors in KIP.
- void l4_kernel_info_set_mem_desc (l4_kernel_info_mem_desc_t *md, l4_addr_t start, l4_addr_t end, unsigned type, unsigned virt, unsigned sub_type) L4_NOTHROW

  Populate a memory descriptor.
- l4_umword_t l4_kernel_info_get_mem_desc_start (l4_kernel_info_mem_desc_t *md) L4_NOTHROW

  Get start address of the region described by the memory descriptor.
- l4_umword_t l4_kernel_info_get_mem_desc_end (l4_kernel_info_mem_desc_t *md) L4_NOTHROW

  Get end address of the region described by the memory descriptor.
- l4_umword_t l4_kernel_info_get_mem_desc_type (l4_kernel_info_mem_desc_t *md) L4_NOTHROW

  Get type of the memory region.
- l4_umword_t l4_kernel_info_get_mem_desc_subtype (l4_kernel_info_mem_desc_t *md) L4_NOTHROW

  Get sub-type of memory region.
- l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (l4_kernel_info_mem_desc_t *md) L4_NOTHROW

  Get virtual flag of the memory descriptor.

16.349.1 Detailed Description

Memory description functions.

Definition in file memdesc.h.

16.350 memdesc.h

```
00001
00006 /*
00007  * (c) 2007-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #ifndef __L4SYS__MEMDESC_H__
00025 #define __L4SYS__MEMDESC_H__
00026
00027 #include <l4/sys/kip.h>
00028
00044 enum l4_mem_type_t
00045 {
00046   l4_mem_type_undefined    = 0x0,
00047   l4_mem_type_conventional = 0x1,
00048   l4_mem_type_reserved     = 0x2,
00049   l4_mem_type_dedicated    = 0x3,
00050   l4_mem_type_shared       = 0x4,
00051
00052   l4_mem_type_info         = 0xd,
00053   l4_mem_type_bootloader   = 0xe,
00054   l4_mem_type_archspecific = 0xf,
00055 };
00056
```

```
enum l4_mem_info_sub_type_t
{
  l4_mem_info_acpi_rsdp = 0
};


typedef struct l4_kernel_info_mem_desc_t
{
  l4_umword_t l;
  l4_umword_t h;
} l4_kernel_info_mem_desc_t;


L4_INLINE
l4_kernel_info_mem_desc_t *
l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW;

L4_INLINE
unsigned
l4_kernel_info_get_num_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW;

L4_INLINE
void
l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *md,
                            l4_addr_t start,
          l4_addr_t end,
          unsigned type,
          unsigned virt,
          unsigned sub_type) L4_NOTHROW;

L4_INLINE
l4_umword_t
l4_kernel_info_get_mem_desc_start(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;

L4_INLINE
l4_umword_t
l4_kernel_info_get_mem_desc_end(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;

L4_INLINE
l4_umword_t
l4_kernel_info_get_mem_desc_type(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;

L4_INLINE
l4_umword_t
l4_kernel_info_get_mem_desc_subtype(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;

L4_INLINE
l4_umword_t
l4_kernel_info_get_mem_desc_is_virtual(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;

/**************************************************************************
 * Implementations
 **************************************************************************/

L4_INLINE
l4_kernel_info_mem_desc_t *
l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW
{
  return (l4_kernel_info_mem_desc_t *)(((l4_addr_t)kip)
      + (kip->mem_info » (sizeof(l4_umword_t) * 4)));
}

L4_INLINE
unsigned
l4_kernel_info_get_num_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW
{
  return kip->mem_info & ((1UL « (sizeof(l4_umword_t)*4)) -1);
}

L4_INLINE
void
l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *md,
                            l4_addr_t start,
          l4_addr_t end,
          unsigned type,
          unsigned virt,
          unsigned sub_type) L4_NOTHROW
{
  md->l = (start & ~0x3ffUL) | (type & 0x0f) | ((sub_type « 4) & 0x0f0)
    | (virt ? 0x200 : 0x0);
  md->h = end;
}


L4_INLINE
l4_umword_t
l4_kernel_info_get_mem_desc_start(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
```

```
{
  return md->l & ~0x3ffUL;
}

L4_INLINE
l4_umword_t
l4_kernel_info_get_mem_desc_end(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
{
  return md->h | 0x3ffUL;
}

L4_INLINE
l4_umword_t
l4_kernel_info_get_mem_desc_type(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
{
  return md->l & 0xf;
}

L4_INLINE
l4_umword_t
l4_kernel_info_get_mem_desc_subtype(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
{
  return (md->l & 0xf0) » 4;
}

L4_INLINE
l4_umword_t
l4_kernel_info_get_mem_desc_is_virtual(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
{
  return md->l & 0x200;
}

#endif /* ! __L4SYS__MEMDESC_H__ */
```

16.351 l4/sys/meta File Reference

Meta interface for getting dynamic type information about objects behind capabilities.

```
#include "kobject"
#include "cxx/ipc_iface"
#include "cxx/ipc_string"
```

Include dependency graph for meta:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Meta

  Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects.

Namespaces

- L4

  L4 low-level kernel interface.

16.351.1 Detailed Description

Meta interface for getting dynamic type information about objects behind capabilities.

Definition in file meta.

16.352 meta

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include "kobject"
00028 #include "cxx/ipc_iface"
00029 #include "cxx/ipc_string"
00030
00031 namespace L4 {
00032
00037 class Meta : public Kobject_t<Meta, Kobject, L4_PROTO_META>
00038 {
00039 public:
00046   L4_INLINE_RPC(l4_msgtag_t, num_interfaces, ());
00047
00061   L4_INLINE_RPC(l4_msgtag_t, interface, (l4_umword_t idx, long *proto,
00062                                          L4::Ipc::String<char> *name));
00063
00075   L4_INLINE_RPC(l4_msgtag_t, supports, (l4_mword_t protocol));
00076
00077   typedef L4::Typeid::Rpcs<num_interfaces_t, interface_t, supports_t> Rpcs;
00078 };
00079
00080 }
```

16.353 l4/sys/pager File Reference

Pager and Io_pager C++ interface.

```
#include <l4/sys/capability>
#include <l4/sys/types.h>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for pager:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Io_pager

  Io_pager interface.
- class L4::Pager

  Pager interface including the Io_pager interface.

Namespaces

- L4

  L4 low-level kernel interface.

16.353.1 Detailed Description

Pager and Io_pager C++ interface.

Definition in file pager.

16.354 pager

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/sys/capability>
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/cxx/ipc_types>
00028 #include <l4/sys/cxx/ipc_iface>
00029
00030 namespace L4 {
00031
00061 class L4_EXPORT Io_pager :
00062   public Kobject_0t<Io_pager, L4_PROTO_IO_PAGE_FAULT>
00063 {
00064 public:
00080   L4_INLINE_RPC(
00081       l4_msgtag_t, io_page_fault, (l4_fpage_t io_pfa, l4_umword_t pc,
00082                                    L4::Ipc::Rcv_fpage rwin,
00083                                    L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00084
00085   typedef L4::Typeid::Rpc_nocode<io_page_fault_t> Rpcs;
00086 };
00087
00098 class L4_EXPORT Pager :
00099   public Kobject_t<Pager, Io_pager, L4_PROTO_PAGE_FAULT>
00100 {
00101 public:
00134   L4_INLINE_RPC(
00135       l4_msgtag_t, page_fault, (l4_umword_t pfa, l4_umword_t pc,
00136                                 L4::Ipc::Rcv_fpage rwin,
00137                                 L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00138
00139   typedef L4::Typeid::Rpc_nocode<page_fault_t> Rpcs;
00140 };
00141
00142 }
```

16.355 l4/sys/platform_control File Reference

Platform control object.

```
#include <l4/sys/capability>
#include <l4/sys/platform_control.h>
```

`#include <l4/sys/cxx/ipc_iface>`
Include dependency graph for platform_control:

Data Structures

- class L4::Platform_control

  L4 C++ interface for controlling platform-wide properties, see Platform Control C API for the C interface.

Namespaces

- L4

  L4 low-level kernel interface.

16.355.1 Detailed Description

Platform control object.

Definition in file platform_control.

16.356 platform_control

```
// vi:set ft=cpp: -*- Mode: C++ -*-
/*
 * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
 *          Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *          Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */

#pragma once

#include <l4/sys/capability>
#include <l4/sys/platform_control.h>
#include <l4/sys/cxx/ipc_iface>

namespace L4 {

class L4_EXPORT Platform_control
: public Kobject_t<Platform_control, Kobject, L4_PROTO_PLATFORM_CTL>
{
public:
  L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SUSPEND_OP,
                   l4_msgtag_t, system_suspend, (l4_umword_t extras));

  L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
                   l4_msgtag_t, system_shutdown, (l4_umword_t reboot));

  L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP,
                   l4_msgtag_t, cpu_allow_shutdown,
                   (l4_umword_t phys_id, l4_umword_t enable));

  L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ENABLE_OP,
                   l4_msgtag_t, cpu_enable, (l4_umword_t phys_id));

  L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_DISABLE_OP,
                   l4_msgtag_t, cpu_disable, (l4_umword_t phys_id));

  typedef L4::Typeid::Rpcs_sys<system_suspend_t, system_shutdown_t,
                               cpu_allow_shutdown_t, cpu_enable_t,
                               cpu_disable_t> Rpcs;
};

}

```

16.357 l4/sys/platform_control.h File Reference

Platform control object.

```
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for platform_control.h:

This graph shows which files directly or indirectly include this file:

Enumerations

- enum L4_platform_ctl_ops {
  L4_PLATFORM_CTL_SYS_SUSPEND_OP = 0UL , L4_PLATFORM_CTL_SYS_SHUTDOWN_OP = 1UL , L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP = 2UL , L4_PLATFORM_CTL_CPU_ENABLE_OP = 3UL ,
  L4_PLATFORM_CTL_CPU_DISABLE_OP = 4UL }

  Operations on platform-control objects.
- enum L4_platform_ctl_proto { L4_PROTO_PLATFORM_CTL = 0 }

  Predefined protocol type for messages to platform-control objects.

Functions

- l4_msgtag_t l4_platform_ctl_system_suspend (l4_cap_idx_t pfc, l4_umword_t extras) L4_NOTHROW

  Enter suspend to RAM.
- l4_msgtag_t l4_platform_ctl_system_shutdown (l4_cap_idx_t pfc, l4_umword_t reboot) L4_NOTHROW

  Shutdown or reboot the system.
- l4_msgtag_t l4_platform_ctl_cpu_allow_shutdown (l4_cap_idx_t pfc, l4_umword_t phys_id, l4_umword_t enable) L4_NOTHROW

  Allow a CPU to be shut down.
- l4_msgtag_t l4_platform_ctl_cpu_enable (l4_cap_idx_t pfc, l4_umword_t phys_id) L4_NOTHROW

  Enable an offline CPU.
- l4_msgtag_t l4_platform_ctl_cpu_disable (l4_cap_idx_t pfc, l4_umword_t phys_id) L4_NOTHROW

  Disable an online CPU.

16.357.1 Detailed Description

Platform control object.

Definition in file platform_control.h.

16.358 platform_control.h

```
00001
00005 /*
00006  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction.  Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License.  This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021
00022 #pragma once
00023
00024 #include <l4/sys/types.h>
00025 #include <l4/sys/utcb.h>
00026
00056 L4_INLINE l4_msgtag_t
00057 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00058                                l4_umword_t extras) L4_NOTHROW;
00059
00063 L4_INLINE l4_msgtag_t
00064 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00065                                   l4_umword_t extras,
00066                                   l4_utcb_t *utcb) L4_NOTHROW;
00067
00068
00077 L4_INLINE l4_msgtag_t
00078 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00079                                 l4_umword_t reboot) L4_NOTHROW;
00080
00084 L4_INLINE l4_msgtag_t
00085 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00086                                   l4_umword_t reboot,
00087                                   l4_utcb_t *utcb) L4_NOTHROW;
00088
00098 L4_INLINE l4_msgtag_t
00099 l4_platform_ctl_cpu_allow_shutdown(l4_cap_idx_t pfc,
00100                                    l4_umword_t phys_id,
00101                                    l4_umword_t enable) L4_NOTHROW;
00102
```

```
L4_INLINE l4_msgtag_t
l4_platform_ctl_cpu_allow_shutdown_u(l4_cap_idx_t pfc,
                                     l4_umword_t phys_id,
                                     l4_umword_t enable,
                                     l4_utcb_t *utcb) L4_NOTHROW;
L4_INLINE l4_msgtag_t
l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
                           l4_umword_t phys_id) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
                             l4_umword_t phys_id,
                             l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
                            l4_umword_t phys_id) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
                              l4_umword_t phys_id,
                              l4_utcb_t *utcb) L4_NOTHROW;
 /* ends l4_platform_control_api group */


enum L4_platform_ctl_ops
{
  L4_PLATFORM_CTL_SYS_SUSPEND_OP        = 0UL,
  L4_PLATFORM_CTL_SYS_SHUTDOWN_OP       = 1UL,
  L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP = 2UL,
  L4_PLATFORM_CTL_CPU_ENABLE_OP         = 3UL,
  L4_PLATFORM_CTL_CPU_DISABLE_OP        = 4UL,
};

enum L4_platform_ctl_proto
{
  L4_PROTO_PLATFORM_CTL = 0
};

/* IMPLEMENTATION ----------------------------------------------------------*/

#include <l4/sys/ipc.h>

L4_INLINE l4_msgtag_t
l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
                                 l4_umword_t extras,
                                 l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[0] = L4_PLATFORM_CTL_SYS_SUSPEND_OP;
  v->mr[1] = extras;
  return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
                                          L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
                                  l4_umword_t reboot,
                                  l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[0] = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP;
  v->mr[1] = reboot;
  return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
                                          L4_IPC_NEVER);
}


L4_INLINE l4_msgtag_t
l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
                               l4_umword_t extras) L4_NOTHROW
{
  return l4_platform_ctl_system_suspend_u(pfc, extras, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
                                l4_umword_t reboot) L4_NOTHROW
{
  return l4_platform_ctl_system_shutdown_u(pfc, reboot, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_platform_ctl_cpu_allow_shutdown_u(l4_cap_idx_t pfc,
                                     l4_umword_t phys_id,
                                     l4_umword_t enable,
                                     l4_utcb_t *utcb) L4_NOTHROW
```

```
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[0] = L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP;
  v->mr[1] = phys_id;
  v->mr[2] = enable;
  return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 3, 0, 0),
                                          L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_platform_ctl_cpu_allow_shutdown(l4_cap_idx_t pfc,
                                   l4_umword_t phys_id,
                                   l4_umword_t enable) L4_NOTHROW
{
  return l4_platform_ctl_cpu_allow_shutdown_u(pfc, phys_id, enable, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
                             l4_umword_t phys_id,
                             l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[0] = L4_PLATFORM_CTL_CPU_ENABLE_OP;
  v->mr[1] = phys_id;
  return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
                                          L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
                              l4_umword_t phys_id,
                              l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[0] = L4_PLATFORM_CTL_CPU_DISABLE_OP;
  v->mr[1] = phys_id;
  return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
                                          L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
                           l4_umword_t phys_id) L4_NOTHROW
{
  return l4_platform_ctl_cpu_enable_u(pfc, phys_id, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
                            l4_umword_t phys_id) L4_NOTHROW
{
  return l4_platform_ctl_cpu_disable_u(pfc, phys_id, l4_utcb());
}
```

16.359 l4/sys/rcv_endpoint File Reference

The C++ Receive endpoint interface.

```
#include <l4/sys/rcv_endpoint.h>
#include <l4/sys/types.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for rcv_endpoint:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Rcv_endpoint

  Interface for kernel objects that allow to receive IPC from them.

Namespaces

- L4

  L4 low-level kernel interface.

16.359.1 Detailed Description

The C++ Receive endpoint interface.

Definition in file rcv_endpoint.

16.360 rcv_endpoint

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #pragma once
00023
00024 #include <l4/sys/rcv_endpoint.h>
00025 #include <l4/sys/types.h>
00026 #include <l4/sys/capability>
00027 #include <l4/sys/cxx/ipc_iface>
00028
00029 namespace L4 {
00030
00031 class Thread;
00032
00040 class L4_EXPORT Rcv_endpoint :
00041   public Kobject_t<Rcv_endpoint, Kobject, L4_PROTO_KOBJECT,
00042                    Type_info::Demand_t<1> >
00043 {
00044 public:
00066   L4_INLINE_RPC_OP(L4_RCV_EP_BIND_OP,
00067       l4_msgtag_t, bind_thread, (Ipc::Cap<Thread> t, l4_umword_t label));
00068
00069   typedef L4::Typeid::Rpcs_sys<bind_thread_t> Rpcs;
00070 };
00071
00072 }
```

16.361 l4/sys/rcv_endpoint.h File Reference

Receive endpoint C interface.

```
#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for rcv_endpoint.h:

This graph shows which files directly or indirectly include this file:

Enumerations

- enum L4_rcv_ep_ops { L4_RCV_EP_BIND_OP = 0x10 }

  Receive endpoint operations.

Functions

- l4_msgtag_t l4_rcv_ep_bind_thread (l4_cap_idx_t ep, l4_cap_idx_t thread, l4_umword_t label)

  Bind the IPC receive endpoint to a thread.

16.361.1 Detailed Description

Receive endpoint C interface.

Definition in file rcv_endpoint.h.

16.361.2 Enumeration Type Documentation

16.361.2.1 L4_rcv_ep_ops

```
enum L4_rcv_ep_ops
```

Receive endpoint operations.

Enumerator

L4_RCV_EP_BIND_OP	Bind operation.

Definition at line 62 of file rcv_endpoint.h.

16.362 rcv_endpoint.h

```
00001
00005 /*
00006  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction.  Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License.  This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021 #pragma once
00022
00023 #include <l4/sys/utcb.h>
00024 #include <l4/sys/types.h>
00025
00049 L4_INLINE l4_msgtag_t
00050 l4_rcv_ep_bind_thread(l4_cap_idx_t ep, l4_cap_idx_t thread,
00051                       l4_umword_t label);
00052
00057 L4_INLINE l4_msgtag_t
00058 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep, l4_cap_idx_t thread,
00059                         l4_umword_t label, l4_utcb_t *utcb);
00060
00062 enum L4_rcv_ep_ops
00063 {
00064   L4_RCV_EP_BIND_OP     = 0x10,
00065 };
00066
00067 /* IMPLEMENTATION ----------------------------------------------------------*/
00068
00069 #include <l4/sys/ipc.h>
00070
00071 L4_INLINE l4_msgtag_t
00072 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep,
00073                         l4_cap_idx_t thread, l4_umword_t label,
00074                         l4_utcb_t *utcb)
00075 {
00076   l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00077   m->mr[0] = L4_RCV_EP_BIND_OP;
00078   m->mr[1] = label;
00079   m->mr[2] = l4_map_obj_control(0, 0);
00080   m->mr[3] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00081   return l4_ipc_call(ep, utcb, l4_msgtag(L4_PROTO_KOBJECT, 2, 1, 0),
00082                      L4_IPC_NEVER);
00083 }
```

```
L4_INLINE l4_msgtag_t
l4_rcv_ep_bind_thread(l4_cap_idx_t ep, l4_cap_idx_t thread,
                      l4_umword_t label)
{
  return l4_rcv_ep_bind_thread_u(ep, thread, label, l4_utcb());
}
```

16.363 l4/sys/scheduler File Reference

Scheduler object functions.

```
#include <l4/sys/icu>
#include <l4/sys/scheduler.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for scheduler:

Data Structures

- class L4::Scheduler

  C++ interface of the Scheduler kernel object, see Scheduler for the C interface.

Namespaces

- L4

  L4 low-level kernel interface.

16.363.1 Detailed Description

Scheduler object functions.

Definition in file scheduler.

16.364 scheduler

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/icu>
00027 #include <l4/sys/scheduler.h>
00028 #include <l4/sys/capability>
00029 #include <l4/sys/cxx/ipc_iface>
00030
00031 namespace L4 {
00032
00057 class L4_EXPORT Scheduler :
00058   public Kobject_t<Scheduler, Icu, L4_PROTO_SCHEDULER,
00059                    Type_info::Demand_t<1> >
00060 {
00061 public:
00062   // ABI function for 'info' call
00063   L4_INLINE_RPC_NF_OP(L4_SCHEDULER_INFO_OP,
00064       l4_msgtag_t, info, (l4_umword_t gran_offset, l4_umword_t *map,
00065                           l4_umword_t *cpu_max, l4_umword_t *sched_classes));
00066
00086   l4_msgtag_t info(l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus,
00087                    l4_umword_t *sched_classes = nullptr,
00088                    l4_utcb_t *utcb = l4_utcb()) const noexcept
00089   {
00090     l4_umword_t max = 0;
00091     l4_umword_t sc = 0;
00092     l4_msgtag_t t =
00093       info_t::call(c(), cpus->gran_offset, &cpus->map, &max, &sc, utcb);
00094     if (cpu_max) *cpu_max = max;
00095     if (sched_classes) *sched_classes = sc;
00096     return t;
00097   }
00098
00120   L4_INLINE_RPC_OP(L4_SCHEDULER_RUN_THREAD_OP,
00121       l4_msgtag_t, run_thread, (Ipc::Cap<Thread> thread, l4_sched_param_t const &sp));
00122
00149   L4_INLINE_RPC_OP(L4_SCHEDULER_IDLE_TIME_OP,
00150       l4_msgtag_t, idle_time, (l4_sched_cpu_set_t const &cpus,
00151                                l4_kernel_clock_t *us));
00152
00162   bool is_online(l4_umword_t cpu, l4_utcb_t *utcb = l4_utcb()) const noexcept
00163   { return l4_scheduler_is_online_u(cap(), cpu, utcb); }
00164
00165   typedef L4::Typeid::Rpcs_sys<info_t, run_thread_t, idle_time_t> Rpcs;
00166 };
00167 }
```

16.365 l4/sys/scheduler.h File Reference

Scheduler object functions.

```
#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for scheduler.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct l4_sched_cpu_set_t

  CPU sets.
- struct l4_sched_param_t

  Scheduler parameter set.

Typedefs

- typedef struct l4_sched_cpu_set_t l4_sched_cpu_set_t

  CPU sets.
- typedef struct l4_sched_param_t l4_sched_param_t

  Scheduler parameter set.

Enumerations

- enum L4_scheduler_classes { L4_SCHEDULER_CLASS_FIXED_PRIO = 1UL << 1 , L4_SCHEDULER_CLASS_WFQ = 1UL << 2 }

  Supported scheduler classes.
- enum L4_scheduler_ops { L4_SCHEDULER_INFO_OP = 0UL , L4_SCHEDULER_RUN_THREAD_OP = 1UL , L4_SCHEDULER_IDLE_TIME_OP = 2UL }

  Operations on the Scheduler object.

Functions

- l4_sched_cpu_set_t l4_sched_cpu_set (l4_umword_t offset, unsigned char granularity, l4_umword_t map=1) L4_NOTHROW
- l4_msgtag_t l4_scheduler_info (l4_cap_idx_t scheduler, l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus) L4_NOTHROW

  Get scheduler information.
- l4_sched_param_t l4_sched_param (unsigned prio, l4_umword_t quantum=0) L4_NOTHROW

  Construct scheduler parameter.
- l4_msgtag_t l4_scheduler_run_thread (l4_cap_idx_t scheduler, l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW

  Run a thread on a Scheduler.
- l4_msgtag_t l4_scheduler_idle_time (l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus, l4_kernel_clock_t *us) L4_NOTHROW

  Query the idle time (in µs) of a CPU.
- int l4_scheduler_is_online (l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW

  Query if a CPU is online.

16.365.1 Detailed Description

Scheduler object functions.

Definition in file scheduler.h.

16.366 scheduler.h

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>

enum L4_scheduler_classes
{
  L4_SCHEDULER_CLASS_FIXED_PRIO = 1UL « 1,
  L4_SCHEDULER_CLASS_WFQ        = 1UL « 2,
};

typedef struct l4_sched_cpu_set_t
{
  l4_umword_t gran_offset;

  l4_umword_t map;

#ifdef __cplusplus
  unsigned char granularity() const { return gran_offset » 24; }
  unsigned offset() const { return gran_offset & 0x00ffffff; }
  void set(unsigned char granularity, unsigned offset)
  { gran_offset = ((l4_umword_t)granularity « 24) | (offset & 0x00ffffff); }
#endif
} l4_sched_cpu_set_t;

L4_INLINE l4_sched_cpu_set_t
l4_sched_cpu_set(l4_umword_t offset, unsigned char granularity,
                 l4_umword_t map L4_DEFAULT_PARAM(1)) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_scheduler_info(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
                  l4_sched_cpu_set_t *cpus) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
                    l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes,
                    l4_utcb_t *utcb) L4_NOTHROW;


typedef struct l4_sched_param_t
{
  l4_sched_cpu_set_t affinity;
  l4_umword_t        prio;
  l4_umword_t        quantum;
} l4_sched_param_t;

L4_INLINE l4_sched_param_t
l4_sched_param(unsigned prio,
               l4_umword_t quantum L4_DEFAULT_PARAM(0)) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_scheduler_run_thread(l4_cap_idx_t scheduler,
                        l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
                          l4_sched_param_t const *sp, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_scheduler_idle_time(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
                       l4_kernel_clock_t *us) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
                         l4_kernel_clock_t *us, l4_utcb_t *utcb) L4_NOTHROW;

```

```
L4_INLINE int
l4_scheduler_is_online(l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW;

L4_INLINE int
l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
                         l4_utcb_t *utcb) L4_NOTHROW;



enum L4_scheduler_ops
{
  L4_SCHEDULER_INFO_OP       = 0UL,
  L4_SCHEDULER_RUN_THREAD_OP = 1UL,
  L4_SCHEDULER_IDLE_TIME_OP  = 2UL,
};

/*************** Implementations ******************/

L4_INLINE l4_sched_cpu_set_t
l4_sched_cpu_set(l4_umword_t offset, unsigned char granularity,
                 l4_umword_t map) L4_NOTHROW
{
  l4_sched_cpu_set_t cs;
  cs.gran_offset = ((l4_umword_t)granularity << 24) | offset;
  cs.map         = map;
  return cs;
}

L4_INLINE l4_sched_param_t
l4_sched_param(unsigned prio, l4_umword_t quantum) L4_NOTHROW
{
  l4_sched_param_t sp;
  sp.prio     = prio;
  sp.quantum  = quantum;
  sp.affinity = l4_sched_cpu_set(0, ~0, 1);
  return sp;
}


L4_INLINE l4_msgtag_t
l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
                    l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes,
                    l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
  l4_msgtag_t res;

  m->mr[0] = L4_SCHEDULER_INFO_OP;
  m->mr[1] = cpus->gran_offset;

  res = l4_ipc_call(scheduler, utcb, l4_msgtag(L4_PROTO_SCHEDULER, 2, 0, 0), L4_IPC_NEVER);

  if (l4_msgtag_has_error(res))
    return res;

  cpus->map = m->mr[0];

  if (cpu_max)
    *cpu_max = m->mr[1];

  if (sched_classes)
    *sched_classes = m->mr[2];

  return res;
}

L4_INLINE l4_msgtag_t
l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
                          l4_sched_param_t const *sp, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
  m->mr[0] = L4_SCHEDULER_RUN_THREAD_OP;
  m->mr[1] = sp->affinity.gran_offset;
  m->mr[2] = sp->affinity.map;
  m->mr[3] = sp->prio;
  m->mr[4] = sp->quantum;
  m->mr[5] = l4_map_obj_control(0, 0);
  m->mr[6] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;

  return l4_ipc_call(scheduler, utcb, l4_msgtag(L4_PROTO_SCHEDULER, 5, 1, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
                         l4_kernel_clock_t *us, l4_utcb_t *utcb) L4_NOTHROW
```

```
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  l4_msgtag_t res;

  v->mr[0] = L4_SCHEDULER_IDLE_TIME_OP;
  v->mr[1] = cpus->gran_offset;
  v->mr[2] = cpus->map;

  res = l4_ipc_call(scheduler, utcb,
                    l4_msgtag(L4_PROTO_SCHEDULER, 3, 0, 0), L4_IPC_NEVER);

  if (l4_msgtag_has_error(res))
    return res;

  *us = v->mr64[l4_utcb_mr64_idx(0)];

  return res;
}


L4_INLINE int
l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
                         l4_utcb_t *utcb) L4_NOTHROW
{
  l4_sched_cpu_set_t s;
  l4_msgtag_t r;
  s.gran_offset = cpu;
  r = l4_scheduler_info_u(scheduler, NULL, &s, NULL, utcb);
  if (l4_msgtag_has_error(r) || l4_msgtag_label(r) < 0)
    return 0;

  return s.map & 1;
}


L4_INLINE l4_msgtag_t
l4_scheduler_info(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
                  l4_sched_cpu_set_t *cpus) L4_NOTHROW
{
  return l4_scheduler_info_u(scheduler, cpu_max, cpus, NULL, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_scheduler_info_with_classes(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
                               l4_sched_cpu_set_t *cpus,
                               l4_umword_t *sched_classes) L4_NOTHROW
{
  return l4_scheduler_info_u(scheduler, cpu_max, cpus, sched_classes, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_scheduler_run_thread(l4_cap_idx_t scheduler,
                        l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW
{
  return l4_scheduler_run_thread_u(scheduler, thread, sp, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_scheduler_idle_time(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
                       l4_kernel_clock_t *us) L4_NOTHROW
{
  return l4_scheduler_idle_time_u(scheduler, cpus, us, l4_utcb());
}

L4_INLINE int
l4_scheduler_is_online(l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW
{
  return l4_scheduler_is_online_u(scheduler, cpu, l4_utcb());
}
```

16.367 l4/sys/semaphore File Reference

Semaphore class definition.

```
#include <l4/sys/irq>
#include <l4/sys/semaphore.h>
```

Include dependency graph for semaphore:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct L4::Semaphore

  C++ Kernel-provided semaphore interface, see Kernel-provided semaphore for the C interface.

Namespaces

- L4

  L4 low-level kernel interface.

16.367.1 Detailed Description

Semaphore class definition.

Definition in file semaphore.

16.368 semaphore

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/sys/irq>
00026 #include <l4/sys/semaphore.h>
00027
00028 namespace L4 {
00029
00052 struct Semaphore : Kobject_t<Semaphore, Triggerable, L4_PROTO_SEMAPHORE>
00053 {
00066   l4_msgtag_t up(l4_utcb_t *utcb = l4_utcb()) noexcept
00067   { return trigger(utcb); }
00068
00086   l4_msgtag_t down(l4_timeout_t timeout = L4_IPC_NEVER,
00087                    l4_utcb_t *utcb = l4_utcb()) noexcept
00088   { return l4_semaphore_down_u(cap(), timeout, utcb);  }
00089 };
00090
00091 }
```

16.369 l4/sys/semaphore.h File Reference

C semaphore interface.

```
#include <l4/sys/irq.h>
```

Include dependency graph for semaphore.h:

This graph shows which files directly or indirectly include this file:

Functions

- l4_msgtag_t l4_semaphore_up (l4_cap_idx_t sem) L4_NOTHROW

  Semaphore up operation (wrapper for trigger()).
- l4_msgtag_t l4_semaphore_down (l4_cap_idx_t sem, l4_timeout_t timeout) L4_NOTHROW

  Semaphore down operation.

16.369.1 Detailed Description

C semaphore interface.

Definition in file semaphore.h.

16.370 semaphore.h

```
00001
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction.  Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License.  This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022
00023 #pragma once
00024
00025 #include <l4/sys/irq.h>
00026
00036 enum L4_semaphore_op
00037 {
00038   L4_SEMAPHORE_OP_DOWN    = 0,
00039   // semaphore up is IRQ_OP_TRIGGER with IRQ/Triggerable protocol
00040 };
00041
00047 L4_INLINE l4_msgtag_t
00048 l4_semaphore_up(l4_cap_idx_t sem) L4_NOTHROW
00049 {
00050   return l4_irq_trigger(sem);
00051 }
00052
00056 L4_INLINE l4_msgtag_t
00057 l4_semaphore_up_u(l4_cap_idx_t sem, l4_utcb_t *utcb) L4_NOTHROW
00058 {
00059   return l4_irq_trigger_u(sem, utcb);
00060 }
00061
00067 L4_INLINE l4_msgtag_t
00068 l4_semaphore_down(l4_cap_idx_t sem, l4_timeout_t timeout) L4_NOTHROW;
00069
00073 L4_INLINE l4_msgtag_t
00074 l4_semaphore_down_u(l4_cap_idx_t sem, l4_timeout_t to,
00075                     l4_utcb_t *utcb) L4_NOTHROW;
00076
00077
00078 L4_INLINE l4_msgtag_t
00079 l4_semaphore_down_u(l4_cap_idx_t sem, l4_timeout_t to,
00080                     l4_utcb_t *utcb) L4_NOTHROW
00081 {
00082   l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00083   m->mr[0] = L4_SEMAPHORE_OP_DOWN;
00084   return l4_ipc_call(sem, utcb, l4_msgtag(L4_PROTO_SEMAPHORE, 1, 0, 0), to);
00085 }
00086
00087
00088 L4_INLINE l4_msgtag_t
00089 l4_semaphore_down(l4_cap_idx_t sem, l4_timeout_t to) L4_NOTHROW
00090 {
00091   return l4_semaphore_down_u(sem, to, l4_utcb());
00092 }
00093
```

16.371 l4/sys/smart_capability File Reference

L4::Capability class.

`#include <l4/sys/capability>`
Include dependency graph for smart_capability:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Smart_cap< T, SMART >

  Smart capability class.

Namespaces

- L4

  L4 low-level kernel interface.

Functions

- template<typename T , typename F , typename SMART >
  Smart_cap< T, SMART > L4::cap_cast (Smart_cap< F, SMART > const &c) noexcept

  `static_cast` for (smart) capabilities.
- template<typename T , typename F , typename SMART >
  Smart_cap< T, SMART > L4::cap_reinterpret_cast (Smart_cap< F, SMART > const &c) noexcept

  `reinterpret_cast` for (smart) capabilities.

16.371.1 Detailed Description

L4::Capability class.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file smart_capability.

16.372 smart_capability

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00009 /*
00010  * (c) 2008-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction.  Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License.  This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029
00030 namespace L4 {
00031
00035 template< typename T, typename SMART >
00036 class Smart_cap : public Cap_base, private SMART
00037 {
00038 public:
00039
00040   SMART const &smart() const noexcept { return *this; }
00041
00042   void _delete() noexcept
00043   {
00044     SMART::free(const_cast<Smart_cap<T,SMART>&>(*this));
00045   }
00046
00047   Cap<T> release() const noexcept
00048   {
00049     l4_cap_idx_t r = cap();
```

```
    SMART::invalidate(const_cast<Smart_cap<T,SMART>&>(*this));

    return Cap<T>(r);
  }

  void reset() noexcept
  {
    _c = L4_INVALID_CAP;
  }

  Smart_cap() noexcept : Cap_base(Invalid) {}

  Smart_cap(Cap_base::Cap_type t) noexcept : Cap_base(t) {}

  template< typename O >
  Smart_cap(Cap<O> const &p) noexcept : Cap_base(p.cap())
  { T* __t = ((O*)100); (void)__t; }

  template< typename O >
  Smart_cap(Cap<O> const &p, SMART const &smart) noexcept
  : Cap_base(p.cap()), SMART(smart)
  { T* __t = ((O*)100); (void)__t; }

  template< typename O >
  Smart_cap(Smart_cap<O, SMART> const &o) noexcept
  : Cap_base(SMART::copy(o)), SMART(o.smart())
  { T* __t = ((O*)100); (void)__t; }

  Smart_cap(Smart_cap const &o) noexcept
  : Cap_base(SMART::copy(o)), SMART(o.smart())
  { }

  template< typename O >
  Smart_cap(typename Cap<O>::Cap_type cap) noexcept : Cap_base(cap)
  { T* __t = ((O*)100); (void)__t; }

  void operator = (typename Cap<T>::Cap_type cap) noexcept
  {
    _delete();
    _c = cap;
  }

  template< typename O >
  void operator = (Smart_cap<O, SMART> const &o) noexcept
  {
    _delete();
    _c = this->SMART::copy(o).cap();
    this->SMART::operator = (o.smart());
    // return *this;
  }

  Smart_cap const &operator = (Smart_cap const &o) noexcept
  {
    if (&o == this)
      return *this;

    _delete();
    _c = this->SMART::copy(o).cap();
    this->SMART::operator = (o.smart());
    return *this;
  }

#if __cplusplus >= 201103L
  template< typename O >
  Smart_cap(Smart_cap<O, SMART> &&o) noexcept
  : Cap_base(o.release()), SMART(o.smart())
  { T* __t = ((O*)100); (void)__t; }

  Smart_cap(Smart_cap &&o) noexcept
  : Cap_base(o.release()), SMART(o.smart())
  { }

  template< typename O >
  void operator = (Smart_cap<O, SMART> &&o) noexcept
  {
    _delete();
    _c = o.release().cap();
    this->SMART::operator = (o.smart());
    // return *this;
  }

  Smart_cap const &operator = (Smart_cap &&o) noexcept
  {
    if (&o == this)
      return *this;

    _delete();
```

```
    _c = o.release().cap();
    this->SMART::operator = (o.smart());
    return *this;
  }
#endif

  Cap<T> operator -> () const noexcept { return Cap<T>(_c); }

  Cap<T> get() const noexcept { return Cap<T>(_c); }

  ~Smart_cap() noexcept { _delete(); }
};

template< typename T >
class Weak_cap : public Cap_base
{
public:
  Weak_cap() noexcept : Cap_base(Invalid) {}

  template< typename O >
  Weak_cap(typename Cap<O>::Cap_type t) noexcept : Cap_base(t)
  { T* __t = ((O*)100); (void)__t; }

  template< typename O, typename S >
  Weak_cap(Smart_cap<O, S> const &c) noexcept : Cap_base(c.cap())
  { T* __t = ((O*)100); (void)__t; }

  Weak_cap(Weak_cap const &o) noexcept : Cap_base(o) {}

  template< typename O >
  Weak_cap(Weak_cap<O> const &o) noexcept : Cap_base(o)
  { T* __t = ((O*)100); (void)__t; }

};

namespace Cap_traits {
  template< typename T1, typename T2 >
  struct Type { enum { Equal = false }; };

  template< typename T1 >
  struct Type<T1,T1> { enum { Equal = true }; };
};

template< typename T, typename F, typename SMART >
inline
Smart_cap<T, SMART> cap_cast(Smart_cap<F, SMART> const &c) noexcept
{
   (void)static_cast<T const *>(reinterpret_cast<F const *>(100));
  return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
}


template< typename T, typename F, typename SMART >
inline
Smart_cap<T, SMART> cap_reinterpret_cast(Smart_cap<F, SMART> const &c) noexcept
{
  return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
}


}

```

16.373 l4/sys/task File Reference

Common task related definitions.

```
#include <l4/sys/task.h>
#include <l4/sys/capability>
```

Include dependency graph for task:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Task

  C++ interface of the Task kernel object, see Task for the C interface.

Namespaces

- L4

  L4 low-level kernel interface.

16.373.1 Detailed Description

Common task related definitions.

Definition in file task.

16.374 task

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task.h>
00028 #include <l4/sys/capability>
00029
00030 namespace L4 {
00031
00044 class Task :
00045   public Kobject_t<Task, Kobject, L4_PROTO_TASK,
00046                    Type_info::Demand_t<2> >
00047 {
00048 public:
00075   l4_msgtag_t map(Cap<Task> const &src_task,
00076                   l4_fpage_t const &snd_fpage, l4_umword_t snd_base,
00077                   l4_utcb_t *utcb = l4_utcb()) noexcept
00078   { return l4_task_map_u(cap(), src_task.cap(), snd_fpage, snd_base, utcb); }
00079
00102   l4_msgtag_t unmap(l4_fpage_t const &fpage,
00103                     l4_umword_t map_mask,
00104                     l4_utcb_t *utcb = l4_utcb()) noexcept
00105   { return l4_task_unmap_u(cap(), fpage, map_mask, utcb); }
00106
00121   l4_msgtag_t unmap_batch(l4_fpage_t const *fpages,
00122                           unsigned num_fpages,
00123                           l4_umword_t map_mask,
00124                           l4_utcb_t *utcb = l4_utcb()) noexcept
00125   { return l4_task_unmap_batch_u(cap(), fpages, num_fpages, map_mask, utcb); }
00126
00141   l4_msgtag_t delete_obj(L4::Cap<void> obj,
00142                          l4_utcb_t *utcb = l4_utcb()) noexcept
00143   { return l4_task_delete_obj_u(cap(), obj.cap(), utcb); }
00144
00160   l4_msgtag_t release_cap(L4::Cap<void> cap,
00161                           l4_utcb_t *utcb = l4_utcb()) noexcept
00162   { return l4_task_release_cap_u(this->cap(), cap.cap(), utcb); }
00163
00180   l4_msgtag_t cap_valid(Cap<void> const &cap,
00181                         l4_utcb_t *utcb = l4_utcb()) noexcept
00182   { return l4_task_cap_valid_u(this->cap(), cap.cap(), utcb); }
00183
00196   l4_msgtag_t cap_equal(Cap<void> const &cap_a,
00197                         Cap<void> const &cap_b,
00198                         l4_utcb_t *utcb = l4_utcb()) noexcept
00199   { return l4_task_cap_equal_u(cap(), cap_a.cap(), cap_b.cap(), utcb); }
00200
00215   l4_msgtag_t add_ku_mem(l4_fpage_t const &fpage,
00216                          l4_utcb_t *utcb = l4_utcb()) noexcept
00217   { return l4_task_add_ku_mem_u(cap(), fpage, utcb); }
00218
00219 };
00220 }
00221
00222
```

16.375 l4/sys/task.h File Reference

Common task related definitions.

```
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for task.h:

This graph shows which files directly or indirectly include this file:

Enumerations

- enum L4_task_ops {
  L4_TASK_MAP_OP = 0UL , L4_TASK_UNMAP_OP = 1UL , L4_TASK_CAP_INFO_OP = 2UL , L4_TASK_ADD_KU_MEM_OP = 3UL ,
  L4_TASK_LDT_SET_X86_OP = 0x11UL , L4_TASK_MAP_VGICC_ARM_OP = 0x12UL }

  Operations on task objects.

Functions

- l4_msgtag_t l4_task_map (l4_cap_idx_t dst_task, l4_cap_idx_t src_task, l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW

  Map resources available in the source task to a destination task.

- l4_msgtag_t l4_task_unmap (l4_cap_idx_t task, l4_fpage_t fpage, l4_umword_t map_mask) L4_NOTHROW

  Revoke rights from the task.
- l4_msgtag_t l4_task_unmap_batch (l4_cap_idx_t task, l4_fpage_t const *fpages, unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW

  Revoke rights from a task.
- l4_msgtag_t l4_task_delete_obj (l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW

  Release capability and delete object.
- l4_msgtag_t l4_task_release_cap (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW

  Release object capability.
- l4_msgtag_t l4_task_cap_valid (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW

  Check whether a capability is present (refers to an object).
- l4_msgtag_t l4_task_cap_equal (l4_cap_idx_t task, l4_cap_idx_t cap_a, l4_cap_idx_t cap_b) L4_NOTHROW

  Test whether two capabilities point to the same object with the same rights.
- l4_msgtag_t l4_task_add_ku_mem (l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW

  Add kernel-user memory.

16.375.1 Detailed Description

Common task related definitions.

Definition in file task.h.

16.376 task.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/utcb.h>
00028
00068 L4_INLINE l4_msgtag_t
00069 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00070             l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW;
00071
00075 L4_INLINE l4_msgtag_t
00076 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00077               l4_fpage_t snd_fpage, l4_umword_t snd_base, l4_utcb_t *utcb) L4_NOTHROW;
00078
00101 L4_INLINE l4_msgtag_t
00102 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00103               l4_umword_t map_mask) L4_NOTHROW;
00104
00108 L4_INLINE l4_msgtag_t
00109 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00110                 l4_umword_t map_mask, l4_utcb_t *utcb) L4_NOTHROW;
00111
00131 L4_INLINE l4_msgtag_t
00132 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
```

```
                     unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
                      unsigned num_fpages, l4_umword_t map_mask,
                      l4_utcb_t *u) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
                     l4_utcb_t *u) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_task_release_cap(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
                      l4_utcb_t *u) L4_NOTHROW;


L4_INLINE l4_msgtag_t
l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
                  l4_cap_idx_t cap_b) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t ku_mem,
                     l4_utcb_t *u) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW;


L4_INLINE l4_msgtag_t
l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
                    l4_cap_idx_t cap_b, l4_utcb_t *utcb) L4_NOTHROW;

enum L4_task_ops
{
  L4_TASK_MAP_OP           = 0UL,
  L4_TASK_UNMAP_OP         = 1UL,
  L4_TASK_CAP_INFO_OP      = 2UL,
  L4_TASK_ADD_KU_MEM_OP    = 3UL,
  L4_TASK_LDT_SET_X86_OP   = 0x11UL,
  L4_TASK_MAP_VGICC_ARM_OP = 0x12UL,
};


/* IMPLEMENTATION -----------------------------------------------------------*/

#include <l4/sys/ipc.h>


L4_INLINE l4_msgtag_t
l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
              l4_fpage_t snd_fpage, l4_umword_t snd_base, l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(u);
  v->mr[0] = L4_TASK_MAP_OP;
  v->mr[3] = l4_map_obj_control(0,0);
  v->mr[4] = l4_obj_fpage(src_task, 0, L4_CAP_FPAGE_RWS).raw;
  v->mr[1] = snd_base;
  v->mr[2] = snd_fpage.raw;
  return l4_ipc_call(dst_task, u, l4_msgtag(L4_PROTO_TASK, 3, 1, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
                l4_umword_t map_mask, l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(u);
  v->mr[0] = L4_TASK_UNMAP_OP;
  v->mr[1] = map_mask;
  v->mr[2] = fpage.raw;
  return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
                      unsigned num_fpages, l4_umword_t map_mask,
```

```
                        l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(u);
  v->mr[0] = L4_TASK_UNMAP_OP;
  v->mr[1] = map_mask;
  __builtin_memcpy(&v->mr[2], fpages, num_fpages * sizeof(l4_fpage_t));
  return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2 + num_fpages, 0, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap, l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(u);
  v->mr[0] = L4_TASK_CAP_INFO_OP;
  v->mr[1] = cap & ~1UL;
  return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
                    l4_cap_idx_t cap_b, l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(u);
  v->mr[0] = L4_TASK_CAP_INFO_OP;
  v->mr[1] = cap_a;
  v->mr[2] = cap_b;
  return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t ku_mem,
                     l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(u);
  v->mr[0] = L4_TASK_ADD_KU_MEM_OP;
  v->mr[1] = ku_mem.raw;
  return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
}



L4_INLINE l4_msgtag_t
l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
            l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW
{
  return l4_task_map_u(dst_task, src_task, snd_fpage, snd_base, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
              l4_umword_t map_mask) L4_NOTHROW
{
  return l4_task_unmap_u(task, fpage, map_mask, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
                    unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW
{
  return l4_task_unmap_batch_u(task, fpages, num_fpages, map_mask,
                               l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
                     l4_utcb_t *u) L4_NOTHROW
{
  return l4_task_unmap_u(task, l4_obj_fpage(obj, 0, L4_CAP_FPAGE_RWSD),
                         L4_FP_DELETE_OBJ, u);
}

L4_INLINE l4_msgtag_t
l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW
{
  return l4_task_delete_obj_u(task, obj, l4_utcb());
}


L4_INLINE l4_msgtag_t
l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
                      l4_utcb_t *u) L4_NOTHROW
{
  return l4_task_unmap_u(task, l4_obj_fpage(cap, 0, L4_CAP_FPAGE_RWSD),
                         L4_FP_ALL_SPACES, u);
}

L4_INLINE l4_msgtag_t
```

```
00399 l4_task_release_cap(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
00400 {
00401   return l4_task_release_cap_u(task, cap, l4_utcb());
00402 }
00403
00404 L4_INLINE l4_msgtag_t
00405 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
00406 {
00407   return l4_task_cap_valid_u(task, cap, l4_utcb());
00408 }
00409
00410 L4_INLINE l4_msgtag_t
00411 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00412                   l4_cap_idx_t cap_b) L4_NOTHROW
00413 {
00414   return l4_task_cap_equal_u(task, cap_a, cap_b, l4_utcb());
00415 }
00416
00417 L4_INLINE l4_msgtag_t
00418 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW
00419 {
00420   return l4_task_add_ku_mem_u(task, ku_mem, l4_utcb());
00421 }
```

16.377 l4/sys/thread File Reference

Common thread related definitions.

```
#include <l4/sys/capability>
#include <l4/sys/thread.h>
```

Include dependency graph for thread:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Thread

  C++ L4 kernel thread interface, see Thread for the C interface.
- class L4::Thread::Attr

  Thread attributes used for control_commit().
- class L4::Thread::Modify_senders

  Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

Namespaces

- L4

  L4 low-level kernel interface.

16.377.1 Detailed Description

Common thread related definitions.

Definition in file thread.

16.378 thread

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/capability>
00028 #include <l4/sys/thread.h>
00029
00030 namespace L4 {
00031
00058 class Thread :
00059   public Kobject_t<Thread, Kobject, L4_PROTO_THREAD,
00060                    Type_info::Demand_t<1> >
00061 {
00062 public:
00089   l4_msgtag_t ex_regs(l4_addr_t ip, l4_addr_t sp,
00090                       l4_umword_t flags,
00091                       l4_utcb_t *utcb = l4_utcb()) noexcept
00092   { return l4_thread_ex_regs_u(cap(), ip, sp, flags, utcb); }
00093
00122   l4_msgtag_t ex_regs(l4_addr_t *ip, l4_addr_t *sp,
00123                       l4_umword_t *flags,
00124                       l4_utcb_t *utcb = l4_utcb()) noexcept
00125   { return l4_thread_ex_regs_ret_u(cap(), ip, sp, flags, utcb); }
00126
00127
00140   class Attr
00141   {
00142   private:
00143      friend class L4::Thread;
00144      l4_utcb_t *_u;
00145
00146    public:
00153      explicit Attr(l4_utcb_t *utcb = l4_utcb()) noexcept : _u(utcb)
00154      { l4_thread_control_start_u(utcb); }
00155
00163      void pager(Cap<void> const &pager) noexcept
00164      { l4_thread_control_pager_u(pager.cap(), _u); }
00165
00172      Cap<void> pager() noexcept
00173      { return Cap<void>(l4_utcb_mr_u(_u)->mr[1]); }
00174
00182      void exc_handler(Cap<void> const &exc_handler) noexcept
00183      { l4_thread_control_exc_handler_u(exc_handler.cap(), _u); }
00184
00191      Cap<void> exc_handler() noexcept
00192      { return Cap<void>(l4_utcb_mr_u(_u)->mr[2]); }
00193
00215      void bind(l4_utcb_t *thread_utcb, Cap<Task> const &task) noexcept
00216      { l4_thread_control_bind_u(thread_utcb, task.cap(), _u); }
00217
00221      void alien(int on) noexcept
00222      { l4_thread_control_alien_u(_u, on); }
00223
00229      void ux_host_syscall(int on) noexcept
00230      { l4_thread_control_ux_host_syscall_u(_u, on); }
00231
00232    };
00233
00239   l4_msgtag_t control(Attr const &attr) noexcept
00240   { return l4_thread_control_commit_u(cap(), attr._u); }
00241
00249   l4_msgtag_t switch_to(l4_utcb_t *utcb = l4_utcb()) noexcept
00250   { return l4_thread_switch_u(cap(), utcb); }
00251
00260   l4_msgtag_t stats_time(l4_kernel_clock_t *us,
00261                          l4_utcb_t *utcb = l4_utcb()) noexcept
00262   { return l4_thread_stats_time_u(cap(), us, utcb); }
```

```
00263
00271   l4_msgtag_t vcpu_resume_start(l4_utcb_t *utcb = l4_utcb()) noexcept
00272   { return l4_thread_vcpu_resume_start_u(utcb); }
00273
00281   l4_msgtag_t vcpu_resume_commit(l4_msgtag_t tag,
00282                                  l4_utcb_t *utcb = l4_utcb()) noexcept
00283   { return l4_thread_vcpu_resume_commit_u(cap(), tag, utcb); }
00284
00305    l4_msgtag_t vcpu_control(l4_addr_t vcpu_state, l4_utcb_t *utcb = l4_utcb())
00306      noexcept
00307    { return l4_thread_vcpu_control_u(cap(), vcpu_state, utcb); }
00308
00338    l4_msgtag_t vcpu_control_ext(l4_addr_t ext_vcpu_state,
00339                                 l4_utcb_t *utcb = l4_utcb()) noexcept
00340    { return l4_thread_vcpu_control_ext_u(cap(), ext_vcpu_state, utcb); }
00341
00364   l4_msgtag_t register_del_irq(Cap<Irq> irq, l4_utcb_t *u = l4_utcb()) noexcept
00365   { return l4_thread_register_del_irq_u(cap(), irq.cap(), u); }
00366
00385   class Modify_senders
00386   {
00387   private:
00388     friend class Thread;
00389     l4_utcb_t *utcb;
00390     unsigned cnt;
00391
00392   public:
00393     explicit Modify_senders(l4_utcb_t *u = l4_utcb()) noexcept
00394     : utcb(u), cnt(1)
00395     {
00396       l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
00397     }
00398
00418     int add(l4_umword_t match_mask, l4_umword_t match,
00419             l4_umword_t del_bits, l4_umword_t add_bits) noexcept
00420     {
00421       l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00422       if (cnt >= L4_UTCB_GENERIC_DATA_SIZE - 4)
00423         return -L4_ENOMEM;
00424       m->mr[cnt++] = match_mask;
00425       m->mr[cnt++] = match;
00426       m->mr[cnt++] = del_bits;
00427       m->mr[cnt++] = add_bits;
00428       return 0;
00429     }
00430   };
00431
00448   l4_msgtag_t modify_senders(Modify_senders const &todo) noexcept
00449   {
00450     return l4_ipc_call(cap(), todo.utcb, l4_msgtag(L4_PROTO_THREAD, todo.cnt, 0, 0), L4_IPC_NEVER);
00451   }
00452 };
00453 }
```

16.379 l4/cxx/thread File Reference

Thread implementation.

```
#include <l4/sys/capability>
#include <l4/sys/types.h>
```

Include dependency graph for thread:

This graph shows which files directly or indirectly include this file:

Namespaces

- cxx

  Our C++ library.

16.379.1 Detailed Description

Thread implementation.

Definition in file thread.

16.380 thread

```
00001
00005 /*
00006  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012
00013 #ifndef CXX_THREAD_H__
00014 #define CXX_THREAD_H__
00015
00016 #include <l4/sys/capability>
00017 #include <l4/sys/types.h>
00018
00019 namespace cxx {
00020
00021   class Thread
00022   {
00023   public:
00024
00025     enum State
00026     {
00027       Dead    = 0,
00028       Running = 1,
00029       Stopped = 2,
00030     };
00031
00032     Thread(bool initiate);
00033     Thread(void *stack);
00034     Thread(void *stack, L4::Cap<L4::Thread> const &cap);
00035     virtual ~Thread();
00036     void execute() asm ("L4_Thread_execute");
00037     virtual void run() = 0;
00038     virtual void shutdown() asm ("L4_Thread_shutdown");
00039     void start();
00040     void stop();
00041
00042     L4::Cap<L4::Thread> self() const throw()
00043     { return _cap; }
00044
00045     State state() const
00046     { return _state; }
00047
00048     static void start_cxx_thread(Thread *_this)
00049       asm ("L4_Thread_start_cxx_thread");
00050
00051     static void kill_cxx_thread(Thread *_this)
00052       asm ("L4_Thread_kill_cxx_thread");
00053
00054     static void set_pager(L4::Cap<void>const &p) throw()
00055     { _pager = p; }
00056
00057   private:
00058     int create();
00059
00060     L4::Cap<L4::Thread> _cap;
00061     State _state;
00062
00063   protected:
00064     void *_stack;
00065
00066   private:
00067     static L4::Cap<void> _pager;
00068     static L4::Cap<void> _master;
00069   };
00070
00071 };
00072
00073 #endif  /* CXX_THREAD_H__ */
00074
```

16.381 l4/sys/typeinfo_svr File Reference

Type information server template.

```
#include <l4/sys/meta>
#include <l4/sys/cxx/ipc_epiface>
```

Include dependency graph for typeinfo_svr:

This graph shows which files directly or indirectly include this file:

Namespaces

- L4

  L4 low-level kernel interface.

16.381.1 Detailed Description

Type information server template.

Definition in file typeinfo_svr.

16.382 typeinfo_svr

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
```

```
#pragma once

#include <l4/sys/meta>
#include <l4/sys/cxx/ipc_epiface>

namespace L4 { namespace Util {

template<typename KO, typename IOS>
long handle_meta_request(IOS &ios)
{
  using L4::Ipc::Msg::dispatch_call;
  typedef L4::Ipc::Detail::Meta_svr<KO> Msvr;
  l4_msgtag_t tag = dispatch_call<L4::Meta::Rpcs>((Msvr *)0, ios.utcb(),
                                                  ios.tag(), 0);
  ios.set_ipc_params(tag);
  return tag.label();
}

}}
```

16.383 l4/sys/types.h File Reference

Common L4 ABI Data Types.

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>
#include <l4/sys/__l4_fpage.h>
#include <l4/sys/__timeout.h>
```

Include dependency graph for types.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct l4_msgtag_t

  Message tag data structure.

Typedefs

- typedef struct l4_msgtag_t l4_msgtag_t

  Message tag data structure.
- typedef unsigned long l4_cap_idx_t

  L4 Capability selector Type.

Enumerations

- enum l4_msgtag_protocol {
  L4_PROTO_NONE = 0 , L4_PROTO_ALLOW_SYSCALL = 1 , L4_PROTO_PF_EXCEPTION = 1 , L4_PROTO_IRQ = -1L ,
  L4_PROTO_PAGE_FAULT = -2L , L4_PROTO_PREEMPTION = -3L , L4_PROTO_SYS_EXCEPTION = -4L , L4_PROTO_EXCEPTION = -5L ,
  L4_PROTO_SIGMA0 = -6L , L4_PROTO_IO_PAGE_FAULT = -8L , L4_PROTO_KOBJECT = -10L , L4_PROTO_TASK = -11L ,
  L4_PROTO_THREAD = -12L , L4_PROTO_LOG = -13L , L4_PROTO_SCHEDULER = -14L , L4_PROTO_FACTORY = -15L ,
  L4_PROTO_VM = -16L , L4_PROTO_DMA_SPACE = -17L , L4_PROTO_IRQ_SENDER = -18L , L4_PROTO_IRQ_MUX = -19L ,
  L4_PROTO_SEMAPHORE = -20L , L4_PROTO_META = -21L , L4_PROTO_IOMMU = -22L , L4_PROTO_DEBUGGER = -23L ,
  L4_PROTO_SMCCC = -24L }

  Message tag for IPC operations.
- enum l4_msgtag_flags { L4_MSGTAG_ERROR , L4_MSGTAG_TRANSFER_FPU , L4_MSGTAG_SCHEDULE , **L4_MSGTAG_PROPAGATE** = 0x4000 , L4_MSGTAG_FLAGS }

  Flags for message tags.

Functions

- l4_msgtag_t l4_msgtag (long label, unsigned words, unsigned items, unsigned flags) L4_NOTHROW

  Create a message tag from the specified values.
- long l4_msgtag_label (l4_msgtag_t t) L4_NOTHROW

  Get the protocol of tag.
- unsigned l4_msgtag_words (l4_msgtag_t t) L4_NOTHROW

  Get the number of untyped words.
- unsigned l4_msgtag_items (l4_msgtag_t t) L4_NOTHROW

  Get the number of typed items.
- unsigned l4_msgtag_flags (l4_msgtag_t t) L4_NOTHROW

  Get the flags.
- unsigned l4_msgtag_has_error (l4_msgtag_t t) L4_NOTHROW

  Test for error indicator flag.
- unsigned l4_msgtag_is_page_fault (l4_msgtag_t t) L4_NOTHROW

  Test for page-fault protocol.
- unsigned l4_msgtag_is_preemption (l4_msgtag_t t) L4_NOTHROW

  Test for preemption protocol.
- unsigned l4_msgtag_is_sys_exception (l4_msgtag_t t) L4_NOTHROW

  Test for system-exception protocol.
- unsigned l4_msgtag_is_exception (l4_msgtag_t t) L4_NOTHROW

  Test for exception protocol.
- unsigned l4_msgtag_is_sigma0 (l4_msgtag_t t) L4_NOTHROW

Test for sigma0 protocol.

- unsigned l4_msgtag_is_io_page_fault (l4_msgtag_t t) L4_NOTHROW

  Test for IO-page-fault protocol.

- unsigned l4_is_invalid_cap (l4_cap_idx_t c) L4_NOTHROW

  Test if a capability selector is the invalid capability.

- unsigned l4_is_valid_cap (l4_cap_idx_t c) L4_NOTHROW

  Test if a capability selector is a valid selector.

- unsigned l4_capability_equal (l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW

  Test if two capability selectors are equal.

- l4_cap_idx_t l4_capability_next (l4_cap_idx_t c) L4_NOTHROW

  Get the next capability selector after `c`.

16.383.1 Detailed Description

Common L4 ABI Data Types.

Definition in file types.h.

16.383.2 Function Documentation

16.383.2.1 l4_capability_next()

```
l4_cap_idx_t l4_capability_next (
            l4_cap_idx_t c )  [inline]
```

Get the next capability selector after `c`.

Parameters

c	The capability selector for which the next selector shall be computed.

Returns

The next capability selector after `c`.

Definition at line 463 of file types.h.

16.384 types.h

```
00001 /****************************************************************************/
00007 /*
00008  * (c) 2008-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  *
```

```
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
/*****************************************************************************/
#pragma once

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>


enum l4_msgtag_protocol
{
  L4_PROTO_NONE          = 0,
  L4_PROTO_ALLOW_SYSCALL = 1,
  L4_PROTO_PF_EXCEPTION  = 1,

  L4_PROTO_IRQ           =  -1L,
  L4_PROTO_PAGE_FAULT    =  -2L,
  L4_PROTO_PREEMPTION    =  -3L,
  L4_PROTO_SYS_EXCEPTION =  -4L,
  L4_PROTO_EXCEPTION     =  -5L,
  L4_PROTO_SIGMA0        =  -6L,
  L4_PROTO_IO_PAGE_FAULT =  -8L,
  L4_PROTO_KOBJECT       = -10L,
  L4_PROTO_TASK          = -11L,
  L4_PROTO_THREAD        = -12L,
  L4_PROTO_LOG           = -13L,
  L4_PROTO_SCHEDULER     = -14L,
  L4_PROTO_FACTORY       = -15L,
  L4_PROTO_VM            = -16L,
  L4_PROTO_DMA_SPACE     = -17L,
  L4_PROTO_IRQ_SENDER    = -18L,
  L4_PROTO_IRQ_MUX       = -19L,
  L4_PROTO_SEMAPHORE     = -20L,
  L4_PROTO_META          = -21L,
  L4_PROTO_IOMMU         = -22L,
  L4_PROTO_DEBUGGER      = -23L,
  L4_PROTO_SMCCC         = -24L,
};

enum L4_varg_type
{
  L4_VARG_TYPE_NIL    = 0x00,
  L4_VARG_TYPE_UMWORD = 0x01,
  L4_VARG_TYPE_MWORD  = 0x81,
  L4_VARG_TYPE_STRING = 0x02,
  L4_VARG_TYPE_FPAGE  = 0x03,

  L4_VARG_TYPE_SIGN   = 0x80,
};


enum l4_msgtag_flags
{
  // flags for received IPC
  L4_MSGTAG_ERROR        = 0x8000,

  // flags for sending IPC
  L4_MSGTAG_TRANSFER_FPU = 0x1000,
  L4_MSGTAG_SCHEDULE     = 0x2000,
  L4_MSGTAG_PROPAGATE    = 0x4000,

  L4_MSGTAG_FLAGS        = 0xf000,
};


typedef struct l4_msgtag_t
{
  l4_mword_t raw;
#ifdef __cplusplus
  long label() const L4_NOTHROW { return raw >> 16; }
  void label(long v) L4_NOTHROW { raw = (raw & 0x0ffff) | ((l4_umword_t)v << 16); }
  unsigned words() const L4_NOTHROW { return raw & 0x3f; }
  unsigned items() const L4_NOTHROW { return (raw >> 6) & 0x3f; }
  unsigned flags() const L4_NOTHROW { return raw & 0xf000; }
  bool is_page_fault() const L4_NOTHROW { return label() == L4_PROTO_PAGE_FAULT; }
```

```
  bool is_preemption() const L4_NOTHROW { return label() == L4_PROTO_PREEMPTION; }
  bool is_sys_exception() const L4_NOTHROW { return label() == L4_PROTO_SYS_EXCEPTION; }
  bool is_exception() const L4_NOTHROW { return label() == L4_PROTO_EXCEPTION; }
  bool is_sigma0() const L4_NOTHROW { return label() == L4_PROTO_SIGMA0; }
  bool is_io_page_fault() const L4_NOTHROW { return label() == L4_PROTO_IO_PAGE_FAULT; }
  unsigned has_error() const L4_NOTHROW { return raw & L4_MSGTAG_ERROR; }
#endif
} l4_msgtag_t;



L4_INLINE l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
                                unsigned flags) L4_NOTHROW;

L4_INLINE long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW;

L4_INLINE unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW;

L4_INLINE unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW;

L4_INLINE unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW;

L4_INLINE unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW;

L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW;

L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t) L4_NOTHROW;

L4_INLINE unsigned l4_msgtag_is_sys_exception(l4_msgtag_t t) L4_NOTHROW;

L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW;

L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW;

L4_INLINE unsigned l4_msgtag_is_io_page_fault(l4_msgtag_t t) L4_NOTHROW;

typedef unsigned long l4_cap_idx_t;

L4_INLINE unsigned l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW;

L4_INLINE unsigned l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW;

L4_INLINE unsigned l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW;

L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW;

/* ************************************************************************ */
/* Implementation */

L4_INLINE unsigned
l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW
{ return c & L4_INVALID_CAP_BIT; }

L4_INLINE unsigned
l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW
{ return !(c & L4_INVALID_CAP_BIT); }

L4_INLINE unsigned
l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW
{ return (c1 » L4_CAP_SHIFT) == (c2 » L4_CAP_SHIFT); }


L4_INLINE
l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
                      unsigned flags) L4_NOTHROW
{
  return (l4_msgtag_t){  (l4_mword_t)((l4_umword_t)label « 16)
                       | (l4_mword_t)(words & 0x3f)
                       | (l4_mword_t)((items & 0x3f) « 6)
                       | (l4_mword_t)(flags & 0xf000)};
}



L4_INLINE
long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW
{ return t.raw » 16; }

L4_INLINE
unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW
{ return t.raw & 0x3f; }

L4_INLINE
unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW
{ return (t.raw » 6) & 0x3f; }

L4_INLINE
```

```
00435 unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW
00436 { return t.raw & 0xf000; }
00437
00438
00439 L4_INLINE
00440 unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW
00441 { return t.raw & L4_MSGTAG_ERROR; }
00442
00443
00444
00445 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW
00446 { return l4_msgtag_label(t) == L4_PROTO_PAGE_FAULT; }
00447
00448 L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t) L4_NOTHROW
00449 { return l4_msgtag_label(t) == L4_PROTO_PREEMPTION; }
00450
00451 L4_INLINE unsigned l4_msgtag_is_sys_exception(l4_msgtag_t t) L4_NOTHROW
00452 { return l4_msgtag_label(t) == L4_PROTO_SYS_EXCEPTION; }
00453
00454 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW
00455 { return l4_msgtag_label(t) == L4_PROTO_EXCEPTION; }
00456
00457 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW
00458 { return l4_msgtag_label(t) == L4_PROTO_SIGMA0; }
00459
00460 L4_INLINE unsigned l4_msgtag_is_io_page_fault(l4_msgtag_t t) L4_NOTHROW
00461 { return l4_msgtag_label(t) == L4_PROTO_IO_PAGE_FAULT; }
00462
00463 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW
00464 { return c + L4_CAP_OFFSET; }
00465
00466 #include <l4/sys/__l4_fpage.h>
00467 #include <l4/sys/__timeout.h>
```

16.385 l4/sys/utcb.h File Reference

UTCB definitions.

```
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
```

Include dependency graph for utcb.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- union l4_msg_regs_t

  Encapsulation of the message-register block in the UTCB.
- struct l4_buf_regs_t

  Encapsulation of the buffer-registers block in the UTCB.
- struct l4_thread_regs_t

  Encapsulation of the thread-control-register block of the UTCB.

Typedefs

- typedef struct l4_utcb_t l4_utcb_t

  Opaque type for the UTCB.
- typedef union l4_msg_regs_t l4_msg_regs_t

  Encapsulation of the message-register block in the UTCB.
- typedef struct l4_buf_regs_t l4_buf_regs_t

  Encapsulation of the buffer-registers block in the UTCB.
- typedef struct l4_thread_regs_t l4_thread_regs_t

  Encapsulation of the thread-control-register block of the UTCB.

Functions

- l4_utcb_t ∗ l4_utcb (void) L4_NOTHROW L4_PURE

  Get the UTCB address.
- l4_msg_regs_t ∗ l4_utcb_mr (void) L4_NOTHROW L4_PURE

  Get the message-register block of a UTCB.
- l4_buf_regs_t ∗ l4_utcb_br (void) L4_NOTHROW L4_PURE

  Get the buffer-register block of a UTCB.
- l4_thread_regs_t ∗ l4_utcb_tcr (void) L4_NOTHROW L4_PURE

  Get the thread-control-register block of a UTCB.
- l4_exc_regs_t ∗ l4_utcb_exc (void) L4_NOTHROW L4_PURE

  Get the message-register block of a UTCB (for an exception IPC).
- l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const ∗u) L4_NOTHROW L4_PURE

  Access function to get the program counter of the exception state.
- void l4_utcb_exc_pc_set (l4_exc_regs_t ∗u, l4_addr_t pc) L4_NOTHROW

  Set the program counter register in the exception state.
- unsigned long l4_utcb_exc_typeval (l4_exc_regs_t const ∗u) L4_NOTHROW L4_PURE

  Get the value out of an exception UTCB that describes the type of exception.
- int l4_utcb_exc_is_pf (l4_exc_regs_t const ∗u) L4_NOTHROW L4_PURE

Check whether an exception IPC is a page fault.

- l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE

  Function to get the L4 style page fault address out of an exception.

- int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE

  Check whether an exception IPC was triggered via l4_thread_ex_regs().

- void l4_utcb_inherit_fpu (int switch_on) L4_NOTHROW

  Enable or disable inheritance of FPU state to receiver.

- l4_timeout_s l4_timeout_abs (l4_kernel_clock_t pint, int br) L4_NOTHROW

  Set an absolute timeout.

- unsigned l4_utcb_mr64_idx (unsigned idx) L4_NOTHROW

  Get index into 64bit message registers alias from native-sized index.

16.385.1 Detailed Description

UTCB definitions.

Definition in file utcb.h.

16.386 utcb.h

```
00001 /****************************************************************************/
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction.  Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License.  This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 /****************************************************************************/
00027 #ifndef _L4_SYS_UTCB_H
00028 #define _L4_SYS_UTCB_H
00029
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/compiler.h>
00032 #include <l4/sys/l4int.h>
00033
00067 typedef struct l4_utcb_t l4_utcb_t;
00068
00078 typedef union l4_msg_regs_t
00079 {
00080   l4_umword_t mr[L4_UTCB_GENERIC_DATA_SIZE];
00081   l4_uint64_t mr64[L4_UTCB_GENERIC_DATA_SIZE / (sizeof(l4_uint64_t)/sizeof(l4_umword_t))];
00082 } l4_msg_regs_t;
00083
00093 typedef struct l4_buf_regs_t
00094 {
00096   l4_umword_t bdr;
00097
00099   l4_umword_t br[L4_UTCB_GENERIC_BUFFERS_SIZE];
00100 } l4_buf_regs_t;
00101
00110 typedef struct l4_thread_regs_t
00111 {
00113   l4_umword_t  error;
00115   l4_timeout_t xfer;
00117   l4_umword_t  user[3];
00118 } l4_thread_regs_t;
00119
```

```
__BEGIN_DECLS

L4_CV l4_utcb_t *l4_utcb_wrap(void) L4_NOTHROW L4_PURE;

L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW L4_PURE;

L4_INLINE l4_utcb_t *l4_utcb(void) L4_NOTHROW L4_PURE;

L4_INLINE l4_msg_regs_t *l4_utcb_mr(void) L4_NOTHROW L4_PURE;

L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;

L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW L4_PURE;

L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;

L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void) L4_NOTHROW L4_PURE;

L4_INLINE l4_thread_regs_t *l4_utcb_tcr_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;

L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW L4_PURE;

L4_INLINE l4_exc_regs_t *l4_utcb_exc_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;

L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;

L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW;

L4_INLINE unsigned long l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;

L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;

L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;


L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;

L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW;

L4_INLINE void l4_utcb_inherit_fpu_u(l4_utcb_t *u, int switch_on) L4_NOTHROW;

L4_INLINE
l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t pint, int br,
                              l4_utcb_t *utcb) L4_NOTHROW;
L4_INLINE
l4_timeout_s l4_timeout_abs(l4_kernel_clock_t pint, int br) L4_NOTHROW;

L4_INLINE
unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW;

/*****************************************************************************
 * Implementations
 *****************************************************************************/

L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW
{ return (l4_msg_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }

L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u) L4_NOTHROW
{ return (l4_buf_regs_t*)((char*)u + L4_UTCB_BUF_REGS_OFFSET); }

L4_INLINE l4_thread_regs_t *l4_utcb_tcr_u(l4_utcb_t *u) L4_NOTHROW
{ return (l4_thread_regs_t*)((char*)u + L4_UTCB_THREAD_REGS_OFFSET); }

L4_INLINE l4_exc_regs_t *l4_utcb_exc_u(l4_utcb_t *u) L4_NOTHROW
{ return (l4_exc_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }

L4_INLINE void l4_utcb_inherit_fpu_u(l4_utcb_t *u, int switch_on) L4_NOTHROW
{
  if (switch_on)
    l4_utcb_br_u(u)->bdr |= L4_UTCB_INHERIT_FPU;
  else
    l4_utcb_br_u(u)->bdr &= ~L4_UTCB_INHERIT_FPU;
}

L4_INLINE l4_utcb_t *l4_utcb(void) L4_NOTHROW
{
#ifdef L4SYS_USE_UTCB_WRAP
  return l4_utcb_wrap();
#else
  return l4_utcb_direct();
#endif
}



L4_INLINE l4_msg_regs_t *l4_utcb_mr(void) L4_NOTHROW
```

```
00353 { return l4_utcb_mr_u(l4_utcb()); }
00354
00355 L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW
00356 { return l4_utcb_br_u(l4_utcb()); }
00357
00358 L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void) L4_NOTHROW
00359 { return l4_utcb_tcr_u(l4_utcb()); }
00360
00361 L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW
00362 { return l4_utcb_exc_u(l4_utcb()); }
00363
00364 L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW
00365 { l4_utcb_inherit_fpu_u(l4_utcb(), switch_on); }
00366
00367 L4_INLINE
00368 l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t val, int pos,
00369                               l4_utcb_t *utcb) L4_NOTHROW
00370 {
00371   union T
00372   {
00373     l4_kernel_clock_t t;
00374     l4_umword_t m[sizeof(l4_kernel_clock_t)/sizeof(l4_umword_t)];
00375   };
00376   l4_timeout_s to;
00377   to.t = 0x8000 | pos;
00378   ((union T*)(l4_utcb_br_u(utcb)->br + pos))->t = val;
00379   return to;
00380 }
00381
00382 L4_INLINE
00383 l4_timeout_s l4_timeout_abs(l4_kernel_clock_t val, int pos) L4_NOTHROW
00384 { return l4_timeout_abs_u(val, pos, l4_utcb()); }
00385
00386 L4_INLINE unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW
00387 { return idx / (sizeof(l4_uint64_t) / sizeof(l4_umword_t)); }
00388
00389 __END_DECLS
00390
00391 #endif /* ! _L4_SYS_UTCB_H */
```

16.387 arm/l4/sys/utcb.h File Reference

UTCB definitions for ARM.

`#include <l4/sys/types.h>`

Include dependency graph for utcb.h:

Data Structures

- struct l4_exc_regs_t

  UTCB structure for exceptions.

Typedefs

- typedef struct l4_exc_regs_t l4_exc_regs_t

  UTCB structure for exceptions.

Enumerations

- enum L4_utcb_consts_arm

  UTCB constants for ARM.

Functions

- l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const ∗u) L4_NOTHROW

  Access function to get the program counter of the exception state.
- void l4_utcb_exc_pc_set (l4_exc_regs_t ∗u, l4_addr_t pc) L4_NOTHROW

  Set the program counter register in the exception state.
- l4_umword_t l4_utcb_exc_typeval (l4_exc_regs_t const ∗u) L4_NOTHROW

  Get the value out of an exception UTCB that describes the type of exception.
- int l4_utcb_exc_is_pf (l4_exc_regs_t const ∗u) L4_NOTHROW

  Check whether an exception IPC is a page fault.
- l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const ∗u) L4_NOTHROW

  Function to get the L4 style page fault address out of an exception.
- int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const ∗u) L4_NOTHROW

  Check whether an exception IPC was triggered via l4_thread_ex_regs().

16.387.1 Detailed Description

UTCB definitions for ARM.

Definition in file utcb.h.

16.388 utcb.h

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#ifndef __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
#define __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__

#include <l4/sys/types.h>

typedef struct l4_exc_regs_t
{
  l4_umword_t pfa;
  l4_umword_t err;
  l4_umword_t r[13];
  l4_umword_t sp;
  l4_umword_t ulr;
  l4_umword_t _dummy1;
  l4_umword_t pc;
  l4_umword_t cpsr;
  l4_umword_t tpidruro;
  l4_umword_t tpidrurw;
} l4_exc_regs_t;

enum L4_utcb_consts_arm
{
  L4_UTCB_EXCEPTION_REGS_SIZE    = sizeof(l4_exc_regs_t) / sizeof(l4_umword_t),
  L4_UTCB_GENERIC_DATA_SIZE      = 63,
  L4_UTCB_GENERIC_BUFFERS_SIZE   = 58,

  L4_UTCB_MSG_REGS_OFFSET        = 0,
  L4_UTCB_BUF_REGS_OFFSET        = 64 * sizeof(l4_umword_t),
  L4_UTCB_THREAD_REGS_OFFSET     = 123 * sizeof(l4_umword_t),

  L4_UTCB_INHERIT_FPU            = 1UL << 24,

  L4_UTCB_OFFSET                 = 512,
};

#include_next <l4/sys/utcb.h>

/*
 * ==================================================================
 * Implementations.
 */

#ifdef __GNUC__
L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
{
  register l4_utcb_t *utcb __asm__ ("r0");
  __asm__ ("mov lr, pc          \n"
           "mov pc, #0xffffff00 \n"
           : "=r"(utcb) : : "lr");
  return utcb;
}
#endif

L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
{
  return u->pc;
}

L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
{
  u->pc = pc;
}

L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
{
  return u->err >> 26;
}
```

```
L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
{
  return ((u->err » 26) & 0x30) == 0x20;
}

L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
{
  return (u->pfa & ~7UL) | ((u->err » 5) & 2);
}

L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
{
  return l4_utcb_exc_typeval(u) == 0x3e;
}

#endif /* ! __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__ */
```

16.389 amd64/l4/sys/utcb.h File Reference

UTCB definitions for amd64.

`#include <l4/sys/types.h>`
Include dependency graph for utcb.h:

Data Structures

- struct l4_exc_regs_t

  UTCB structure for exceptions.

Typedefs

- typedef struct l4_exc_regs_t l4_exc_regs_t

  UTCB structure for exceptions.

Enumerations

- enum L4_utcb_consts_amd64

  UTCB constants for AMD64.

Functions

- l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const *u) L4_NOTHROW

  Access function to get the program counter of the exception state.
- void l4_utcb_exc_pc_set (l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW

  Set the program counter register in the exception state.
- l4_umword_t l4_utcb_exc_typeval (l4_exc_regs_t const *u) L4_NOTHROW

  Get the value out of an exception UTCB that describes the type of exception.
- int l4_utcb_exc_is_pf (l4_exc_regs_t const *u) L4_NOTHROW

  Check whether an exception IPC is a page fault.
- l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW

  Function to get the L4 style page fault address out of an exception.
- int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW

  Check whether an exception IPC was triggered via l4_thread_ex_regs().

16.389.1 Detailed Description

UTCB definitions for amd64.

Definition in file utcb.h.

16.390 utcb.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 /*****************************************************************************/
00025 #ifndef __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__
00026 #define __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__
00027
00028 #include <l4/sys/types.h>
00029
00039 enum L4_utcb_consts_amd64
00040 {
00041   L4_UTCB_EXCEPTION_REGS_SIZE    = 26,
00042   L4_UTCB_GENERIC_DATA_SIZE      = 63,
00043   L4_UTCB_GENERIC_BUFFERS_SIZE   = 58,
00044
00045   L4_UTCB_MSG_REGS_OFFSET        = 0,
00046   L4_UTCB_BUF_REGS_OFFSET        = 64 * sizeof(l4_umword_t),
00047   L4_UTCB_THREAD_REGS_OFFSET     = 123 * sizeof(l4_umword_t),
00048
```

```
  L4_UTCB_INHERIT_FPU            = 1UL « 24,
  L4_UTCB_OFFSET                 = 1024,
};

typedef struct l4_exc_regs_t
{
  l4_umword_t r15;
  l4_umword_t r14;
  l4_umword_t r13;
  l4_umword_t r12;
  l4_umword_t r11;
  l4_umword_t r10;
  l4_umword_t r9;
  l4_umword_t r8;
  l4_umword_t rdi;
  l4_umword_t rsi;
  l4_umword_t rbp;
  l4_umword_t pfa;
  l4_umword_t rbx;
  l4_umword_t rdx;
  l4_umword_t rcx;
  l4_umword_t rax;
  l4_umword_t trapno;
  l4_umword_t err;
  l4_umword_t ip;
  l4_umword_t dummy1;
  l4_umword_t flags;
  l4_umword_t sp;
  l4_umword_t ss;
  l4_umword_t fs_base;
  l4_umword_t gs_base;
  l4_uint16_t ds, es, fs, gs;
} l4_exc_regs_t;


#include_next <l4/sys/utcb.h>

/*
 * ================================================================
 * Implementations.
 */

L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
{
  l4_utcb_t *res;
  __asm__ ( "mov %%gs:0, %0 \n" : "=r"(res));
  return res;
}

L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
{
  return u->ip;
}

L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
{
  u->ip = pc;
}

L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
{
  return u->trapno;
}

L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
{
  return u->trapno == 14;
}

L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
{
  return (u->pfa & ~7UL) | (u->err & 2);
}

L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
{
  return l4_utcb_exc_typeval(u) == 0xff;
}

#endif /* ! __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__ */
```

16.391 x86/l4/sys/utcb.h File Reference

UTCB definitions for X86.

```
#include <l4/sys/types.h>
```

Include dependency graph for utcb.h:

Data Structures

- struct l4_exc_regs_t

  UTCB structure for exceptions.

Typedefs

- typedef struct l4_exc_regs_t l4_exc_regs_t

  UTCB structure for exceptions.

Enumerations

- enum L4_utcb_consts_x86 {
  L4_UTCB_EXCEPTION_REGS_SIZE = 19 , L4_UTCB_GENERIC_DATA_SIZE = 63 , L4_UTCB_GENERIC_BUFFERS_SIZE = 58 , L4_UTCB_MSG_REGS_OFFSET = 0 ,
  L4_UTCB_BUF_REGS_OFFSET = 64 ∗ sizeof(l4_umword_t) , L4_UTCB_THREAD_REGS_OFFSET = 123 ∗ sizeof(l4_umword_t) , L4_UTCB_INHERIT_FPU = 1UL << 24 , L4_UTCB_OFFSET = 512 }

  UTCB constants for x86.

Functions

- l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const *u) L4_NOTHROW

  Access function to get the program counter of the exception state.
- void l4_utcb_exc_pc_set (l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW

  Set the program counter register in the exception state.
- l4_umword_t l4_utcb_exc_typeval (l4_exc_regs_t const *u) L4_NOTHROW

  Get the value out of an exception UTCB that describes the type of exception.
- int l4_utcb_exc_is_pf (l4_exc_regs_t const *u) L4_NOTHROW

  Check whether an exception IPC is a page fault.
- l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW

  Function to get the L4 style page fault address out of an exception.
- int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW

  Check whether an exception IPC was triggered via l4_thread_ex_regs().

16.391.1 Detailed Description

UTCB definitions for X86.

Definition in file utcb.h.

16.392 utcb.h

```
00001 /*****************************************************************************/
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 /*****************************************************************************/
00026 #ifndef __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00027 #define __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00028
00029 #include <l4/sys/types.h>
00030
00041 enum L4_utcb_consts_x86
00042 {
00044   L4_UTCB_EXCEPTION_REGS_SIZE    = 19,
00045
00047   L4_UTCB_GENERIC_DATA_SIZE      = 63,
00048
00050   L4_UTCB_GENERIC_BUFFERS_SIZE   = 58,
00051
00053   L4_UTCB_MSG_REGS_OFFSET        = 0,
00054
00056   L4_UTCB_BUF_REGS_OFFSET        = 64 * sizeof(l4_umword_t),
00057
00059   L4_UTCB_THREAD_REGS_OFFSET     = 123 * sizeof(l4_umword_t),
00060
00062   L4_UTCB_INHERIT_FPU            = 1UL « 24,
00063
00065   L4_UTCB_OFFSET                 = 512,
00066 };
00067
00072 typedef struct l4_exc_regs_t
```

```
{
  l4_umword_t es;
  l4_umword_t ds;
  l4_umword_t gs;
  l4_umword_t fs;
  l4_umword_t edi;
  l4_umword_t esi;
  l4_umword_t ebp;
  l4_umword_t pfa;
  l4_umword_t ebx;
  l4_umword_t edx;
  l4_umword_t ecx;
  l4_umword_t eax;
  l4_umword_t trapno;
  l4_umword_t err;
  l4_umword_t ip;
  l4_umword_t dummy1;
  l4_umword_t flags;
  l4_umword_t sp;
  l4_umword_t ss;
} l4_exc_regs_t;

#include_next <l4/sys/utcb.h>

/*
 * ==================================================================
 * Implementations.
 */

L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
{
  l4_utcb_t *utcb;
  __asm__ ("mov %%fs:0, %0" : "=r" (utcb));
  return utcb;
}

L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
{
  return u->ip;
}

L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
{
  u->ip = pc;
}

L4_INLINE void l4_utcb_exc_sp_set(l4_exc_regs_t *u, l4_addr_t sp) L4_NOTHROW
{
  u->sp = sp;
}

L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
{
  return u->trapno;
}

L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
{
  return u->trapno == 14;
}

L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
{
  return (u->pfa & ~7UL) | (u->err & 2);
}

L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
{
  return l4_utcb_exc_typeval(u) == 0xff;
}

#endif /* ! __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__ */
```

16.393 l4/sys/vcon File Reference

C++ Virtual console interface.

```
#include <l4/sys/icu>
#include <l4/sys/vcon.h>
```

```
#include <l4/sys/capability>
```

Include dependency graph for vcon:

This graph shows which files directly or indirectly include this file:

Data Structures

- class L4::Vcon

  C++ L4 Vcon interface, see Virtual Console for the C interface.

Namespaces

- L4

  L4 low-level kernel interface.

16.393.1 Detailed Description

C++ Virtual console interface.

Definition in file vcon.

16.394 vcon

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/icu>
00028 #include <l4/sys/vcon.h>
00029 #include <l4/sys/capability>
00030
00031 namespace L4 {
00032
00056 class Vcon :
00057   public Kobject_t<Vcon, Icu, L4_PROTO_LOG>
00058 {
00059 public:
00075   l4_msgtag_t
00076   send(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00077   { return l4_vcon_send_u(cap(), buf, size, utcb); }
00078
00089   long
00090   write(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00091   { return l4_vcon_write_u(cap(), buf, size, utcb); }
00092
00106   int
00107   read(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00108   { return l4_vcon_read_u(cap(), buf, size, utcb); }
00109
00131   int
00132   read_with_flags(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00133   { return l4_vcon_read_with_flags_u(cap(), buf, size, utcb); }
00134
00144   l4_msgtag_t
00145   set_attr(l4_vcon_attr_t const *attr, l4_utcb_t *utcb = l4_utcb()) const noexcept
00146   { return l4_vcon_set_attr_u(cap(), attr, utcb); }
00147
00157   l4_msgtag_t
00158   get_attr(l4_vcon_attr_t *attr, l4_utcb_t *utcb = l4_utcb()) const noexcept
00159   { return l4_vcon_get_attr_u(cap(), attr, utcb); }
00160
00161   typedef L4::Typeid::Raw_ipc<Vcon> Rpcs;
00162 };
00163
00164 }
```

16.395 l4/sys/vcon.h File Reference

Virtual console interface.

```
#include <l4/sys/ipc.h>
```

Include dependency graph for vcon.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct l4_vcon_attr_t

  Vcon attribute structure.

Typedefs

- typedef struct l4_vcon_attr_t l4_vcon_attr_t

  Vcon attribute structure.

Enumerations

- enum L4_vcon_size_consts { L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) ∗ sizeof(l4_umword_t) , L4_VCON_READ_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 1) ∗ sizeof(l4_umword_t) }

  Size constants.
- enum L4_vcon_read_flags { L4_VCON_READ_SIZE_MASK = 0x3fffffff , L4_VCON_READ_STAT_BREAK = 1 << 30 , L4_VCON_READ_STAT_DONE = 1 << 31 }

  Vcon read flags.
- enum L4_vcon_i_flags { L4_VCON_INLCR = 000100 , L4_VCON_IGNCR = 000200 , L4_VCON_ICRNL = 000400 }

  Input flags.
- enum L4_vcon_o_flags { L4_VCON_ONLCR = 000004 , L4_VCON_OCRNL = 000010 , L4_VCON_ONLRET = 000040 }

  Output flags.
- enum L4_vcon_l_flags { L4_VCON_ICANON = 000002 , L4_VCON_ECHO = 000010 }

  Local flags.
- enum L4_vcon_ops { L4_VCON_WRITE_OP = 0UL , L4_VCON_READ_OP = 1UL , L4_VCON_SET_ATTR_OP = 2UL , L4_VCON_GET_ATTR_OP = 3UL }

  Operations on vcon objects.

Functions

- l4_msgtag_t l4_vcon_send (l4_cap_idx_t vcon, char const ∗buf, unsigned size) L4_NOTHROW

  Send data to virtual console.
- l4_msgtag_t l4_vcon_send_u (l4_cap_idx_t vcon, char const ∗buf, unsigned size, l4_utcb_t ∗utcb) L4_NOTHROW

  Send data to `this` virtual console.
- long l4_vcon_write (l4_cap_idx_t vcon, char const ∗buf, unsigned size) L4_NOTHROW

  Write data to virtual console.
- long l4_vcon_write_u (l4_cap_idx_t vcon, char const ∗buf, unsigned size, l4_utcb_t ∗utcb) L4_NOTHROW

  Write data to `this` virtual console.
- int l4_vcon_read (l4_cap_idx_t vcon, char ∗buf, unsigned size) L4_NOTHROW

  Read data from virtual console.
- int l4_vcon_read_u (l4_cap_idx_t vcon, char ∗buf, unsigned size, l4_utcb_t ∗utcb) L4_NOTHROW

  Read data from `this` virtual console.
- int l4_vcon_read_with_flags (l4_cap_idx_t vcon, char ∗buf, unsigned size) L4_NOTHROW

  Read data from virtual console, extended version including flags.
- l4_msgtag_t l4_vcon_set_attr (l4_cap_idx_t vcon, l4_vcon_attr_t const ∗attr) L4_NOTHROW

  Set attributes of a Vcon.
- l4_msgtag_t l4_vcon_set_attr_u (l4_cap_idx_t vcon, l4_vcon_attr_t const ∗attr, l4_utcb_t ∗utcb) L4_NOTHROW

  Set the attributes of `this` virtual console.
- l4_msgtag_t l4_vcon_get_attr (l4_cap_idx_t vcon, l4_vcon_attr_t ∗attr) L4_NOTHROW

  Get attributes of a Vcon.
- l4_msgtag_t l4_vcon_get_attr_u (l4_cap_idx_t vcon, l4_vcon_attr_t ∗attr, l4_utcb_t ∗utcb) L4_NOTHROW

  Get attributes of `this` virtual console.
- void l4_vcon_set_attr_raw (l4_vcon_attr_t ∗attr) L4_NOTHROW

  Set terminal attributes to disable all special processing.

16.395.1 Detailed Description

Virtual console interface.

Definition in file vcon.h.

16.395.2 Enumeration Type Documentation

16.395.2.1 L4_vcon_read_flags

```
enum L4_vcon_read_flags
```

Vcon read flags.

Enumerator

L4_VCON_READ_SIZE_MASK	Size mask.
L4_VCON_READ_STAT_BREAK	Break condition flag.
L4_VCON_READ_STAT_DONE	Done condition flag.

Definition at line 178 of file vcon.h.

16.396 vcon.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/ipc.h>
00027
00067 L4_INLINE l4_msgtag_t
00068 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW;
00069
00076 L4_INLINE l4_msgtag_t
00077 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
00078
00090 L4_INLINE long
00091 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW;
00092
00099 L4_INLINE long
00100 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
00101
```

```
enum L4_vcon_size_consts
{
  L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t),
  L4_VCON_READ_SIZE  = (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(l4_umword_t),
};

L4_INLINE int
l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW;

L4_INLINE int
l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE int
l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW;

L4_INLINE int
l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
                          l4_utcb_t *utcb) L4_NOTHROW;

enum L4_vcon_read_flags
{
  L4_VCON_READ_SIZE_MASK  = 0x3fffffff,
  L4_VCON_READ_STAT_BREAK = 1 « 30,
  L4_VCON_READ_STAT_DONE  = 1 « 31,
};

typedef struct l4_vcon_attr_t
{
  l4_umword_t i_flags;
  l4_umword_t o_flags;
  l4_umword_t l_flags;

#ifdef __cplusplus
  inline void set_raw();
#endif
} l4_vcon_attr_t;

enum L4_vcon_i_flags
{
  L4_VCON_INLCR  = 000100,
  L4_VCON_IGNCR  = 000200,
  L4_VCON_ICRNL  = 000400,
};

enum L4_vcon_o_flags
{
  L4_VCON_ONLCR  = 000004,
  L4_VCON_OCRNL  = 000010,
  L4_VCON_ONLRET = 000040,
};

enum L4_vcon_l_flags
{
  L4_VCON_ICANON = 000002,
  L4_VCON_ECHO   = 000010,
};

L4_INLINE l4_msgtag_t
l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_vcon_set_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr,
                   l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_vcon_get_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t *attr,
                   l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE void
l4_vcon_set_attr_raw(l4_vcon_attr_t *attr) L4_NOTHROW;


enum L4_vcon_ops
{
  L4_VCON_WRITE_OP       = 0UL,
  L4_VCON_READ_OP        = 1UL,
  L4_VCON_SET_ATTR_OP    = 2UL,
  L4_VCON_GET_ATTR_OP    = 3UL,
};

/******* Implementations *******************/

L4_INLINE l4_msgtag_t
l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
```

```
{
  l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
  mr->mr[0] = L4_VCON_WRITE_OP;
  mr->mr[1] = size;
  __builtin_memcpy(&mr->mr[2], buf, size);
  return l4_ipc_send(vcon, utcb,
                     l4_msgtag(L4_PROTO_LOG, 2 + l4_bytes_to_mwords(size),
                               0, L4_MSGTAG_SCHEDULE),
                     L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
{
  return l4_vcon_send_u(vcon, buf, size, l4_utcb());
}

L4_INLINE long
l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msgtag_t t;

  if (size > L4_VCON_WRITE_SIZE)
    size = L4_VCON_WRITE_SIZE;

  t = l4_vcon_send_u(vcon, buf, size, utcb);
  if (l4_msgtag_has_error(t))
    return l4_error(t);

  return (long) size;
}

L4_INLINE long
l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
{
  return l4_vcon_write_u(vcon, buf, size, l4_utcb());
}

L4_INLINE int
l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
                          l4_utcb_t *utcb) L4_NOTHROW
{
  int ret;
  unsigned r;
  l4_msg_regs_t *mr;

  mr = l4_utcb_mr_u(utcb);
  mr->mr[0] = (size << 16) | L4_VCON_READ_OP;

  ret = l4_error_u(l4_ipc_call(vcon, utcb,
                               l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
                               L4_IPC_NEVER),
                   utcb);
  if (ret < 0)
    return ret;

  r = mr->mr[0] & L4_VCON_READ_SIZE_MASK;

  if (!(mr->mr[0] & L4_VCON_READ_STAT_DONE)) // !eof
    ret = size + 1;
  else if (r < size)
    ret = r;
  else
    ret = size;

  if (L4_LIKELY(buf != NULL))
    __builtin_memcpy(buf, &mr->mr[1], r < size ? r : size);

  return ret | (mr->mr[0] & ~(L4_VCON_READ_STAT_DONE | L4_VCON_READ_SIZE_MASK));
}

L4_INLINE int
l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
{
  return l4_vcon_read_with_flags_u(vcon, buf, size, l4_utcb());
}

L4_INLINE int
l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
{
  int r = l4_vcon_read_with_flags_u(vcon, buf, size, utcb);
  if (r < 0)
    return r;

  return r & L4_VCON_READ_SIZE_MASK;
}
```

```
L4_INLINE int
l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
{
  return l4_vcon_read_u(vcon, buf, size, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_vcon_set_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr,
                   l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);

  mr->mr[0] = L4_VCON_SET_ATTR_OP;
  __builtin_memcpy(&mr->mr[1], attr, sizeof(*attr));

  return l4_ipc_call(vcon, utcb,
                     l4_msgtag(L4_PROTO_LOG, 4, 0, 0),
                     L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW
{
  return l4_vcon_set_attr_u(vcon, attr, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_vcon_get_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t *attr,
                   l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msgtag_t res;
  l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);

  mr->mr[0] = L4_VCON_GET_ATTR_OP;

  res = l4_ipc_call(vcon, utcb,
                    l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
                    L4_IPC_NEVER);
  if (l4_error_u(res, utcb) >= 0)
    __builtin_memcpy(attr, &mr->mr[1], sizeof(*attr));

  return res;
}

L4_INLINE l4_msgtag_t
l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW
{
  return l4_vcon_get_attr_u(vcon, attr, l4_utcb());
}

L4_INLINE void
l4_vcon_set_attr_raw(l4_vcon_attr_t *attr) L4_NOTHROW
{
  attr->i_flags = 0;
  attr->o_flags = 0;
  attr->l_flags = 0;
}

#ifdef __cplusplus
inline void
l4_vcon_attr_t::set_raw()
{ l4_vcon_set_attr_raw(this); }
#endif
```

16.397 l4/sys/vhw.h File Reference

Descriptors for virtual hardware (under UX).

```
#include <l4/sys/types.h>
#include <l4/sys/kip.h>
```

Include dependency graph for vhw.h:

Data Structures

- struct l4_vhw_entry

  Description of a device.
- struct l4_vhw_descriptor

  Virtual hardware devices description.

Enumerations

- enum l4_vhw_entry_type { L4_TYPE_VHW_NONE , L4_TYPE_VHW_FRAMEBUFFER , L4_TYPE_VHW_INPUT , L4_TYPE_VHW_NET }

  Type of device.

16.397.1 Detailed Description

Descriptors for virtual hardware (under UX).

Definition in file vhw.h.

16.398 vhw.h

```
/*****************************************************************************/
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
/*****************************************************************************/
#ifndef _L4_SYS_VHW_H
#define _L4_SYS_VHW_H

#include <l4/sys/types.h>
#include <l4/sys/kip.h>

enum l4_vhw_entry_type {
  L4_TYPE_VHW_NONE,
  L4_TYPE_VHW_FRAMEBUFFER,
  L4_TYPE_VHW_INPUT,
  L4_TYPE_VHW_NET,
};

struct l4_vhw_entry {
  enum l4_vhw_entry_type type;
  l4_uint32_t            provider_pid;
  l4_addr_t              mem_start;
  l4_addr_t              mem_size;
  l4_uint32_t            irq_no;
  l4_uint32_t            fd;
};

struct l4_vhw_descriptor {
  l4_uint32_t magic;
  l4_uint8_t  version;
  l4_uint8_t  count;
  l4_uint8_t  pad1;
  l4_uint8_t  pad2;
  struct l4_vhw_entry descs[];
};

enum {
  L4_VHW_MAGIC = 0x56687765,
};

static inline struct l4_vhw_descriptor *
l4_vhw_get(l4_kernel_info_t *kip) L4_NOTHROW
{
  struct l4_vhw_descriptor *v
    = (struct l4_vhw_descriptor *)(((unsigned long)kip) + kip->vhw_offset);

  if (v->magic == L4_VHW_MAGIC)
    return v;

  return NULL;
}

static inline struct l4_vhw_entry *
l4_vhw_get_entry(struct l4_vhw_descriptor *v, int entry) L4_NOTHROW
{
  return v->descs + entry;
}

static inline struct l4_vhw_entry *
l4_vhw_get_entry_type(struct l4_vhw_descriptor *v, enum l4_vhw_entry_type t) L4_NOTHROW
{
  int i;
  struct l4_vhw_entry *e = v->descs;

  for (i = 0; i < v->count; i++, e++)
    if (e->type == t)
      return e;

  return NULL;
}
```

```
00114
00115 #endif /* ! _L4_SYS_VHW_H */
```

16.399 l4/sys/vm File Reference

Virtualization interface.

```
#include <l4/sys/vm.h>
#include <l4/sys/task>
```

Include dependency graph for vm:

Data Structures

- class L4::Vm

  Virtual machine host address space.

Namespaces

- L4

  L4 low-level kernel interface.

16.399.1 Detailed Description

Virtualization interface.

Definition in file vm.

16.400 vm

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/vm.h>
00028 #include <l4/sys/task>
00029
00030 namespace L4 {
00031
00041 class Vm : public Kobject_t<Vm, Task, L4_PROTO_VM>
00042 {
00043 protected:
00044   Vm();
00045
00046 private:
00047   Vm(Vm const &);
00048   void operator = (Vm const &);
00049 };
00050
00051 };
```

16.401 l4/util/assert.h File Reference

Some useful assert-style macros.

```
#include <stdio.h>
#include <assert.h>
```

Include dependency graph for assert.h:

This graph shows which files directly or indirectly include this file:

16.401.1 Detailed Description

Some useful assert-style macros.

Date

09/2009

Author

Bjoern Doebel doebel@tudos.org

Definition in file assert.h.

16.402 assert.h

```
00001 /****************************************************************************/
00009 /*
00010  * (c) 2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016
00017 /****************************************************************************/
00018 #pragma once
00019
00020 #ifdef NDEBUG
00021
00022 #define DO_NOTHING              do {} while (0)
00023 #define ASSERT_ASSERT(x)        DO_NOTHING
00024 #define ASSERT_VALID(c)         DO_NOTHING
00025 #define ASSERT_EQUAL(a,b)       DO_NOTHING
00026 #define ASSERT_NOT_EQUAL(a,b)   DO_NOTHING
00027 #define ASSERT_LOWER_EQ(a,b)    DO_NOTHING
00028 #define ASSERT_GREATER_EQ(a,b)  DO_NOTHING
00029 #define ASSERT_BETWEEN(a,b,c)   DO_NOTHING
00030 #define ASSERT_IPC_OK(i)        DO_NOTHING
00031 #define ASSERT_OK(e)            do { (void)e; } while (0)
00032 #define ASSERT_NOT_NULL(p)      DO_NOTHING
00033 #ifndef assert
00034 #define assert(cond)            DO_NOTHING
00035 #endif
00036
00037 #else // NDEBUG
00038
00039 #ifndef ASSERT_PRINTF
00040 #include <stdio.h>
00041 #define ASSERT_PRINTF printf
00042 #endif
00043 #ifndef ASSERT_ASSERT
00044 #include <assert.h>
00045 #define ASSERT_ASSERT(x) assert(x)
```

```
#endif

#define ASSERT_VALID(cap) \
  do { \
    typeof(cap) _cap = cap; \
    if (l4_is_invalid_cap(_cap)) { \
      ASSERT_PRINTF("%s: Cap invalid.\n", __func__); \
      ASSERT_ASSERT(!l4_is_invalid_cap(_cap)); \
    } \
  } while (0)


#define ASSERT_EQUAL(a, b) \
  do { \
    typeof(a) _a = a; \
    typeof(b) _b = b; \
    if (_a != _b) { \
      ASSERT_PRINTF("%s:\n", __func__); \
      ASSERT_PRINTF("    "#a" (%lx) != "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
      ASSERT_ASSERT(_a == _b); \
    } \
  } while (0)


#define ASSERT_NOT_EQUAL(a, b) \
  do { \
    typeof(a) _a = a; \
    typeof(b) _b = b; \
    if (_a == _b) { \
      ASSERT_PRINTF("%s:\n", __func__); \
      ASSERT_PRINTF("    "#a" (%lx) == "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
      ASSERT_ASSERT(_a != _b); \
    } \
  } while (0)


#define ASSERT_LOWER_EQ(val, max) \
  do { \
    typeof(val) _val = val; \
    typeof(max) _max = max; \
    if (_val > _max) { \
      ASSERT_PRINTF("%s:\n", __func__); \
      ASSERT_PRINTF("    "#val" (%lx) > "#max" (%lx)\n", (unsigned long)_val, (unsigned long)_max); \
      ASSERT_ASSERT(_val <= _max); \
    } \
  } while (0)


#define ASSERT_GREATER_EQ(val, min) \
  do { \
    typeof(val) _val = val; \
    typeof(min) _min = min; \
    if (_val < _min) { \
      ASSERT_PRINTF("%s:\n", __func__); \
      ASSERT_PRINTF("    "#val" (%lx) < "#min" (%lx)\n", (unsigned long)_val, (unsigned long)_min); \
      ASSERT_ASSERT(_val >= _min); \
    } \
  } while (0)


#define ASSERT_BETWEEN(val, min, max) \
  ASSERT_LOWER_EQ((val), (max)); \
  ASSERT_GREATER_EQ((val), (min));


#define ASSERT_IPC_OK(msgtag) \
  do { \
    int _r = l4_ipc_error(msgtag, l4_utcb()); \
    if (_r) { \
      ASSERT_PRINTF("%s: IPC Error: %lx\n", __func__, _r); \
      ASSERT_ASSERT(_r == 0); \
    } \
  } while (0)

#define ASSERT_OK(val)          ASSERT_EQUAL((val), 0)
#define ASSERT_NOT_NULL(ptr)    ASSERT_NOT_EQUAL((ptr), (void *)0)

#endif // NDEBUG
```

16.403 l4/sys/assert.h File Reference

Low-level assert implementation.

```
#include <l4/sys/compiler.h>
#include <l4/sys/thread.h>
#include <l4/sys/vcon.h>
```

Include dependency graph for assert.h:

This graph shows which files directly or indirectly include this file:

Macros

- #define l4_assert(expr)

  Low-level assert.

16.403.1 Detailed Description

Low-level assert implementation.

Definition in file assert.h.

16.403.2 Macro Definition Documentation

16.403.2.1 l4_assert

```
#define l4_assert(
            expr )
```

Value:

```
l4_assert_fn(expr, __FILE__ ":" L4_stringify(__LINE__) ": Assertion \"" \
                   L4_stringify(expr) "\" failed.\n")
```

Low-level assert.

Parameters

expr	Expression to be evaluate for the assertion.

This assertion is a low-level implementation that directly uses kernel primitives. Only use l4_assert() when the standard assert() functionality is not available.

Definition at line 43 of file assert.h.

16.404 assert.h

```
00001
00005 /*
00006  * (c) 2015 Adam Lackorzynski <adam@l4re.org>
00007  *
00008  * This file is part of L4Re and distributed under the terms of the
00009  * GNU General Public License 2.
00010  * Please see the COPYING-GPL-2 file for details.
00011  *
00012  * As a special exception, you may use this file as part of a free software
00013  * library without restriction.  Specifically, if other files instantiate
00014  * templates or use macros or inline functions from this file, or you compile
00015  * this file and link it with other files to produce an executable, this
00016  * file does not by itself cause the resulting executable to be covered by
00017  * the GNU General Public License.  This exception does not however
00018  * invalidate any other reasons why the executable file might be covered by
00019  * the GNU General Public License.
00020  */
00021 #pragma once
00022
00023 #ifdef NDEBUG
00024
00025 #define l4_assert(x) do { } while (0)
00026 #define l4_check(x) do { (void)(x); } while (0)
00027
00028 #else
00029
00030 #include <l4/sys/compiler.h>
00031 #include <l4/sys/thread.h>
00032 #include <l4/sys/vcon.h>
```

```
#define l4_assert(expr) \
  l4_assert_fn(expr, __FILE__ ":" L4_stringify(__LINE__) ": Assertion \"" \
                     L4_stringify(expr) "\" failed.\n")

#define l4_check(expr) l4_assert(expr)

L4_ALWAYS_INLINE
void l4_assert_fn(bool expr, const char *text) L4_NOTHROW;

L4_INLINE L4_NORETURN
void l4_assert_abort(const char *text) L4_NOTHROW;


/* IMPLEMENTATION -----------------------------------------------------------*/

L4_INLINE L4_NORETURN
void l4_assert_abort(const char *text) L4_NOTHROW
{
  l4_vcon_write(L4_BASE_LOG_CAP, text, __builtin_strlen(text));
  for (;;)
    l4_thread_ex_regs(L4_INVALID_CAP, ~0UL, ~0UL,
                      L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);
}

L4_ALWAYS_INLINE
void l4_assert_fn(bool expr, const char *text) L4_NOTHROW
{
  if (L4_LIKELY(expr))
    return;

  l4_assert_abort(text);
}

#endif /* NDEBUG */
```

16.405 l4/util/atomic.h File Reference

atomic operations header and generic implementations

`#include <l4/sys/l4int.h>`
`#include <l4/sys/compiler.h>`
Include dependency graph for atomic.h:

This graph shows which files directly or indirectly include this file:

Functions

- int l4util_cmpxchg64 (volatile l4_uint64_t *dest, l4_uint64_t cmp_val, l4_uint64_t new_val)

  Atomic compare and exchange (64 bit version)
- int l4util_cmpxchg32 (volatile l4_uint32_t *dest, l4_uint32_t cmp_val, l4_uint32_t new_val)

  Atomic compare and exchange (32 bit version)
- int l4util_cmpxchg16 (volatile l4_uint16_t *dest, l4_uint16_t cmp_val, l4_uint16_t new_val)

  Atomic compare and exchange (16 bit version)
- int l4util_cmpxchg8 (volatile l4_uint8_t *dest, l4_uint8_t cmp_val, l4_uint8_t new_val)

  Atomic compare and exchange (8 bit version)
- int l4util_cmpxchg (volatile l4_umword_t *dest, l4_umword_t cmp_val, l4_umword_t new_val)

  Atomic compare and exchange (machine wide fields)
- l4_uint32_t l4util_xchg32 (volatile l4_uint32_t *dest, l4_uint32_t val)

  Atomic exchange (32 bit version)
- l4_uint16_t l4util_xchg16 (volatile l4_uint16_t *dest, l4_uint16_t val)

  Atomic exchange (16 bit version)
- l4_uint8_t l4util_xchg8 (volatile l4_uint8_t *dest, l4_uint8_t val)

  Atomic exchange (8 bit version)
- l4_umword_t l4util_xchg (volatile l4_umword_t *dest, l4_umword_t val)

  Atomic exchange (machine wide fields)
- void l4util_atomic_add (volatile long *dest, long val)

  Atomic add.
- void l4util_atomic_inc (volatile long *dest)

  Atomic increment.

Atomic add/sub/and/or (8,16,32 bit version) without result

- void l4util_add8 (volatile l4_uint8_t *dest, l4_uint8_t val)
- void l4util_add16 (volatile l4_uint16_t *dest, l4_uint16_t val)
- void l4util_add32 (volatile l4_uint32_t *dest, l4_uint32_t val)
- void l4util_sub8 (volatile l4_uint8_t *dest, l4_uint8_t val)
- void l4util_sub16 (volatile l4_uint16_t *dest, l4_uint16_t val)
- void l4util_sub32 (volatile l4_uint32_t *dest, l4_uint32_t val)
- void l4util_and8 (volatile l4_uint8_t *dest, l4_uint8_t val)
- void l4util_and16 (volatile l4_uint16_t *dest, l4_uint16_t val)

- void l4util_and32 (volatile l4_uint32_t *dest, l4_uint32_t val)
- void l4util_or8 (volatile l4_uint8_t *dest, l4_uint8_t val)
- void l4util_or16 (volatile l4_uint16_t *dest, l4_uint16_t val)
- void l4util_or32 (volatile l4_uint32_t *dest, l4_uint32_t val)

Atomic add/sub/and/or operations (8,16,32 bit) with result

- l4_uint8_t l4util_add8_res (volatile l4_uint8_t *dest, l4_uint8_t val)
- l4_uint16_t l4util_add16_res (volatile l4_uint16_t *dest, l4_uint16_t val)
- l4_uint32_t l4util_add32_res (volatile l4_uint32_t *dest, l4_uint32_t val)
- l4_uint8_t l4util_sub8_res (volatile l4_uint8_t *dest, l4_uint8_t val)
- l4_uint16_t l4util_sub16_res (volatile l4_uint16_t *dest, l4_uint16_t val)
- l4_uint32_t l4util_sub32_res (volatile l4_uint32_t *dest, l4_uint32_t val)
- l4_uint8_t l4util_and8_res (volatile l4_uint8_t *dest, l4_uint8_t val)
- l4_uint16_t l4util_and16_res (volatile l4_uint16_t *dest, l4_uint16_t val)
- l4_uint32_t l4util_and32_res (volatile l4_uint32_t *dest, l4_uint32_t val)
- l4_uint8_t l4util_or8_res (volatile l4_uint8_t *dest, l4_uint8_t val)
- l4_uint16_t l4util_or16_res (volatile l4_uint16_t *dest, l4_uint16_t val)
- l4_uint32_t l4util_or32_res (volatile l4_uint32_t *dest, l4_uint32_t val)

Atomic inc/dec (8,16,32 bit) without result

- void l4util_inc8 (volatile l4_uint8_t *dest)
- void l4util_inc16 (volatile l4_uint16_t *dest)
- void l4util_inc32 (volatile l4_uint32_t *dest)
- void l4util_dec8 (volatile l4_uint8_t *dest)
- void l4util_dec16 (volatile l4_uint16_t *dest)
- void l4util_dec32 (volatile l4_uint32_t *dest)

Atomic inc/dec (8,16,32 bit) with result

- l4_uint8_t l4util_inc8_res (volatile l4_uint8_t *dest)
- l4_uint16_t l4util_inc16_res (volatile l4_uint16_t *dest)
- l4_uint32_t l4util_inc32_res (volatile l4_uint32_t *dest)
- l4_uint8_t l4util_dec8_res (volatile l4_uint8_t *dest)
- l4_uint16_t l4util_dec16_res (volatile l4_uint16_t *dest)
- l4_uint32_t l4util_dec32_res (volatile l4_uint32_t *dest)

16.405.1 Detailed Description

atomic operations header and generic implementations

Date

10/20/2000

Author

Lars Reuther `reuther@os.inf.tu-dresden.de`, Jork Loeser `jork@os.inf.tu-dresden.de`

Definition in file atomic.h.

16.406 atomic.h

```
/*****************************************************************************/
/*
 * (c) 2000-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

/*****************************************************************************/
#ifndef __L4UTIL__INCLUDE__ATOMIC_H__
#define __L4UTIL__INCLUDE__ATOMIC_H__

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

/*****************************************************************************
 *** Prototypes
 *****************************************************************************/

EXTERN_C_BEGIN

L4_INLINE int
l4util_cmpxchg64(volatile l4_uint64_t * dest,
                 l4_uint64_t cmp_val, l4_uint64_t new_val);

L4_INLINE int
l4util_cmpxchg32(volatile l4_uint32_t * dest,
                 l4_uint32_t cmp_val, l4_uint32_t new_val);

L4_INLINE int
l4util_cmpxchg16(volatile l4_uint16_t * dest,
                 l4_uint16_t cmp_val, l4_uint16_t new_val);

L4_INLINE int
l4util_cmpxchg8(volatile l4_uint8_t * dest,
                l4_uint8_t cmp_val, l4_uint8_t new_val);

L4_INLINE int
l4util_cmpxchg(volatile l4_umword_t * dest,
               l4_umword_t cmp_val, l4_umword_t new_val);

L4_INLINE l4_uint32_t
l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val);

L4_INLINE l4_uint16_t
l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val);

L4_INLINE l4_uint8_t
l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val);

L4_INLINE l4_umword_t
l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val);


L4_INLINE void
l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val);
L4_INLINE void
l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val);
L4_INLINE void
l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val);
L4_INLINE void
l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val);
L4_INLINE void
l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val);
L4_INLINE void
l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val);
L4_INLINE void
l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val);
L4_INLINE void
l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val);
L4_INLINE void
l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val);
L4_INLINE void
l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val);
L4_INLINE void
l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val);
L4_INLINE void
l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val);


L4_INLINE l4_uint8_t
l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
L4_INLINE l4_uint16_t
l4util_add16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
```

```
L4_INLINE l4_uint32_t
l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
L4_INLINE l4_uint8_t
l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
L4_INLINE l4_uint16_t
l4util_sub16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
L4_INLINE l4_uint32_t
l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
L4_INLINE l4_uint8_t
l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
L4_INLINE l4_uint16_t
l4util_and16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
L4_INLINE l4_uint32_t
l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
L4_INLINE l4_uint8_t
l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
L4_INLINE l4_uint16_t
l4util_or16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
L4_INLINE l4_uint32_t
l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val);


L4_INLINE void
l4util_inc8(volatile l4_uint8_t *dest);
L4_INLINE void
l4util_inc16(volatile l4_uint16_t *dest);
L4_INLINE void
l4util_inc32(volatile l4_uint32_t *dest);
L4_INLINE void
l4util_dec8(volatile l4_uint8_t *dest);
L4_INLINE void
l4util_dec16(volatile l4_uint16_t *dest);
L4_INLINE void
l4util_dec32(volatile l4_uint32_t *dest);


L4_INLINE l4_uint8_t
l4util_inc8_res(volatile l4_uint8_t *dest);
L4_INLINE l4_uint16_t
l4util_inc16_res(volatile l4_uint16_t *dest);
L4_INLINE l4_uint32_t
l4util_inc32_res(volatile l4_uint32_t *dest);
L4_INLINE l4_uint8_t
l4util_dec8_res(volatile l4_uint8_t *dest);
L4_INLINE l4_uint16_t
l4util_dec16_res(volatile l4_uint16_t *dest);
L4_INLINE l4_uint32_t
l4util_dec32_res(volatile l4_uint32_t *dest);

L4_INLINE void
l4util_atomic_add(volatile long *dest, long val);

L4_INLINE void
l4util_atomic_inc(volatile long *dest);

EXTERN_C_END

/*****************
 * IMPLEMENTAION *
 *****************/

L4_INLINE int
l4util_cmpxchg64(volatile l4_uint64_t * dest,
                 l4_uint64_t cmp_val, l4_uint64_t new_val)
{
  return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
                                     __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
}

L4_INLINE int
l4util_cmpxchg32(volatile l4_uint32_t * dest,
                 l4_uint32_t cmp_val, l4_uint32_t new_val)
{
  return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
                                     __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
}

L4_INLINE int
l4util_cmpxchg16(volatile l4_uint16_t * dest,
                 l4_uint16_t cmp_val, l4_uint16_t new_val)
{
  return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
                                     __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
}

L4_INLINE int
l4util_cmpxchg8(volatile l4_uint8_t * dest,
```

```
                 l4_uint8_t cmp_val, l4_uint8_t new_val)
{
  return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
                                     __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
}

L4_INLINE int
l4util_cmpxchg(volatile l4_umword_t * dest,
               l4_umword_t cmp_val, l4_umword_t new_val)
{
  return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
                                     __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
}

L4_INLINE l4_uint32_t
l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val)
{
  return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
}

L4_INLINE l4_uint16_t
l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val)
{
  return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
}

L4_INLINE l4_uint8_t
l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val)
{
  return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
}

L4_INLINE l4_umword_t
l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val)
{
  return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
}

L4_INLINE void
l4util_inc8(volatile l4_uint8_t *dest)
{ __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_inc16(volatile l4_uint16_t *dest)
{ __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_inc32(volatile l4_uint32_t *dest)
{ __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_atomic_inc(volatile long *dest)
{ __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_dec8(volatile l4_uint8_t *dest)
{ __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_dec16(volatile l4_uint16_t *dest)
{ __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_dec32(volatile l4_uint32_t *dest)
{ __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }


L4_INLINE l4_uint8_t
l4util_inc8_res(volatile l4_uint8_t *dest)
{ return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint16_t
l4util_inc16_res(volatile l4_uint16_t *dest)
{ return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint32_t
l4util_inc32_res(volatile l4_uint32_t *dest)
{ return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint8_t
l4util_dec8_res(volatile l4_uint8_t *dest)
{ return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint16_t
l4util_dec16_res(volatile l4_uint16_t *dest)
{ return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }

```

```
L4_INLINE l4_uint32_t
l4util_dec32_res(volatile l4_uint32_t *dest)
{ return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }

L4_INLINE l4_umword_t
l4util_dec_res(volatile l4_umword_t *dest)
{ return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val)
{ __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val)
{ __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val)
{ __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_atomic_add(volatile long *dest, long val)
{ __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val)
{ __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val)
{ __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val)
{ __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val)
{ __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val)
{ __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val)
{ __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val)
{ __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val)
{ __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE void
l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val)
{ __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint8_t
l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
{ return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint16_t
l4util_add16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
{ return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint32_t
l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
{ return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint8_t
l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
{ return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint16_t
l4util_sub16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
{ return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint32_t
l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
{ return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint8_t
l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
{ return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
```

```
L4_INLINE l4_uint16_t
l4util_and16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
{ return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint32_t
l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
{ return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint8_t
l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
{ return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint16_t
l4util_or16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
{ return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }

L4_INLINE l4_uint32_t
l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
{ return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }

#endif /* ! __L4UTIL__INCLUDE__ATOMIC_H__ */
```

16.407 arm/l4/sys/atomic.h File Reference

Atomic memory modifications.

`#include <l4/sys/compiler.h>`

Include dependency graph for atomic.h:

16.407.1 Detailed Description

Atomic memory modifications.

Definition in file atomic.h.

16.408 atomic.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/compiler.h>
00027
00028 EXTERN_C long int
00029 l4_atomic_add(volatile long int* mem, long int offset) L4_NOTHROW L4_LONG_CALL;
00030
00031 EXTERN_C long int
00032 l4_atomic_xchg(volatile long int* mem, long int newval) L4_NOTHROW L4_LONG_CALL;
00033
00034 EXTERN_C long int
00035 l4_atomic_cmpxchg(volatile long int* mem, long int oldval, long int newval) L4_NOTHROW L4_LONG_CALL;
```

16.409 l4/cxx/atomic.h File Reference

Atomic template.

`#include <l4/util/atomic.h>`

Include dependency graph for atomic.h:

Namespaces

- L4

  L4 low-level kernel interface.

16.409.1 Detailed Description

Atomic template.

Definition in file atomic.h.

16.410 atomic.h

```
00001
00005 /*
00006  * (c) 2004-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction.  Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License.  This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023 #pragma once
00024
00025 #include <l4/util/atomic.h>
00026
00027 extern "C" void  ____error_compare_and_swap_does_not_support_3_bytes____();
00028 extern "C" void  ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00029
00030 namespace L4
00031 {
00032   template< typename X >
00033   inline int compare_and_swap(X volatile *dst, X old_val, X new_val)
00034   {
00035     switch (sizeof(X))
00036       {
00037       case 1:
00038   return l4util_cmpxchg8((l4_uint8_t volatile*)dst, old_val, new_val);
00039       case 2:
00040   return l4util_cmpxchg16((l4_uint16_t volatile *)dst, old_val, new_val);
00041       case 3: ____error_compare_and_swap_does_not_support_3_bytes____();
00042       case 4:
00043   return l4util_cmpxchg32((l4_uint32_t volatile*)dst, old_val, new_val);
00044       default:
00045         ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00046       }
00047     return 0;
00048   }
00049 }
```

16.411 l4/util/backtrace.h File Reference

Backtrace.

```
#include <sys/cdefs.h>
```

Include dependency graph for backtrace.h:

This graph shows which files directly or indirectly include this file:

Functions

- int l4util_backtrace (void ∗∗pc_array, int max_len)

  Fill backtrace structure.

16.411.1 Detailed Description

Backtrace.

Definition in file backtrace.h.

16.411.2 Function Documentation

16.411.2.1 l4util_backtrace()

```
int l4util_backtrace (
            void ** pc_array,
            int max_len )
```

Fill backtrace structure.

Parameters

pc_array	Array of instruction pointers.
max_len	Length of array.

Returns

Number of entries

16.412 backtrace.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #pragma once
00014
00015 #include <sys/cdefs.h>
00016
00017 __BEGIN_DECLS
00018
00026 int l4util_backtrace(void **pc_array, int max_len);
00027
00028 __END_DECLS
```

16.413 l4/util/base64.h File Reference

base 64 encoding and decoding functions adapted from Bob Trower 08/04/01

```
#include <l4/sys/compiler.h>
```

Include dependency graph for base64.h:

Functions

- void base64_encode (const char *infile, unsigned int in_size, char **outfile)

  base-64-encode string infile
- void base64_decode (const char *infile, unsigned int in_size, char **outfile)

  decode base-64-encoded string infile

16.413.1 Detailed Description

base 64 encoding and decoding functions adapted from Bob Trower 08/04/01

Date

04/26/2002

Author

Joerg Nothnagel `jn6@os.inf.tu-dresden.de`

Definition in file base64.h.

16.414 base64.h

```
00001
00010 /*
00011  * (c) 2008-2009 Author(s)
00012  *     economic rights: Technische Universität Dresden (Germany)
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU Lesser General Public License 2.1.
00015  * Please see the COPYING-LGPL-2.1 file for details.
00016  */
00017
00018 #ifndef B64_EN_DECODE
00019 #define B64_EN_DECODE
00020
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00041 L4_CV void base64_encode( const char *infile, unsigned int in_size, char **outfile);
00042
00053 L4_CV void base64_decode(const char *infile, unsigned int in_size, char **outfile);
00054
00055 EXTERN_C_END
00056
00058 #endif //B64_EN_DECODE
```

16.415 l4/util/bitops.h File Reference

bit manipulation functions

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/util/bitops_arch.h>
#include <l4/util/atomic.h>
```

Include dependency graph for bitops.h:

This graph shows which files directly or indirectly include this file:

Macros

- #define l4util_test_and_clear_bit(b, dest) l4util_btr(b, dest)

  define some more usual names

Functions

- void l4util_set_bit (int b, volatile l4_umword_t *dest)

  Set bit in memory.
- void l4util_clear_bit (int b, volatile l4_umword_t *dest)

  Clear bit in memory.
- void l4util_complement_bit (int b, volatile l4_umword_t *dest)

  Complement bit in memory.
- int l4util_test_bit (int b, const volatile l4_umword_t *dest)

  Test bit (return value of bit)
- int l4util_bts (int b, volatile l4_umword_t *dest)

  Bit test and set.
- int l4util_btr (int b, volatile l4_umword_t *dest)

  Bit test and reset.
- int l4util_btc (int b, volatile l4_umword_t *dest)

  Bit test and complement.
- int l4util_bsr (l4_umword_t word)

  Bit scan reverse.
- int l4util_bsf (l4_umword_t word)

  Bit scan forward.
- int l4util_find_first_set_bit (const void *dest, l4_size_t size)

  Find the first set bit in a memory region.
- int l4util_find_first_zero_bit (const void *dest, l4_size_t size)

  Find the first zero bit in a memory region.
- int l4util_next_power2 (unsigned long val)

  Find the next power of 2 for a given number.

16.415.1 Detailed Description

bit manipulation functions

Date

07/03/2001

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file bitops.h.

16.416 bitops.h

```
00001 /*****************************************************************************/
00009 /*
00010  * (c) 2000-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016
00017 /*****************************************************************************/
00018 #ifndef __L4UTIL__INCLUDE__BITOPS_H__
00019 #define __L4UTIL__INCLUDE__BITOPS_H__
00020
00021 /* L4 includes */
00022 #include <l4/sys/l4int.h>
00023 #include <l4/sys/compiler.h>
00024
00026 #define l4util_test_and_clear_bit(b, dest)  l4util_btr(b, dest)
00027 #define l4util_test_and_set_bit(b, dest)  l4util_bts(b, dest)
00028 #define l4util_test_and_change_bit(b, dest) l4util_btc(b, dest)
00029 #define l4util_log2(word)     l4util_bsr(word)
00030
00031 /*****************************************************************************
00032  *** Prototypes
00033  *****************************************************************************/
00034
00035 EXTERN_C_BEGIN
00036
00049 L4_INLINE void
00050 l4util_set_bit(int b, volatile l4_umword_t * dest);
00051
00059 L4_INLINE void
00060 l4util_clear_bit(int b, volatile l4_umword_t * dest);
00061
00069 L4_INLINE void
00070 l4util_complement_bit(int b, volatile l4_umword_t * dest);
00071
00081 L4_INLINE int
00082 l4util_test_bit(int b, const volatile l4_umword_t * dest);
00083
00095 L4_INLINE int
00096 l4util_bts(int b, volatile l4_umword_t * dest);
00097
00109 L4_INLINE int
00110 l4util_btr(int b, volatile l4_umword_t * dest);
00111
00123 L4_INLINE int
00124 l4util_btc(int b, volatile l4_umword_t * dest);
00125
00137 L4_INLINE int
00138 l4util_bsr(l4_umword_t word);
00139
00151 L4_INLINE int
00152 l4util_bsf(l4_umword_t word);
00153
00164 L4_INLINE int
00165 l4util_find_first_set_bit(const void * dest, l4_size_t size);
00166
```

```
L4_INLINE int
l4util_find_first_zero_bit(const void * dest, l4_size_t size);


L4_INLINE int
l4util_next_power2(unsigned long val);

EXTERN_C_END

/*****************************************************************************
 *** Implementation of specific version
 *****************************************************************************/

#include <l4/util/bitops_arch.h>

/*****************************************************************************
 *** Generic implementations
 *****************************************************************************/

#ifndef __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
#include <l4/util/atomic.h>
L4_INLINE void
l4util_set_bit(int b, volatile l4_umword_t * dest)
{
  l4_umword_t oldval, newval;

  dest += b / (sizeof(*dest) * 8); /* advance dest to the proper element */
  b    &= sizeof(*dest) * 8 - 1;   /* modulo; cut off all upper bits */

  do
    {
      oldval = *dest;
      newval = oldval | (1UL << b);
    }
  while (!l4util_cmpxchg(dest, oldval, newval));
}
#endif

#ifndef __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
#include <l4/util/atomic.h>
L4_INLINE void
l4util_clear_bit(int b, volatile l4_umword_t * dest)
{
  l4_umword_t oldval, newval;

  dest += b / (sizeof(*dest) * 8);
  b    &= sizeof(*dest) * 8 - 1;

  do
    {
      oldval = *dest;
      newval = oldval & ~(1UL << b);
    }
  while (!l4util_cmpxchg(dest, oldval, newval));
}
#endif

#ifndef __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
L4_INLINE int
l4util_test_bit(int b, const volatile l4_umword_t * dest)
{
  dest += b / (sizeof(*dest) * 8);
  b    &= sizeof(*dest) * 8 - 1;

  return (*dest >> b) & 1;
}
#endif

#ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
#include <l4/util/atomic.h>
L4_INLINE int
l4util_bts(int b, volatile l4_umword_t * dest)
{
  l4_umword_t oldval, newval;

  dest += b / (sizeof(*dest) * 8);
  b    &= sizeof(*dest) * 8 - 1;

  do
    {
      oldval = *dest;
      newval = oldval | (1UL << b);
    }
  while (!l4util_cmpxchg(dest, oldval, newval));

  /* Return old bit */
  return (oldval >> b) & 1;
```

```
}
#endif

#ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
#include <l4/util/atomic.h>
L4_INLINE int
l4util_btr(int b, volatile l4_umword_t * dest)
{
  l4_umword_t oldval, newval;

  dest += b / (sizeof(*dest) * 8);
  b    &= sizeof(*dest) * 8 - 1;

  do
    {
      oldval = *dest;
      newval = oldval & ~(1UL << b);
    }
  while (!l4util_cmpxchg(dest, oldval, newval));

  /* Return old bit */
  return (oldval >> b) & 1;
}
#endif

#ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
L4_INLINE int
l4util_bsr(l4_umword_t word)
{
  int i;

  if (!word)
    return -1;

  for (i = 8 * sizeof(word) - 1; i >= 0; i--)
    if ((1UL << i) & word)
      return i;

  return -1;
}
#endif

#ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
L4_INLINE int
l4util_bsf(l4_umword_t word)
{
  unsigned int i;

  if (!word)
    return -1;

  for (i = 0; i < sizeof(word) * 8; i++)
    if ((1UL << i) & word)
      return i;

  return -1;
}
#endif

#ifndef __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
L4_INLINE int
l4util_find_first_zero_bit(const void * dest, l4_size_t size)
{
  l4_size_t i, j;
  unsigned long *v = (unsigned long*)dest;

  if (!size)
    return 0;

  size = (size + 31) & ~0x1f; /* Grmbl: adapt to x86 implementation... */

  for (i = j = 0; i < size; i++, j++)
    {
      if (j >= sizeof(*v) * 8)
  {
    j = 0;
    v++;
  }
      if (!((1UL << j) & *v))
  return i;
    }
  return size + 1;
}
#endif

#ifndef __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
L4_INLINE void
```

```
l4util_complement_bit(int b, volatile l4_umword_t * dest)
{
  dest += b / (sizeof(*dest) * 8);
  b    &= sizeof(*dest) * 8 - 1;

  *dest ^= 1UL « b;
}
#endif

/*
 * Adapted from:
 * http://en.wikipedia.org/wiki/Power_of_two#Algorithm_to_find_the_next-highest_power_of_two
 */
L4_INLINE int
l4util_next_power2(unsigned long val)
{
  unsigned i;

  if (val == 0)
    return 1;

  val--;
  for (i=1; i < sizeof(unsigned long)*8; i«=1)
    val = val | val » i;

  return val+1;
}


/* Non-implemented version, catch with a linker warning */

extern int __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(void);

#ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
L4_INLINE int
l4util_btc(int b, volatile l4_umword_t * dest)
{ (void)b; (void)dest; __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(); return 0; }
#endif

#ifndef  __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
L4_INLINE int
l4util_find_first_set_bit(const void * dest, l4_size_t size)
{ (void)dest; (void)size; __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(); return 0; }
#endif

#endif /* ! __L4UTIL__INCLUDE__BITOPS_H__ */
```

16.417 l4/util/elf.h File Reference

ELF definition.

`#include <l4/sys/l4int.h>`

Include dependency graph for elf.h:

Data Structures

- struct Elf32_Ehdr

  ELF32 header.
- struct Elf64_Ehdr

  ELF64 header.
- struct Elf32_Shdr

  ELF32 section header.
- struct Elf64_Shdr

  ELF64 section header.
- struct Elf32_Phdr

  ELF32 program header.
- struct Elf64_Phdr

  ELF64 program header.
- struct Elf32_Dyn

  ELF32 dynamic entry.
- struct Elf64_Dyn

  ELF64 dynamic entry.
- struct Elf32_Rel

  ELF32 relocation entry w/o addend.
- struct Elf32_Rela

  ELF32 relocation entry w/ addend.
- struct Elf64_Rel

  ELF64 relocation entry w/o addend.
- struct Elf64_Rela

  ELF64 relocation entry w/ addend.
- struct Elf32_Sym

  ELF32 symbol table entry.
- struct Elf64_Sym

  ELF64 symbol table entry.
- struct Elf32_Auxv

  Auxiliary vector (32-bit).
- struct Elf64_Auxv

  Auxiliary vector (64-bit).

Macros

- #define ElfW(type) _ElfW(Elf, 32, type)

  Use 64 or 32 bits types depending on the target architecture.
- #define ELF32_R_SYM(i) ((i)>>8)

  Symbol table index.
- #define ELF32_R_TYPE(i) ((unsigned char)(i))
- #define ELF32_R_INFO(s, t) (((s)<<8)+(unsigned char)(t))

  Create info from symbol table index + type.
- #define ELF64_R_SYM(i) ((i)>>32)

  Symbol table index.
- #define ELF64_R_TYPE(i) ((i)&0xffffffffL)
- #define ELF64_R_INFO(s, t) (((s)<<32)+(t)&0xffffffffL)

  Create info from symbol table index + type.
- #define ELF32_ST_BIND(i) ((i)>>4)

- #define ELF32_ST_TYPE(i) ((i)&0xf)
- #define ELF32_ST_INFO(b, t) (((b)<<4)+((t)&0xf))

  Make info from bind + type.
- #define ELF64_ST_BIND(i) ((i)>>4)
- #define ELF64_ST_TYPE(i) ((i)&0xf)
- #define ELF64_ST_INFO(b, t) (((b)<<4)+((t)&0xf))

  Make info from bind + type.

Typedefs

- typedef struct Elf32_Auxv Elf32_Auxv

  Auxiliary vector (32-bit).
- typedef struct Elf64_Auxv Elf64_Auxv

  Auxiliary vector (64-bit).

ELF types

- typedef l4_uint32_t Elf32_Addr

  size 4 align 4
- typedef l4_uint32_t Elf32_Off

  size 4 align 4
- typedef l4_uint16_t Elf32_Half

  size 2 align 2
- typedef l4_uint32_t Elf32_Word

  size 4 align 4
- typedef l4_int32_t Elf32_Sword

  size 4 align 4
- typedef l4_uint64_t Elf64_Addr

  size 8 align 8
- typedef l4_uint64_t Elf64_Off

  size 8 align 8
- typedef l4_uint16_t Elf64_Half

  size 2 align 2
- typedef l4_uint32_t Elf64_Word

  size 4 align 4
- typedef l4_int32_t Elf64_Sword

  size 4 align 4
- typedef l4_uint64_t Elf64_Xword

  size 8 align 8
- typedef l4_int64_t Elf64_Sxword

  size 8 align 8

Enumerations

- enum { EI_NIDENT = 16 }
- enum Elf_ETs {
  ET_NONE = 0 , ET_REL = 1 , ET_EXEC = 2 , ET_DYN = 3 ,
  ET_CORE = 4 , ET_LOPROC = 0xff00 , ET_HIPROC = 0xffff }

  Object file type.

- enum Elf_EMs {
  EM_NONE = 0 , EM_M32 = 1 , EM_SPARC = 2 , EM_386 = 3 ,
  EM_68K = 4 , EM_88K = 5 , EM_860 = 7 , EM_MIPS = 8 ,
  EM_MIPS_RS4_BE = 10 , EM_SPARC64 = 11 , EM_PARISC = 15 , EM_VPP500 = 17 ,
  EM_SPARC32PLUS = 18 , EM_960 = 19 , EM_PPC = 20 , EM_V800 = 36 ,
  EM_FR20 = 37 , EM_RH32 = 38 , EM_RCE = 39 , EM_ARM = 40 ,
  EM_ALPHA = 41 , EM_SH = 42 , EM_SPARCV9 = 43 , EM_TRICORE = 44 ,
  EM_ARC = 45 , EM_H8_300 = 46 , EM_H8_300H = 47 , EM_H8S = 48 ,
  EM_H8_500 = 49 , EM_IA_64 = 50 , EM_MIPS_X = 51 , EM_COLDFIRE = 52 ,
  EM_68HC12 = 53 , EM_X86_64 = 62 , EM_PDSP = 63 , EM_FX66 = 66 ,
  EM_ST9PLUS = 67 , EM_ST7 = 68 , EM_68HC16 = 69 , EM_68HC11 = 70 ,
  EM_68HC08 = 71 , EM_68HC05 = 72 , EM_SVX = 73 , EM_ST19 = 74 ,
  EM_VAX = 75 , EM_CRIS = 76 , EM_JAVELIN = 77 , EM_FIREPATH = 78 ,
  EM_ZSP = 79 , EM_MMIX = 80 , EM_HUANY = 81 , EM_PRISM = 82 ,
  EM_AVR = 83 , EM_FR30 = 84 , EM_D10V = 85 , EM_D30V = 86 ,
  EM_V850 = 87 , EM_M32R = 88 , EM_MN10300 = 89 , EM_MN10200 = 90 ,
  EM_PJ = 91 , EM_OPENRISC = 92 , EM_ARC_A5 = 93 , EM_XTENSA = 94 ,
  EM_ALTERA_NIOS2 = 113 , EM_AARCH64 = 183 , EM_TILEPRO = 188 , EM_MICROBLAZE = 189 ,
  EM_TILEGX = 191 , **EM_NUM** = 192 }

  Required architecture.
- enum Elf_EVs { EV_NONE = 0 , EV_CURRENT = 1 }

  Object file version.
- enum Elf_EIs {
  EI_MAG0 = 0 , EI_MAG1 = 1 , EI_MAG2 = 2 , EI_MAG3 = 3 ,
  EI_CLASS = 4 , EI_DATA = 5 , EI_VERSION = 6 , EI_OSABI = 7 ,
  EI_ABIVERSION = 8 , EI_PAD = 9 }

  Identification Indices.
- enum Elf_MAGs { ELFMAG0 = 0x7f , ELFMAG1 = 'E' , ELFMAG2 = 'L' , ELFMAG3 = 'F' }

  Magic number.
- enum Elf_CIASSs { ELFCLASSNONE = 0 , ELFCLASS32 = 1 , ELFCLASS64 = 2 , ELFCLASSNUM = 3 }

  File class or capacity.
- enum Elf_DATAs { ELFDATANONE = 0 , ELFDATA2LSB = 1 , ELFDATA2MSB = 2 , ELFDATANUM = 3 }

  Data encoding.
- enum Elf_OSABIs {
  ELFOSABI_NONE = 0 , ELFOSABI_SYSV = 0 , ELFOSABI_HPUX = 1 , ELFOSABI_NETBSD = 2 ,
  ELFOSABI_LINUX = 3 , ELFOSABI_SOLARIS = 6 , ELFOSABI_AIX = 7 , ELFOSABI_IRIX = 8 ,
  ELFOSABI_FREEBSD = 9 , ELFOSABI_TRU64 = 10 , ELFOSABI_MODESTO = 11 , ELFOSABI_OPENBSD
  = 12 ,
  ELFOSABI_ARM = 97 , ELFOSABI_STANDALONE = 255 }

  Identify operating system and ABI to which the object is targeted.
- enum Elf_SHNs {
  SHN_UNDEF = 0 , SHN_LORESERVE = 0xff00 , SHN_LOPROC = 0xff00 , SHN_HIPROC = 0xff1f ,
  SHN_ABS = 0xfff1 , SHN_COMMON = 0xfff2 , SHN_HIRESERVE = 0xffff }

  Special section indexes.
- enum Elf_SHTs {
  SHT_NULL = 0 , SHT_PROGBITS = 1 , SHT_SYMTAB = 2 , SHT_STRTAB = 3 ,
  SHT_RELA = 4 , SHT_HASH = 5 , SHT_DYNAMIC = 6 , SHT_NOTE = 7 ,
  SHT_NOBITS = 8 , SHT_REL = 9 , SHT_SHLIB = 10 , SHT_DYNSYM = 11 ,
  SHT_INIT_ARRAY = 14 , SHT_FINI_ARRAY = 15 , SHT_PREINIT_ARRAY = 16 , SHT_GROUP = 17 ,
  SHT_SYMTAB_SHNDX = 18 , SHT_NUM = 19 , SHT_LOOS = 0x60000000 , SHT_HIOS = 0x6fffffff ,
  SHT_LOPROC = 0x70000000 , SHT_HIPROC = 0x7fffffff , SHT_LOUSER = 0x80000000 , SHT_HIUSER =
  0xffffffff }

  Section type.
- enum Elf_SHFs {
  SHF_WRITE = 0x1 , SHF_ALLOC = 0x2 , SHF_EXECINSTR = 0x4 , SHF_MERGE = 0x10 ,
  SHF_STRINGS = 0x20 , SHF_INFO_LINK = 0x40 , SHF_LINK_ORDER = 0x80 , SHF_OS_NONCONFORMING

= 0x100 ,
SHF_GROUP = 0x200 , SHF_TLS = 0x400 , SHF_MASKOS = 0x0ff00000 , SHF_MASKPROC = 0xf0000000 }

Section attribute flags.

- enum Elf_PTs {
PT_NULL = 0 , PT_LOAD = 1 , PT_DYNAMIC = 2 , PT_INTERP = 3 ,
PT_NOTE = 4 , PT_SHLIB = 5 , PT_PHDR = 6 , PT_TLS = 7 ,
PT_NUM = 8 , PT_LOOS = 0x60000000 , PT_HIOS = 0x6fffffff , PT_LOPROC = 0x70000000 ,
PT_HIPROC = 0x7fffffff , PT_GNU_EH_FRAME = PT_LOOS + 0x474e550 , PT_GNU_STACK = PT_LOOS + 0x474e551 , PT_GNU_RELRO = PT_LOOS + 0x474e552 ,
PT_L4_STACK = PT_LOOS + 0x12 , PT_L4_KIP = PT_LOOS + 0x13 , PT_L4_AUX = PT_LOOS + 0x14 }

  Segment types.

- enum ELF_PFs {
PF_X = 0x1 , PF_W = 0x2 , PF_R = 0x4 , PF_MASKOS = 0x0ff00000 ,
PF_MASKPROC = 0x7fffffff }

  Segment permissions.

- enum Elf_NTs_core {
NT_PRSTATUS = 1 , NT_FPREGSET = 2 , NT_PRPSINFO = 3 , NT_PRXREG = 4 ,
NT_TASKSTRUCT = 4 , NT_PLATFORM = 5 , NT_AUXV = 6 , NT_GWINDOWS = 7 ,
NT_ASRS = 8 , NT_PSTATUS = 10 , NT_PSINFO = 13 , NT_PRCRED = 14 ,
NT_UTSNAME = 15 , NT_LWPSTATUS = 16 , NT_LWPSINFO = 17 , NT_PRFPXREG = 20 }

  Legal values for note segment descriptor types for core files.

- enum Elf_NTs_obj { NT_VERSION = 1 }

  Legal values for the note segment descriptor types for object files.

- enum Elf_DTs {
DT_NULL = 0 , DT_NEEDED = 1 , DT_PLTRELSZ = 2 , DT_PLTGOT = 3 ,
DT_HASH = 4 , DT_STRTAB = 5 , DT_SYMTAB = 6 , DT_RELA = 7 ,
DT_RELASZ = 8 , DT_RELAENT = 9 , DT_STRSZ = 10 , DT_SYMENT = 11 ,
DT_INIT = 12 , DT_FINI = 13 , DT_SONAME = 14 , DT_RPATH = 15 ,
DT_SYMBOLIC = 16 , DT_REL = 17 , DT_RELSZ = 18 , DT_RELENT = 19 ,
DT_PTRREL = 20 , DT_DEBUG = 21 , DT_TEXTREL = 22 , DT_JMPREL = 23 ,
DT_BIND_NOW = 24 , DT_INIT_ARRAY = 25 , DT_FINI_ARRAY = 26 , DT_INIT_ARRAYSZ = 27 ,
DT_FINI_ARRAYSZ = 28 , DT_RUNPATH = 29 , DT_FLAGS = 30 , DT_ENCODING = 32 ,
DT_PREINIT_ARRAY = 32 , DT_PREINIT_ARRAYSZ = 33 , DT_NUM = 34 , DT_LOOS = 0x6000000d ,
DT_HIOS = 0x6ffff000 , DT_LOPROC = 0x70000000 , DT_HIPROC = 0x7fffffff }

  Dynamic Array Tags.

- enum Elf_DFs {
DF_ORIGIN = 0x00000001 , DF_SYMBOLIC = 0x00000002 , DF_TEXTREL = 0x00000004 ,
DF_BIND_NOW = 0x00000008 ,
DF_STATIC_TLS = 0x00000010 }

  Values of Elf32_Dyn.d_un.d_val, Elf64_Dyn.d_un.d_val in the DT_FLAGS entry.

- enum Elf_DF_1s {
DF_1_NOW = 0x00000001 , DF_1_GLOBAL = 0x00000002 , DF_1_GROUP = 0x00000004 ,
DF_1_NODELETE = 0x00000008 ,
DF_1_LOADFLTR = 0x00000010 , DF_1_INITFIRST = 0x00000020 , DF_1_NOOPEN = 0x00000040 ,
DF_1_ORIGIN = 0x00000080 ,
DF_1_DIRECT = 0x00000100 , **DF_1_TRANS** = 0x00000200 , DF_1_INTERPOSE = 0x00000400 ,
DF_1_NODEFLIB = 0x00000800 ,
DF_1_NODUMP = 0x00001000 , DF_1_CONFALT = 0x00002000 , DF_1_ENDFILTEE = 0x00004000 ,
DF_1_DISPRELDNE = 0x00008000 ,
DF_1_DISPRELPND = 0x00010000 }

  State flags selectable in the Elf32_Dyn.d_un.d_val / Elf64_Dyn.d_un.d_val element of the DT_FLAGS_1 entry in the dynamic section.

- enum Elf_DTF_1s

  Flags for the feature selection in DT_FEATURE_1.

- enum Elf_DF_P1s { DF_P1_LAZYLOAD = 0x00000001 , DF_P1_GROUPPERM = 0x00000002 }

Flags in the DT_POSFLAG_1 entry effecting only the next DT_∗ entry.

- enum Elf_R_386_s {
  R_386_NONE = 0 , R_386_32 = 1 , R_386_PC32 = 2 , R_386_GOT32 = 3 ,
  R_386_PLT32 = 4 , R_386_COPY = 5 , R_386_GLOB_DAT = 6 , R_386_JMP_SLOT = 7 ,
  R_386_RELATIVE = 8 , R_386_GOTOFF = 9 , R_386_GOTPC = 10 , **R_386_32PLT** = 11 ,
  R_386_TLS_TPOFF = 14 , R_386_TLS_IE = 15 , R_386_TLS_GOTIE = 16 , R_386_TLS_LE = 17 ,
  R_386_TLS_GD = 18 , R_386_TLS_LDM = 19 , **R_386_16** = 20 , **R_386_PC16** = 21 ,
  R_386_8 = 22 , **R_386_PC8** = 23 , R_386_TLS_GD_32 = 24 , R_386_TLS_GD_PUSH = 25 ,
  R_386_TLS_GD_CALL = 26 , R_386_TLS_GD_POP = 27 , R_386_TLS_LDM_32 = 28 , R_386_TLS_LDM_PUSH = 29 ,
  R_386_TLS_LDM_CALL = 30 , R_386_TLS_LDM_POP = 31 , R_386_TLS_LDO_32 = 32 , R_386_TLS_IE_32 = 33 ,
  R_386_TLS_LE_32 = 34 , R_386_TLS_DTPMOD32 = 35 , R_386_TLS_DTPOFF32 = 36 , R_386_TLS_TPOFF32 = 37 ,
  R_386_NUM = 38 }

  Relocation types (processor specific).
- enum Elf_EF_ARM_s { }

  ARM specific declarations.
- enum Elf_STT_ARM_s

  Additional symbol types for Thumb.
- enum Elf_SHF_s_ARM { SHF_ARM_ENTRYSECT = 0x10000000 , SHF_ARM_COMDEF = 0x80000000 }

  ARM-specific values for Elf32_Shdr.sh_flags / Elf64_Shdr.sh_flags.
- enum Elf_ARM_SBs { PF_ARM_SB = 0x10000000 }

  ARM-specific program header flags.
- enum Elf_R_ARM_s {
  R_ARM_NONE = 0 , R_ARM_PC24 = 1 , R_ARM_ABS32 = 2 , R_ARM_REL32 = 3 ,
  R_ARM_PC13 = 4 , R_ARM_ABS16 = 5 , R_ARM_ABS12 = 6 , **R_ARM_THM_ABS5** = 7 ,
  R_ARM_ABS8 = 8 , **R_ARM_SBREL32** = 9 , **R_ARM_THM_PC22** = 10 , **R_ARM_THM_PC8** = 11 ,
  R_ARM_AMP_VCALL9 = 12 , **R_ARM_SWI24** = 13 , **R_ARM_THM_SWI8** = 14 , **R_ARM_XPC25** = 15 ,
  R_ARM_THM_XPC22 = 16 , R_ARM_COPY = 20 , R_ARM_GLOB_DAT = 21 , R_ARM_JUMP_SLOT = 22 ,
  R_ARM_RELATIVE = 23 , R_ARM_GOTOFF = 24 , R_ARM_GOTPC = 25 , R_ARM_GOT32 = 26 ,
  R_ARM_PLT32 = 27 , **R_ARM_ALU_PCREL_7_0** = 32 , **R_ARM_ALU_PCREL_15_8** = 33 , **R_ARM_↩ALU_PCREL_23_15** = 34 ,
  R_ARM_LDR_SBREL_11_0 = 35 , **R_ARM_ALU_SBREL_19_12** = 36 , **R_ARM_ALU_SBREL_27_20** = 37 , **R_ARM_GNU_VTENTRY** = 100 ,
  R_ARM_GNU_VTINHERIT = 101 , R_ARM_THM_PC11 = 102 , R_ARM_THM_PC9 = 103 , **R_ARM_↩RXPC25** = 249 ,
  R_ARM_RSBREL32 = 250 , **R_ARM_THM_RPC22** = 251 , **R_ARM_RREL32** = 252 , **R_ARM_RABS22** = 253 ,
  R_ARM_RPC24 = 254 , **R_ARM_RBASE** = 255 , R_ARM_NUM = 256 }

  ARM relocations.
- enum Elf_R_AARCH64_s { R_AARCH64_NONE = 0 , **R_AARCH64_RELATIVE** = 1027 }

  AARCH64 relocations.
- enum Elf_R_X86_64_s {
  R_X86_64_NONE = 0 , R_X86_64_64 = 1 , R_X86_64_PC32 = 2 , R_X86_64_GOT32 = 3 ,
  R_X86_64_PLT32 = 4 , R_X86_64_COPY = 5 , R_X86_64_GLOB_DAT = 6 , R_X86_64_JUMP_SLOT = 7 ,
  R_X86_64_RELATIVE = 8 , R_X86_64_GOTPCREL = 9 , R_X86_64_32 = 10 , R_X86_64_32S = 11 ,
  R_X86_64_16 = 12 , R_X86_64_PC16 = 13 , R_X86_64_8 = 14 , R_X86_64_PC8 = 15 ,
  R_X86_64_DTPMOD64 = 16 , R_X86_64_DTPOFF64 = 17 , R_X86_64_TPOFF64 = 18 , R_X86_64_TLSGD = 19 ,
  R_X86_64_TLSLD = 20 , R_X86_64_DTPOFF32 = 21 , R_X86_64_GOTTPOFF = 22 , R_X86_64_TPOFF32 = 23 ,
  R_X86_64_NUM = 24 }

  AMD x86-64 relocations.
- enum Elf_STNs

  Symbol Table Entry.

- enum Elf_STBs {
  STB_LOCAL = 0 , STB_GLOBAL = 1 , STB_WEAK = 2 , STB_LOOS = 10 ,
  STB_HIOS = 12 , STB_LOPROC = 13 , STB_HIPROC = 15 }

  Symbol Binding.
- enum Elf_STTs {
  STT_NOTYPE = 0 , STT_OBJECT = 1 , STT_FUNC = 2 , STT_SECTION = 3 ,
  STT_FILE = 4 , STT_LOOS = 10 , STT_HIOS = 12 , STT_LOPROC = 13 ,
  STT_HIPROC = 15 }

  Symbol Types.
- enum Elf_ATs {
  AT_NULL = 0 , AT_IGNORE = 1 , AT_EXECFD = 2 , AT_PHDR = 3 ,
  AT_PHENT = 4 , AT_PHNUM = 5 , AT_PAGESZ = 6 , AT_BASE = 7 ,
  AT_FLAGS = 8 , AT_ENTRY = 9 , AT_NOTELF = 10 , AT_UID = 11 ,
  AT_EUID = 12 , AT_GID = 13 , AT_EGID = 14 , AT_L4_AUX = 0xf0 ,
  AT_L4_ENV = 0xf1 }

  Legal values for Elf32_Auxv.atype / Elf64_Auxv.atype.

16.417.1 Detailed Description

ELF definition.

Date

08/18/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de Alexander Warg aw11@os.inf.tu-dresden.↩de

Many structs from "Executable and Linkable Format (ELF)", Portable Formats Specification, Version 1.1 and "↩System V Application Binary Interface - DRAFT - April 29, 1998" The Santa Cruz Operation, Inc. (see http↩://www.sco.com/developer/gabi/contents.html)

Definition in file elf.h.

16.418 elf.h

```
00001
00019 /*
00020  * (c) 2008-2009 Author(s)
00021  *     economic rights: Technische Universität Dresden (Germany)
00022  * This file is part of TUD:OS and distributed under the terms of the
00023  * GNU Lesser General Public License 2.1.
00024  * Please see the COPYING-LGPL-2.1 file for details.
00025  */
00026
00027 /* (c) 2003-2006 Technische Universitaet Dresden
00028  * This file is part of the exec package, which is distributed under
00029  * the terms of the GNU General Public License 2. Please see the
00030  * COPYING file for details. */
00031
00032 #pragma once
00033
00034 #include <l4/sys/l4int.h>
00035
00046 typedef l4_uint32_t     Elf32_Addr;
00047 typedef l4_uint32_t     Elf32_Off;
00048 typedef l4_uint16_t     Elf32_Half;
00049 typedef l4_uint32_t     Elf32_Word;
```

```
typedef l4_int32_t      Elf32_Sword;
typedef l4_uint64_t     Elf64_Addr;
typedef l4_uint64_t     Elf64_Off;
typedef l4_uint16_t     Elf64_Half;
typedef l4_uint32_t     Elf64_Word;
typedef l4_int32_t      Elf64_Sword;
typedef l4_uint64_t     Elf64_Xword;
typedef l4_int64_t      Elf64_Sxword;
#if L4_MWORD_BITS == 64
# define ElfW(type)      _ElfW(Elf, 64, type)
#else
# define ElfW(type)      _ElfW(Elf, 32, type)
#endif
#define _ElfW(e,w,t)    __ElfW(e, w, _##t)
#define __ElfW(e,w,t)   e##w##t

#if defined(ARCH_x86)
# define L4_ARCH_EI_DATA      ELFDATA2LSB
# define L4_ARCH_E_MACHINE    EM_386
# define L4_ARCH_EI_CLASS     ELFCLASS32
#elif defined(ARCH_amd64)
# define L4_ARCH_EI_DATA      ELFDATA2LSB
# define L4_ARCH_E_MACHINE    EM_X86_64
# define L4_ARCH_EI_CLASS     ELFCLASS64
#elif defined(ARCH_arm)
# define L4_ARCH_EI_DATA      ELFDATA2LSB
# define L4_ARCH_E_MACHINE    EM_ARM
# define L4_ARCH_EI_CLASS     ELFCLASS32
#elif defined(ARCH_arm64)
# define L4_ARCH_EI_DATA      ELFDATA2LSB
# define L4_ARCH_E_MACHINE    EM_AARCH64
# define L4_ARCH_EI_CLASS     ELFCLASS64
#elif defined(ARCH_ppc32)
# define L4_ARCH_EI_DATA      ELFDATA2MSB
# define L4_ARCH_E_MACHINE    EM_PPC
# define L4_ARCH_EI_CLASS     ELFCLASS32
#elif defined(ARCH_sparc)
# define L4_ARCH_EI_DATA      ELFDATA2MSB
# define L4_ARCH_E_MACHINE    EM_SPARC
# define L4_ARCH_EI_CLASS     ELFCLASS32
#elif defined(ARCH_mips)
# define L4_ARCH_EI_DATA      ELFDATA2LSB
# define L4_ARCH_E_MACHINE    EM_MIPS
# ifdef __mips64
#  define L4_ARCH_EI_CLASS     ELFCLASS64
# else
#  define L4_ARCH_EI_CLASS     ELFCLASS32
# endif
#else
# warning elf.h: Unsupported build architecture!
#endif


enum
{
  EI_NIDENT             = 16,
};

typedef struct
{
  unsigned char e_ident[EI_NIDENT];
  Elf32_Half    e_type;
  Elf32_Half    e_machine;
  Elf32_Word    e_version;
  Elf32_Addr    e_entry;
  Elf32_Off     e_phoff;
  Elf32_Off     e_shoff;
  Elf32_Word    e_flags;
  Elf32_Half    e_ehsize;
  Elf32_Half    e_phentsize;
  Elf32_Half    e_phnum;
  Elf32_Half    e_shentsize;
  Elf32_Half    e_shnum;
  Elf32_Half    e_shstrndx;
} Elf32_Ehdr;

typedef struct
{
  unsigned char e_ident[EI_NIDENT];
  Elf64_Half     e_type;
  Elf64_Half     e_machine;
  Elf64_Word     e_version;
  Elf64_Addr     e_entry;
  Elf64_Off      e_phoff;
  Elf64_Off      e_shoff;
  Elf64_Word     e_flags;
  Elf64_Half     e_ehsize;
```

```
  Elf64_Half     e_phentsize;
  Elf64_Half     e_phnum;
  Elf64_Half     e_shentsize;
  Elf64_Half     e_shnum;
  Elf64_Half     e_shstrndx;
} Elf64_Ehdr;

enum Elf_ETs
{
  ET_NONE               = 0,
  ET_REL                = 1,
  ET_EXEC               = 2,
  ET_DYN                = 3,
  ET_CORE               = 4,
  ET_LOPROC             = 0xff00,
  ET_HIPROC             = 0xffff,
};

enum Elf_EMs
{
  EM_NONE               = 0,
  EM_M32                = 1,
  EM_SPARC              = 2,
  EM_386                = 3,
  EM_68K                = 4,
  EM_88K                = 5,
  EM_860                = 7,
  EM_MIPS               = 8,
  EM_MIPS_RS4_BE        = 10,
  EM_SPARC64            = 11,
  EM_PARISC             = 15,
  EM_VPP500             = 17,
  EM_SPARC32PLUS        = 18,
  EM_960                = 19,
  EM_PPC                = 20,
  EM_V800               = 36,
  EM_FR20               = 37,
  EM_RH32               = 38,
  EM_RCE                = 39,
  EM_ARM                = 40,
  EM_ALPHA              = 41,
  EM_SH                 = 42,
  EM_SPARCV9            = 43,
  EM_TRICORE            = 44,
  EM_ARC                = 45,
  EM_H8_300             = 46,
  EM_H8_300H            = 47,
  EM_H8S                = 48,
  EM_H8_500             = 49,
  EM_IA_64              = 50,
  EM_MIPS_X             = 51,
  EM_COLDFIRE           = 52,
  EM_68HC12             = 53,
  EM_X86_64             = 62,
  EM_PDSP               = 63,
  EM_FX66               = 66,
  EM_ST9PLUS            = 67,
  EM_ST7                = 68,
  EM_68HC16             = 69,
  EM_68HC11             = 70,
  EM_68HC08             = 71,
  EM_68HC05             = 72,
  EM_SVX                = 73,
  EM_ST19               = 74,
  EM_VAX                = 75,
  EM_CRIS               = 76,
  EM_JAVELIN            = 77,
  EM_FIREPATH           = 78,
  EM_ZSP                = 79,
  EM_MMIX               = 80,
  EM_HUANY              = 81,
  EM_PRISM              = 82,
  EM_AVR                = 83,
  EM_FR30               = 84,
  EM_D10V               = 85,
  EM_D30V               = 86,
  EM_V850               = 87,
  EM_M32R               = 88,
  EM_MN10300            = 89,
  EM_MN10200            = 90,
  EM_PJ                 = 91,
  EM_OPENRISC           = 92,
  EM_ARC_A5             = 93,
  EM_XTENSA             = 94,
  EM_ALTERA_NIOS2       = 113,
  EM_AARCH64            = 183,
  EM_TILEPRO            = 188,
```

```
  EM_MICROBLAZE          = 189,
  EM_TILEGX              = 191,
  EM_NUM                 = 192,
};

#if 0
#define EM_ALPHA         0x9026  /* interium value used by Linux until the
                                    committee comes up with a final number */
#define EM_S390          0xA390  /* interium value used for IBM S390 */
#endif

enum Elf_EVs
{
  EV_NONE                = 0,
  EV_CURRENT             = 1,
};

enum Elf_EIs
{
  EI_MAG0                = 0,
  EI_MAG1                = 1,
  EI_MAG2                = 2,
  EI_MAG3                = 3,
  EI_CLASS               = 4,
  EI_DATA                = 5,
  EI_VERSION             = 6,
  EI_OSABI               = 7,
  EI_ABIVERSION          = 8,
  EI_PAD                 = 9,
};

enum Elf_MAGs
{
  ELFMAG0                = 0x7f,
  ELFMAG1                = 'E',
  ELFMAG2                = 'L',
  ELFMAG3                = 'F',
};

enum Elf_ClASSs
{
  ELFCLASSNONE           = 0,
  ELFCLASS32             = 1,
  ELFCLASS64             = 2,
  ELFCLASSNUM            = 3,
};

enum Elf_DATAs
{
  ELFDATANONE            = 0,
  ELFDATA2LSB            = 1,
  ELFDATA2MSB            = 2,
  ELFDATANUM             = 3,
};

enum Elf_OSABIs
{
  ELFOSABI_NONE          = 0,
  ELFOSABI_SYSV          = 0,
  ELFOSABI_HPUX          = 1,
  ELFOSABI_NETBSD        = 2,
  ELFOSABI_LINUX         = 3,
  ELFOSABI_SOLARIS       = 6,
  ELFOSABI_AIX           = 7,
  ELFOSABI_IRIX          = 8,
  ELFOSABI_FREEBSD       = 9,
  ELFOSABI_TRU64         = 10,
  ELFOSABI_MODESTO       = 11,
  ELFOSABI_OPENBSD       = 12,
  ELFOSABI_ARM           = 97,
  ELFOSABI_STANDALONE    = 255,
};

enum Elf_SHNs
{
  SHN_UNDEF              = 0,
  SHN_LORESERVE          = 0xff00,
  SHN_LOPROC             = 0xff00,
  SHN_HIPROC             = 0xff1f,
  SHN_ABS                = 0xfff1,
  SHN_COMMON             = 0xfff2,
  SHN_HIRESERVE          = 0xffff,
};

typedef struct
{
  Elf32_Word             sh_name;
```

```
  Elf32_Word            sh_type;
  Elf32_Word            sh_flags;
  Elf32_Addr            sh_addr;
  Elf32_Off             sh_offset;
  Elf32_Word            sh_size;
  Elf32_Word            sh_link;
  Elf32_Word            sh_info;
  Elf32_Word            sh_addralign;
  Elf32_Word            sh_entsize;
} Elf32_Shdr;

typedef struct
{
  Elf64_Word            sh_name;
  Elf64_Word            sh_type;
  Elf64_Xword           sh_flags;
  Elf64_Addr            sh_addr;
  Elf64_Off             sh_offset;
  Elf64_Xword           sh_size;
  Elf64_Word            sh_link;
  Elf64_Word            sh_info;
  Elf64_Xword           sh_addralign;
  Elf64_Xword           sh_entsize;
} Elf64_Shdr;

enum Elf_SHTs
{
  SHT_NULL              = 0,
  SHT_PROGBITS          = 1,
  SHT_SYMTAB            = 2,
  SHT_STRTAB            = 3,
  SHT_RELA              = 4,
  SHT_HASH              = 5,
  SHT_DYNAMIC           = 6,
  SHT_NOTE              = 7,
  SHT_NOBITS            = 8,
  SHT_REL               = 9,
  SHT_SHLIB             = 10,
  SHT_DYNSYM            = 11,
  SHT_INIT_ARRAY        = 14,
  SHT_FINI_ARRAY        = 15,
  SHT_PREINIT_ARRAY     = 16,
  SHT_GROUP             = 17,
  SHT_SYMTAB_SHNDX      = 18,
  SHT_NUM               = 19,
  SHT_LOOS              = 0x60000000,
  SHT_HIOS              = 0x6fffffff,
  SHT_LOPROC            = 0x70000000,
  SHT_HIPROC            = 0x7fffffff,
  SHT_LOUSER            = 0x80000000,
  SHT_HIUSER            = 0xffffffff,
};

enum Elf_SHFs
{
  SHF_WRITE             = 0x1,
  SHF_ALLOC             = 0x2,
  SHF_EXECINSTR         = 0x4,
  SHF_MERGE             = 0x10,
  SHF_STRINGS           = 0x20,
  SHF_INFO_LINK         = 0x40,
  SHF_LINK_ORDER        = 0x80,
  SHF_OS_NONCONFORMING  = 0x100,
  SHF_GROUP             = 0x200,
  SHF_TLS               = 0x400,
  SHF_MASKOS            = 0x0ff00000,
  SHF_MASKPROC          = 0xf0000000,
};


typedef struct
{
  Elf32_Word    p_type;
  Elf32_Off     p_offset;
  Elf32_Addr    p_vaddr;
  Elf32_Addr    p_paddr;
  Elf32_Word    p_filesz;
  Elf32_Word    p_memsz;
  Elf32_Word    p_flags;
  Elf32_Word    p_align;
} Elf32_Phdr;

typedef struct
{
  Elf64_Word    p_type;
  Elf64_Word    p_flags;
  Elf64_Off     p_offset;
```

```
  Elf64_Addr    p_vaddr;
  Elf64_Addr    p_paddr;
  Elf64_Xword   p_filesz;
  Elf64_Xword   p_memsz;
  Elf64_Xword   p_align;
} Elf64_Phdr;

enum Elf_PTs
{
  PT_NULL               = 0,
  PT_LOAD               = 1,
  PT_DYNAMIC            = 2,
  PT_INTERP             = 3,
  PT_NOTE               = 4,
  PT_SHLIB              = 5,
  PT_PHDR               = 6,
  PT_TLS                = 7,
  PT_NUM                = 8,
  PT_LOOS               = 0x60000000,
  PT_HIOS               = 0x6fffffff,
  PT_LOPROC             = 0x70000000,
  PT_HIPROC             = 0x7fffffff,
  PT_GNU_EH_FRAME       = PT_LOOS + 0x474e550,
  PT_GNU_STACK          = PT_LOOS + 0x474e551,
  PT_GNU_RELRO          = PT_LOOS + 0x474e552,
  PT_L4_STACK           = PT_LOOS + 0x12,
  PT_L4_KIP             = PT_LOOS + 0x13,
  PT_L4_AUX             = PT_LOOS + 0x14,
};

enum ELF_PFs
{
  PF_X                  = 0x1,
  PF_W                  = 0x2,
  PF_R                  = 0x4,
  PF_MASKOS             = 0x0ff00000,
  PF_MASKPROC           = 0x7fffffff,
};

enum Elf_NTs_core
{
  NT_PRSTATUS           = 1,
  NT_FPREGSET           = 2,
  NT_PRPSINFO           = 3,
  NT_PRXREG             = 4,
  NT_TASKSTRUCT         = 4,
  NT_PLATFORM           = 5,
  NT_AUXV               = 6,
  NT_GWINDOWS           = 7,
  NT_ASRS               = 8,
  NT_PSTATUS            = 10,
  NT_PSINFO             = 13,
  NT_PRCRED             = 14,
  NT_UTSNAME            = 15,
  NT_LWPSTATUS          = 16,
  NT_LWPSINFO           = 17,
  NT_PRFPXREG           = 20,
};

enum Elf_NTs_obj
{
  NT_VERSION            = 1,
};

typedef struct
{
  Elf32_Sword   d_tag;
  union
  {
    Elf32_Word  d_val;
    Elf32_Addr  d_ptr;
  } d_un;
} Elf32_Dyn;

typedef struct
{
  Elf64_Sxword  d_tag;
  union
  {
    Elf64_Xword d_val;
    Elf64_Addr  d_ptr;
  } d_un;
} Elf64_Dyn;

enum Elf_DTs
{
  DT_NULL               = 0,
```

```
  DT_NEEDED             = 1,
  DT_PLTRELSZ           = 2,
  DT_PLTGOT             = 3,
  DT_HASH               = 4,
  DT_STRTAB             = 5,
  DT_SYMTAB             = 6,
  DT_RELA               = 7,
  DT_RELASZ             = 8,
  DT_RELAENT            = 9,
  DT_STRSZ              = 10,
  DT_SYMENT             = 11,
  DT_INIT               = 12,
  DT_FINI               = 13,
  DT_SONAME             = 14,
  DT_RPATH              = 15,
  DT_SYMBOLIC           = 16,
  DT_REL                = 17,
  DT_RELSZ              = 18,
  DT_RELENT             = 19,
  DT_PTRREL             = 20,
  DT_DEBUG              = 21,
  DT_TEXTREL            = 22,
  DT_JMPREL             = 23,
  DT_BIND_NOW           = 24,
  DT_INIT_ARRAY         = 25,
  DT_FINI_ARRAY         = 26,
  DT_INIT_ARRAYSZ       = 27,
  DT_FINI_ARRAYSZ       = 28,
  DT_RUNPATH            = 29,
  DT_FLAGS              = 30,
  DT_ENCODING           = 32,
  DT_PREINIT_ARRAY      = 32,
  DT_PREINIT_ARRAYSZ    = 33,
  DT_NUM                = 34,
  DT_LOOS               = 0x6000000d,
  DT_HIOS               = 0x6ffff000,
  DT_LOPROC             = 0x70000000,
  DT_HIPROC             = 0x7fffffff,
};

enum Elf_DFs
{
  DF_ORIGIN             = 0x00000001,
  DF_SYMBOLIC           = 0x00000002,
  DF_TEXTREL            = 0x00000004,
  DF_BIND_NOW           = 0x00000008,
  DF_STATIC_TLS         = 0x00000010,
};

enum Elf_DF_1s
{
  DF_1_NOW              = 0x00000001,
  DF_1_GLOBAL           = 0x00000002,
  DF_1_GROUP            = 0x00000004,
  DF_1_NODELETE         = 0x00000008,
  DF_1_LOADFLTR         = 0x00000010,
  DF_1_INITFIRST        = 0x00000020,
  DF_1_NOOPEN           = 0x00000040,
  DF_1_ORIGIN           = 0x00000080,
  DF_1_DIRECT           = 0x00000100,
  DF_1_TRANS            = 0x00000200,
  DF_1_INTERPOSE        = 0x00000400,
  DF_1_NODEFLIB         = 0x00000800,
  DF_1_NODUMP           = 0x00001000,
  DF_1_CONFALT          = 0x00002000,
  DF_1_ENDFILTEE        = 0x00004000,
  DF_1_DISPRELDNE       = 0x00008000,
  DF_1_DISPRELPND       = 0x00010000,
};

enum Elf_DTF_1s
{
  DTF_1_PARINIT         = 0x00000001,
  DTF_1_CONFEXP         = 0x00000002,
};

enum Elf_DF_P1s
{
  DF_P1_LAZYLOAD        = 0x00000001,
  DF_P1_GROUPPERM       = 0x00000002,
};

typedef struct
{
  Elf32_Addr          r_offset;
  Elf32_Word          r_info;
} Elf32_Rel;
```

```
typedef struct
{
  Elf32_Addr          r_offset;
  Elf32_Word          r_info;
  Elf32_Sword         r_addend;
} Elf32_Rela;

typedef struct
{
  Elf64_Addr          r_offset;
  Elf64_Xword         r_info;
} Elf64_Rel;

typedef struct
{
  Elf64_Addr          r_offset;
  Elf64_Xword         r_info;
  Elf64_Sxword        r_addend;
} Elf64_Rela;

#define ELF32_R_SYM(i)    ((i)»8)
#define ELF32_R_TYPE(i)   ((unsigned char)(i))
#define ELF32_R_INFO(s,t) (((s)«8)+(unsigned char)(t))

#define ELF64_R_SYM(i)    ((i)»32)

#define ELF64_R_TYPE(i)   ((i)&0xffffffffL)

#define ELF64_R_INFO(s,t) (((s)«32)+(t)&0xffffffffL)

enum Elf_R_386_s
{
 R_386_NONE             = 0,
 R_386_32               = 1,
 R_386_PC32             = 2,
 R_386_GOT32            = 3,
 R_386_PLT32            = 4,
 R_386_COPY             = 5,
 R_386_GLOB_DAT         = 6,
 R_386_JMP_SLOT         = 7,
 R_386_RELATIVE         = 8,
 R_386_GOTOFF           = 9,
 R_386_GOTPC            = 10,
 R_386_32PLT            = 11,
 R_386_TLS_TPOFF        = 14,
 R_386_TLS_IE           = 15,
 R_386_TLS_GOTIE        = 16,
 R_386_TLS_LE           = 17,
 R_386_TLS_GD           = 18,
 R_386_TLS_LDM          = 19,
 R_386_16               = 20,
 R_386_PC16             = 21,
 R_386_8                = 22,
 R_386_PC8              = 23,
 R_386_TLS_GD_32        = 24,
 R_386_TLS_GD_PUSH      = 25,
 R_386_TLS_GD_CALL      = 26,
 R_386_TLS_GD_POP       = 27,
 R_386_TLS_LDM_32       = 28,
 R_386_TLS_LDM_PUSH     = 29,
 R_386_TLS_LDM_CALL     = 30,
 R_386_TLS_LDM_POP      = 31,
 R_386_TLS_LDO_32       = 32,
 R_386_TLS_IE_32        = 33,
 R_386_TLS_LE_32        = 34,
 R_386_TLS_DTPMOD32     = 35,
 R_386_TLS_DTPOFF32     = 36,
 R_386_TLS_TPOFF32      = 37,
 R_386_NUM              = 38,
};

enum Elf_EF_ARM_s
{
  EF_ARM_RELEXEC        = 0x01,
  EF_ARM_HASENTRY       = 0x02,
  EF_ARM_INTERWORK      = 0x04,
  EF_ARM_APCS_26        = 0x08,
  EF_ARM_APCS_FLOAT     = 0x10,
  EF_ARM_PIC            = 0x20,
  EF_ARM_ALIGN8         = 0x40,
  EF_ARM_NEW_ABI        = 0x80,
  EF_ARM_OLD_ABI        = 0x100,

/* Other constants defined in the ARM ELF spec. version B-01.  */
/* NB. These conflict with values defined above.  */
  EF_ARM_SYMSARESORTED  = 0x04,
```

```
  EF_ARM_DYNSYMSUSESEGIDX = 0x08,
  EF_ARM_MAPSYMSFIRST   = 0x10,
  EF_ARM_EABIMASK       = 0XFF000000,

#define EF_ARM_EABI_VERSION(flags) ((flags) & EF_ARM_EABIMASK)
  EF_ARM_EABI_UNKNOWN   = 0x00000000,
  EF_ARM_EABI_VER1      = 0x01000000,
  EF_ARM_EABI_VER2      = 0x02000000,
};

enum Elf_STT_ARM_s
{
  STT_ARM_TFUNC         = 0xd,
};

enum Elf_SHF_s_ARM
{
  SHF_ARM_ENTRYSECT     = 0x10000000,
  SHF_ARM_COMDEF        = 0x80000000,
};

enum Elf_ARM_SBs
{
  PF_ARM_SB             = 0x10000000,
};

enum Elf_R_ARM_s
{
  R_ARM_NONE            = 0,
  R_ARM_PC24            = 1,
  R_ARM_ABS32           = 2,
  R_ARM_REL32           = 3,
  R_ARM_PC13            = 4,
  R_ARM_ABS16           = 5,
  R_ARM_ABS12           = 6,
  R_ARM_THM_ABS5        = 7,
  R_ARM_ABS8            = 8,
  R_ARM_SBREL32         = 9,
  R_ARM_THM_PC22        = 10,
  R_ARM_THM_PC8         = 11,
  R_ARM_AMP_VCALL9      = 12,
  R_ARM_SWI24           = 13,
  R_ARM_THM_SWI8        = 14,
  R_ARM_XPC25           = 15,
  R_ARM_THM_XPC22       = 16,
  R_ARM_COPY            = 20,
  R_ARM_GLOB_DAT        = 21,
  R_ARM_JUMP_SLOT       = 22,
  R_ARM_RELATIVE        = 23,
  R_ARM_GOTOFF          = 24,
  R_ARM_GOTPC           = 25,
  R_ARM_GOT32           = 26,
  R_ARM_PLT32           = 27,
  R_ARM_ALU_PCREL_7_0   = 32,
  R_ARM_ALU_PCREL_15_8  = 33,
  R_ARM_ALU_PCREL_23_15 = 34,
  R_ARM_LDR_SBREL_11_0  = 35,
  R_ARM_ALU_SBREL_19_12 = 36,
  R_ARM_ALU_SBREL_27_20 = 37,
  R_ARM_GNU_VTENTRY     = 100,
  R_ARM_GNU_VTINHERIT   = 101,
  R_ARM_THM_PC11        = 102,
  R_ARM_THM_PC9         = 103,
  R_ARM_RXPC25          = 249,
  R_ARM_RSBREL32        = 250,
  R_ARM_THM_RPC22       = 251,
  R_ARM_RREL32          = 252,
  R_ARM_RABS22          = 253,
  R_ARM_RPC24           = 254,
  R_ARM_RBASE           = 255,
  R_ARM_NUM             = 256,
};

enum Elf_R_AARCH64_s
{
  R_AARCH64_NONE        = 0,
  R_AARCH64_RELATIVE    = 1027,
};

enum Elf_R_X86_64_s
{
  R_X86_64_NONE         = 0,
  R_X86_64_64           = 1,
  R_X86_64_PC32         = 2,
  R_X86_64_GOT32        = 3,
  R_X86_64_PLT32        = 4,
  R_X86_64_COPY         = 5,
```

```
  R_X86_64_GLOB_DAT     = 6,
  R_X86_64_JUMP_SLOT    = 7,
  R_X86_64_RELATIVE     = 8,
  R_X86_64_GOTPCREL     = 9,
  R_X86_64_32           = 10,
  R_X86_64_32S          = 11,
  R_X86_64_16           = 12,
  R_X86_64_PC16         = 13,
  R_X86_64_8            = 14,
  R_X86_64_PC8          = 15,
  R_X86_64_DTPMOD64     = 16,
  R_X86_64_DTPOFF64     = 17,
  R_X86_64_TPOFF64      = 18,
  R_X86_64_TLSGD        = 19,
  R_X86_64_TLSLD        = 20,
  R_X86_64_DTPOFF32     = 21,
  R_X86_64_GOTTPOFF     = 22,
  R_X86_64_TPOFF32      = 23,
  R_X86_64_NUM          = 24,
};

enum Elf_STNs
{
  STN_UNDEF             = 0,
};

typedef struct
{
  Elf32_Word            st_name;
  Elf32_Addr            st_value;
  Elf32_Word            st_size;
  unsigned char         st_info;
  unsigned char         st_other;
  Elf32_Half            st_shndx;
} Elf32_Sym;

typedef struct
{
  Elf64_Word            st_name;
  unsigned char         st_info;
  unsigned char         st_other;
  Elf64_Half            st_shndx;
  Elf64_Addr            st_value;
  Elf64_Xword           st_size;
} Elf64_Sym;

#define ELF32_ST_BIND(i)     ((i)»4)

#define ELF32_ST_TYPE(i)     ((i)&0xf)

#define ELF32_ST_INFO(b,t)   (((b)«4)+((t)&0xf))

#define ELF64_ST_BIND(i)     ((i)»4)

#define ELF64_ST_TYPE(i)     ((i)&0xf)

#define ELF64_ST_INFO(b,t)   (((b)«4)+((t)&0xf))

enum Elf_STBs
{
  STB_LOCAL     = 0,
  STB_GLOBAL    = 1,
  STB_WEAK      = 2,
  STB_LOOS      = 10,
  STB_HIOS      = 12,
  STB_LOPROC    = 13,
  STB_HIPROC    = 15,
};

enum Elf_STTs
{
  STT_NOTYPE    = 0,
  STT_OBJECT    = 1,
  STT_FUNC      = 2,
  STT_SECTION   = 3,
  STT_FILE      = 4,
  STT_LOOS      = 10,
  STT_HIOS      = 12,
  STT_LOPROC    = 13,
  STT_HIPROC    = 15,
};

enum Elf_ATs
{
  AT_NULL        = 0,
  AT_IGNORE      = 1,
  AT_EXECFD      = 2,
```

```
  AT_PHDR         = 3,
  AT_PHENT        = 4,
  AT_PHNUM        = 5,
  AT_PAGESZ       = 6,
  AT_BASE         = 7,
  AT_FLAGS        = 8,
  AT_ENTRY        = 9,
  AT_NOTELF       = 10,
  AT_UID          = 11,
  AT_EUID         = 12,
  AT_GID          = 13,
  AT_EGID         = 14,
  AT_L4_AUX       = 0xf0,
  AT_L4_ENV       = 0xf1,
};

typedef struct Elf32_Auxv
{
  Elf32_Word atype;
  Elf32_Word avalue;
} Elf32_Auxv;

typedef struct Elf64_Auxv
{
  Elf64_Word atype;
  Elf64_Word avalue;
} Elf64_Auxv;

static inline int l4util_elf_check_magic(ElfW(Ehdr) const *hdr);

static inline int l4util_elf_check_arch(ElfW(Ehdr) const *hdr);

static inline ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) const *hdr);


/* Implementations */

static inline
int l4util_elf_check_magic(ElfW(Ehdr) const *hdr)
{
  return    hdr->e_ident[EI_MAG0] == ELFMAG0
         && hdr->e_ident[EI_MAG1] == ELFMAG1
         && hdr->e_ident[EI_MAG2] == ELFMAG2
         && hdr->e_ident[EI_MAG3] == ELFMAG3;
}

static inline
int l4util_elf_check_arch(ElfW(Ehdr) const *hdr)
{
  return    hdr->e_ident[EI_CLASS] == L4_ARCH_EI_CLASS
         && hdr->e_ident[EI_DATA]  == L4_ARCH_EI_DATA
         && hdr->e_machine         == L4_ARCH_E_MACHINE;
}

static inline
ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) const *hdr)
{
  return (ElfW(Phdr) *)((char *)hdr + hdr->e_phoff);
}
```

16.419 l4/util/getopt.h File Reference

getopt

`#include <l4/sys/compiler.h>`
Include dependency graph for getopt.h:

16.419.1 Detailed Description

getopt

Definition in file getopt.h.

16.420 getopt.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *              Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #ifndef _GETOPT_H
00014 #define _GETOPT_H
00015
00016 #ifndef NULL
00017 #define NULL 0
00018 #endif
00019
00020 #include <l4/sys/compiler.h>
00021
00022 EXTERN_C_BEGIN
00023
00024 /* For communication from 'getopt' to the caller.
00025    When 'getopt' finds an option that takes an argument,
00026    the argument value is returned here.
00027    Also, when 'ordering' is RETURN_IN_ORDER,
00028    each non-option ARGV-element is returned here.  */
00029
00030 extern char *optarg;
00031
00032 /* Index in ARGV of the next element to be scanned.
00033    This is used for communication to and from the caller
00034    and for communication between successive calls to 'getopt'.
00035
00036    On entry to 'getopt', zero means this is the first call; initialize.
00037
00038    When 'getopt' returns -1, this is the index of the first of the
00039    non-option elements that the caller should itself scan.
```

```

   Otherwise, 'optind' communicates from one call to the next
   how much of ARGV has been scanned so far.  */

extern int optind;

/* Callers store zero here to inhibit the error message 'getopt' prints
   for unrecognized options.  */

extern int opterr;

/* Set to an option character which was unrecognized.  */

extern int optopt;

/* Describe the long-named options requested by the application.
   The LONG_OPTIONS argument to getopt_long or getopt_long_only is a vector
   of 'struct option' terminated by an element containing a name which is
   zero.

   The field 'has_arg' is:
   no_argument    (or 0) if the option does not take an argument,
   required_argument  (or 1) if the option requires an argument,
   optional_argument  (or 2) if the option takes an optional argument.

   If the field 'flag' is not NULL, it points to a variable that is set
   to the value given in the field 'val' when the option is found, but
   left unchanged if the option is not found.

   To have a long-named option do something other than set an 'int' to
   a compiled-in constant, such as set a value from 'optarg', set the
   option's 'flag' field to zero and its 'val' field to a nonzero
   value (the equivalent single-letter option character, if there is
   one).  For long options that have a zero 'flag' field, 'getopt'
   returns the contents of the 'val' field.  */

struct option
{
  const char *name;
  /* has_arg can't be an enum because some compilers complain about
     type mismatches in all the code that assumes it is an int.  */
  int has_arg;
  int *flag;
  int val;
};

/* Names for the values of the 'has_arg' field of 'struct option'.  */

#define no_argument   0
#define required_argument 1
#define optional_argument 2

L4_CV int getopt (int argc, char *const *argv, const char *shortopts);

L4_CV int getopt_long (int argc, char *const *argv, const char *shortopts,
           const struct option *longopts, int *longind);
L4_CV int getopt_long_only (int argc, char *const *argv,
          const char *shortopts,
                const struct option *longopts, int *longind);

L4_CV int _getopt_internal (int argc, char *const *argv,
          const char *shortopts,
                const struct option *longopts, int *longind,
          int long_only);

EXTERN_C_END

#endif /* _GETOPT_H */
```

16.421 l4/util/keymap.h File Reference

Event to ASCII key mapping.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for keymap.h:

16.421.1 Detailed Description

Event to ASCII key mapping.

Definition in file keymap.h.

16.422 keymap.h

```
00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #ifndef __L4UTIL__KEYMAP_H__
00013 #define __L4UTIL__KEYMAP_H__
00014
00015 #include <l4/sys/compiler.h>
00016
00017 __BEGIN_DECLS
00018
00019 int l4util_map_event_to_keymap(unsigned value, unsigned shift);
00020
00021 __END_DECLS
00022
00023
00024 #endif /* __L4UTIL__KEYMAP_H__ */
```

16.423 l4/util/kip.h File Reference

```
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for kip.h:

Macros

- #define l4util_kip_for_each_feature(s) for (s += strlen(s) + 1; ∗s; s += strlen(s) + 1)

  Cycle through kernel features given in the KIP.

Functions

- int l4util_kip_kernel_is_ux (l4_kernel_info_t ∗)

  Return whether the kernel is running natively or under UX.
- int l4util_kip_kernel_has_feature (l4_kernel_info_t ∗, const char ∗str)

  Check if kernel supports a feature.
- unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t ∗)

  Return kernel ABI version.

16.424 kip.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/sys/kip.h>
00017 #include <l4/sys/compiler.h>
00018
```

```
EXTERN_C_BEGIN

L4_CV int l4util_kip_kernel_is_ux(l4_kernel_info_t *);

L4_CV int l4util_kip_kernel_has_feature(l4_kernel_info_t *, const char *str);

L4_CV unsigned long l4util_kip_kernel_abi_version(l4_kernel_info_t *);

EXTERN_C_END

#define l4util_kip_for_each_feature(s)        \
    for (s += strlen(s) + 1; *s; s += strlen(s) + 1)

```

16.425 l4/sys/kip.h File Reference

Kernel Info Page access functions.

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/__kip-arch.h>
#include <l4/sys/__kip-64bit.h>
```

Include dependency graph for kip.h:

This graph shows which files directly or indirectly include this file:

Macros

- #define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /∗ "L4µK" ∗/

  Kernel Info Page identifier ("L4µK").

Enumerations

- enum { L4_KIP_OFFS_READ_US = 0x900 , L4_KIP_OFFS_READ_NS = 0x980 }

Functions

- l4_umword_t l4_kip_version (l4_kernel_info_t ∗kip) L4_NOTHROW

  Get the kernel version.
- const char ∗ l4_kip_version_string (l4_kernel_info_t ∗kip) L4_NOTHROW

  Get the kernel version string.
- int l4_kernel_info_version_offset (l4_kernel_info_t ∗kip) L4_NOTHROW

  Return offset in bytes of version_strings relative to the KIP base.
- l4_cpu_time_t l4_kip_clock (l4_kernel_info_t ∗kip) L4_NOTHROW

  Return clock value from the KIP.
- l4_umword_t l4_kip_clock_lw (l4_kernel_info_t ∗kip) L4_NOTHROW

  Return least significant machine word of clock value from the KIP.
- l4_uint64_t l4_kip_clock_ns (l4_kernel_info_t ∗kip) L4_NOTHROW

  Return current clock using the KIP in nanoseconds.

16.425.1 Detailed Description

Kernel Info Page access functions.

Definition in file kip.h.

16.426 kip.h

```
00001
00006 /*
00007  * (c) 2008-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction.  Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License.  This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/l4int.h>
00028
00029 #include <l4/sys/__kip-arch.h>
00030
00034 struct l4_kip_platform_info
```

```
{
  char                              name[16];
  l4_uint32_t                       is_mp;
  struct l4_kip_platform_info_arch arch;
};

#if L4_MWORD_BITS == 32
#  include <l4/sys/__kip-32bit.h>
#else
#  include <l4/sys/__kip-64bit.h>
#endif

enum l4_kernel_info_consts_t
{
  L4_KIP_VERSION_FIASCO      = 0x87004444,
  L4_KIP_VERSION_FIASCO_MASK = 0xff00ffff,
};

enum
{
  L4_KIP_OFFS_READ_US        = 0x900,

  L4_KIP_OFFS_READ_NS        = 0x980,
};

#define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4µK" */


L4_INLINE l4_umword_t l4_kip_version(l4_kernel_info_t *kip) L4_NOTHROW;

L4_INLINE const char *l4_kip_version_string(l4_kernel_info_t *kip) L4_NOTHROW;

L4_INLINE int
l4_kernel_info_version_offset(l4_kernel_info_t *kip) L4_NOTHROW;

L4_INLINE l4_cpu_time_t
l4_kip_clock(l4_kernel_info_t *kip) L4_NOTHROW;

L4_INLINE l4_umword_t
l4_kip_clock_lw(l4_kernel_info_t *kip) L4_NOTHROW;

L4_INLINE l4_uint64_t
l4_kip_clock_ns(l4_kernel_info_t *kip) L4_NOTHROW;

/**************************************************************************
 * Implementations
 **************************************************************************/

L4_INLINE l4_umword_t
l4_kip_version(l4_kernel_info_t *kip) L4_NOTHROW
{ return kip->version & L4_KIP_VERSION_FIASCO_MASK; }

L4_INLINE const char*
l4_kip_version_string(l4_kernel_info_t *k) L4_NOTHROW
{ return (const char *)k + l4_kernel_info_version_offset(k); }

L4_INLINE int
l4_kernel_info_version_offset(l4_kernel_info_t *kip) L4_NOTHROW
{ return kip->offset_version_strings << 4; }

L4_INLINE l4_cpu_time_t
l4_kip_clock(l4_kernel_info_t *kip) L4_NOTHROW
{
  // Use kernel-provided code to determine the current clock.
  typedef l4_uint64_t (*kip_time_fn_read_us)(void);
  kip_time_fn_read_us read_us =
    (kip_time_fn_read_us)((l4_uint8_t*)kip + L4_KIP_OFFS_READ_US);
  return read_us();
}

L4_INLINE l4_cpu_time_t
l4_kip_clock_ns(l4_kernel_info_t *kip) L4_NOTHROW
{
  typedef l4_uint64_t (*kip_time_fn_read_ns)(void);
  kip_time_fn_read_ns read_ns =
    (kip_time_fn_read_ns)((l4_uint8_t*)kip + L4_KIP_OFFS_READ_NS);
  return read_ns();
}

L4_INLINE l4_umword_t
l4_kip_clock_lw(l4_kernel_info_t *kip) L4_NOTHROW
{
  /* We do the casting because the clock field is volatile */
  unsigned long *c = (unsigned long *)&kip->_clock_val;
  l4_mb();
  return c[0];
}
```

16.427 l4/util/kprintf.h File Reference

printf using the kernel debugger

`#include <l4/sys/compiler.h>`
Include dependency graph for kprintf.h:

16.427.1 Detailed Description

printf using the kernel debugger

Date

04/05/2007

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de,

Definition in file kprintf.h.

16.428 kprintf.h

```
/*****************************************************************************/
/*
 * (c) 2007-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

#ifndef __L4UTIL__INCLUDE__KPRINTF_H__
#define __L4UTIL__INCLUDE__KPRINTF_H__

#include <l4/sys/compiler.h>

EXTERN_C_BEGIN

L4_CV int l4_kprintf(const char *fmt, ...)
                     __attribute__((format (printf, 1, 2)));

EXTERN_C_END

#endif /* ! __L4UTIL__INCLUDE__KPRINTF_H__ */
```

16.429 l4/util/list_alloc.h File Reference

Simple list-based allocator.

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for list_alloc.h:

Functions

- void l4la_free (l4la_free_t **first, void *block, l4_size_t size)

  Add free memory to memory pool.
- void * l4la_alloc (l4la_free_t **first, l4_size_t size, unsigned align)

  Allocate memory from pool.
- void l4la_dump (l4la_free_t **first)

  Show all list members.
- void l4la_init (l4la_free_t **first)

  Init memory pool.
- l4_size_t l4la_avail (l4la_free_t **first)

  Show available memory in pool.

16.429.1 Detailed Description

Simple list-based allocator.

Taken from the Fiasco kernel.

Date

Alexander Warg <aw11os.inf.tu-dresden.de> Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file list_alloc.h.

16.429.2 Function Documentation

16.429.2.1 l4la_alloc()

```
void* l4la_alloc (
            l4la_free_t ** first,
            l4_size_t size,
            unsigned align )
```

Allocate memory from pool.

Parameters

first	list identifier
size	length of memory block to allocate
align	alignment

16.429.2.2 l4la_avail()

```
l4_size_t l4la_avail (
            l4la_free_t ** first )
```

Show available memory in pool.

Parameters

first	list identifier

16.429.2.3 l4la_dump()

```
void l4la_dump (
            l4la_free_t ** first )
```

Show all list members.

Parameters

first	list identifier

16.429.2.4 l4la_free()

```
void l4la_free (
            l4la_free_t ** first,
            void * block,
            l4_size_t size )
```

Add free memory to memory pool.

Parameters

first	list identifier
block	address of unused memory block
size	size of memory block

16.429.2.5 l4la_init()

```
void l4la_init (
            l4la_free_t ** first )
```

Init memory pool.

Parameters

first	list identifier

16.430 list_alloc.h

```
00001
00008 /*
00009  * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00010  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016
00017 #ifndef L4UTIL_L4LA_H
00018 #define L4UTIL_L4LA_H
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 typedef struct l4la_free_t_s
00024 {
00025   struct l4la_free_t_s *next;
00026   l4_size_t            size;
00027 } l4la_free_t;
00028
00029 #define L4LA_INITIALIZER  { 0 }
00030
00031 EXTERN_C_BEGIN
00032
00037 L4_CV void      l4la_free(l4la_free_t **first, void *block, l4_size_t size);
00038
00043 L4_CV void*     l4la_alloc(l4la_free_t **first, l4_size_t size, unsigned align);
00044
00047 L4_CV void      l4la_dump(l4la_free_t **first);
```

```
00048
00051 L4_CV void      l4la_init(l4la_free_t **first);
00052
00055 L4_CV l4_size_t l4la_avail(l4la_free_t **first);
00056
00057 EXTERN_C_END
00058
00059 #endif
```

16.431 l4/util/lock.h File Reference

Simple lock implementation.

```
#include <l4/sys/thread.h>
#include <l4/sys/compiler.h>
#include <l4/util/atomic.h>
```

Include dependency graph for lock.h:

16.431.1 Detailed Description

Simple lock implementation.

Does only work if all thread have the same priority!

Date

02/1997

Author

Michael Hohmuth hohmuth@os.inf.tu-dresden.de

Definition in file lock.h.

16.432 lock.h

```
/*****************************************************************************/
/*
 * (c) 2000-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

/*****************************************************************************/
#ifndef __L4UTIL_LOCK_H__
#define __L4UTIL_LOCK_H__

#include <l4/sys/thread.h>
#include <l4/sys/compiler.h>
#include <l4/util/atomic.h>

EXTERN_C_BEGIN

typedef l4_uint32_t l4util_simple_lock_t;

L4_INLINE int  l4_simple_try_lock(l4util_simple_lock_t *lock);
L4_INLINE void l4_simple_unlock(l4util_simple_lock_t *lock);
L4_INLINE int  l4_simple_lock_locked(l4util_simple_lock_t *lock);
L4_INLINE void l4_simple_lock_solid(register l4util_simple_lock_t *p);
L4_INLINE void l4_simple_lock(l4util_simple_lock_t * lock);

L4_INLINE int
l4_simple_try_lock(l4util_simple_lock_t *lock)
{
  return l4util_xchg32(lock, 1) == 0;
}

L4_INLINE void
l4_simple_unlock(l4util_simple_lock_t *lock)
{
  *lock = 0;
}

L4_INLINE int
l4_simple_lock_locked(l4util_simple_lock_t *lock)
{
  return (*lock == 0) ? 0 : 1;
}

L4_INLINE void
l4_simple_lock_solid(register l4util_simple_lock_t *p)
{
  while (l4_simple_lock_locked(p) || !l4_simple_try_lock(p))
    l4_thread_switch(L4_INVALID_CAP);
}

L4_INLINE void
l4_simple_lock(l4util_simple_lock_t * lock)
{
  if (!l4_simple_try_lock(lock))
    l4_simple_lock_solid(lock);
}

EXTERN_C_END

#endif
```

16.433 l4/util/mb_info.h File Reference

Multiboot info structure as defined by GRUB.

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
```

Include dependency graph for mb_info.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct l4util_mb_mod_t

  The structure type "mod_list" is used by the multiboot_info structure.
- struct l4util_mb_addr_range_t

  INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.
- struct l4util_mb_drive_t

  Drive Info structure.
- struct l4util_mb_apm_t

  APM BIOS info.
- struct l4util_mb_vbe_ctrl_t

VBE controller information.

- struct l4util_mb_vbe_mode_t

  VBE mode information.
- struct l4util_mb_info_t

  MultiBoot Info description.

Macros

- #define MB_ARD_MEMORY 1

  usable memory "Type", all others are reserved.
- #define MB_ART_MEMORY 1

  Address Range Types (ART) from "Advanced Configuration and Power Interface Specification" Rev3.0a (p.
- #define MB_ART_RESERVED 2

  in use or reserved by system
- #define MB_ART_ACPI 3

  ACPI Reclaim Memory (RAM that contains ACPI tables)
- #define MB_ART_NVS 4

  ACPI NVS Memory (must not be used by the OS.
- #define MB_ART_UNUSABLE 5

  memory in which errors have been detected
- #define l4util_mb_for_each_mmap_entry(i, mbi)

  Iterate over a memory map provided in a Multiboot info.
- #define L4UTIL_MB_MEMORY 0x00000001

  Flags to be set in the 'flags' parameter above.
- #define L4UTIL_MB_BOOTDEV 0x00000002

  is there a boot device set?
- #define L4UTIL_MB_CMDLINE 0x00000004

  is the command-line defined?
- #define L4UTIL_MB_MODS 0x00000008

  are there modules to do something with?
- #define L4UTIL_MB_AOUT_SYMS 0x00000010

  is there a symbol table loaded?
- #define L4UTIL_MB_ELF_SHDR 0x00000020

  is there an ELF section header table?
- #define L4UTIL_MB_MEM_MAP 0x00000040

  is there a full memory map?
- #define L4UTIL_MB_DRIVE_INFO 0x00000080

  Is there drive info?

- #define L4UTIL_MB_CONFIG_TABLE 0x00000100

  Is there a config table?

- #define L4UTIL_MB_BOOT_LOADER_NAME 0x00000200

  Is there a boot loader name?

- #define L4UTIL_MB_APM_TABLE 0x00000400

  Is there a APM table?

- #define L4UTIL_MB_VIDEO_INFO 0x00000800

  Is there video information?

- #define L4UTIL_MB_VALID 0x2BADB002UL

  If we are multiboot-compliant, this value is present in the eax register.

16.433.1 Detailed Description

Multiboot info structure as defined by GRUB.

Definition in file mb_info.h.

16.433.2 Macro Definition Documentation

16.433.2.1 l4util_mb_for_each_mmap_entry

```
#define l4util_mb_for_each_mmap_entry(
                i,
                mbi )
```

Value:

```
for (i = l4util_mb_first_mmap_entry(mbi);                                    \
     (unsigned long)i < (unsigned long)mbi->mmap_addr + mbi->mmap_length;    \
     i = l4util_mb_next_mmap_entry(i))
```

Iterate over a memory map provided in a Multiboot info.

Parameters

i	Name of a variable of type l4util_mb_addr_range_t ∗ that is consecutively assigned pointers to the entries of the memory map.
mbi	Pointer to the l4util_mb_info_t where the memory map can be found.

Definition at line 334 of file mb_info.h.

16.433.2.2 L4UTIL_MB_MEMORY

```
#define L4UTIL_MB_MEMORY 0x00000001
```

Flags to be set in the 'flags' parameter above.

is there basic lower/upper memory information?

Definition at line 346 of file mb_info.h.

16.433.2.3 MB_ARD_MEMORY

```
#define MB_ARD_MEMORY 1
```

usable memory "Type", all others are reserved.

Definition at line 60 of file mb_info.h.

16.433.2.4 MB_ART_MEMORY

```
#define MB_ART_MEMORY 1
```

Address Range Types (ART) from "Advanced Configuration and Power Interface Specification" Rev3.0a (p. 390). Other values are undefined. available, usable RAM

Definition at line 66 of file mb_info.h.

16.434 mb_info.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013
00014 #ifndef L4UTIL_MB_INFO_H
00015 #define L4UTIL_MB_INFO_H
00016
00017 /*****************************************************************************
00018  * Multiboot (v1)
00019  *****************************************************************************/
00020
00021 #ifndef __ASSEMBLY__
00022
00023 #include <l4/sys/compiler.h>
00024 #include <l4/sys/l4int.h>
00025
00026 /*
00027  * \defgroup l4util_mb_mod Multiboot v1
00028  * \ingroup l4util_api
00029  */
00030
00035 typedef struct
00036 {
00037   l4_uint32_t mod_start;
00038   l4_uint32_t mod_end;
00039   l4_uint32_t cmdline;
00040   l4_uint32_t pad;
00041 } l4util_mb_mod_t;
00042
00043
00050 typedef struct __attribute__((packed))
00051 {
00052   l4_uint32_t struct_size;
00053   l4_uint64_t addr;
00054   l4_uint64_t size;
00055   l4_uint32_t type;
00056   /* unspecified optional padding... */
00057 } l4util_mb_addr_range_t;
00058
00060 #define MB_ARD_MEMORY   1
00061
00066 #define MB_ART_MEMORY   1
00067 #define MB_ART_RESERVED 2
00068 #define MB_ART_ACPI     3
00070 #define MB_ART_NVS      4
00071 #define MB_ART_UNUSABLE 5
00075 typedef struct
00076 {
00077   l4_uint32_t size;
00078   l4_uint8_t drive_number;
00079   l4_uint8_t drive_mode;
00080   l4_uint16_t drive_cylinders;
00081   l4_uint8_t drive_heads;
00082   l4_uint8_t drive_sectors;
00083   l4_uint16_t drive_ports[0];
00084 } l4util_mb_drive_t;
00085
00086 /* Drive Mode.  */
00087 #define MB_DI_CHS_MODE    0
00088 #define MB_DI_LBA_MODE    1
```

```
00089
00090
00092 typedef struct
00093 {
00094   l4_uint16_t version;
00095   l4_uint16_t cseg;
00096   l4_uint32_t offset;
00097   l4_uint16_t cseg_16;
00098   l4_uint16_t dseg_16;
00099   l4_uint16_t flags;
00100   l4_uint16_t cseg_len;
00101   l4_uint16_t cseg_16_len;
00102   l4_uint16_t dseg_16_len;
00103 } __attribute__((packed)) l4util_mb_apm_t;
00104 static_assert(sizeof(l4util_mb_apm_t) == 20, "Check l4util_mb_apm_t");
00105
00106
00108 typedef struct
00109 {
00110   l4_uint8_t signature[4];
00111   l4_uint16_t version;
00112   l4_uint32_t oem_string;
00113   l4_uint32_t capabilities;
00114   l4_uint32_t video_mode;
00115   l4_uint16_t total_memory;
00116   l4_uint16_t oem_software_rev;
00117   l4_uint32_t oem_vendor_name;
00118   l4_uint32_t oem_product_name;
00119   l4_uint32_t oem_product_rev;
00120   l4_uint8_t reserved[222];
00121   l4_uint8_t oem_data[256];
00122 } __attribute__((packed)) l4util_mb_vbe_ctrl_t;
00123 static_assert(sizeof(l4util_mb_vbe_ctrl_t) == 512, "Check l4util_mb_vbe_ctrl_t");
00124
00125
00127 typedef struct
00128 {
00132   l4_uint16_t mode_attributes;
00134   l4_uint8_t win_a_attributes;
00136   l4_uint8_t win_b_attributes;
00138   l4_uint16_t win_granularity;
00140   l4_uint16_t win_size;
00142   l4_uint16_t win_a_segment;
00144   l4_uint16_t win_b_segment;
00146   l4_uint32_t win_func;
00148   l4_uint16_t bytes_per_scanline;
00154   l4_uint16_t x_resolution;
00156   l4_uint16_t y_resolution;
00158   l4_uint8_t x_char_size;
00160   l4_uint8_t y_char_size;
00162   l4_uint8_t number_of_planes;
00164   l4_uint8_t bits_per_pixel;
00166   l4_uint8_t number_of_banks;
00168   l4_uint8_t memory_model;
00170   l4_uint8_t bank_size;
00172   l4_uint8_t number_of_image_pages;
00174   l4_uint8_t reserved0;
00180   l4_uint8_t red_mask_size;
00182   l4_uint8_t red_field_position;
00184   l4_uint8_t green_mask_size;
00186   l4_uint8_t green_field_position;
00188   l4_uint8_t blue_mask_size;
00190   l4_uint8_t blue_field_position;
00192   l4_uint8_t reserved_mask_size;
00194   l4_uint8_t reserved_field_position;
00196   l4_uint8_t direct_color_mode_info;
00202   l4_uint32_t phys_base;
00204   l4_uint32_t reserved1;
00206   l4_uint16_t reversed2;
00212   l4_uint16_t linear_bytes_per_scanline;
00214   l4_uint8_t banked_number_of_image_pages;
00216   l4_uint8_t linear_number_of_image_pages;
00218   l4_uint8_t linear_red_mask_size;
00220   l4_uint8_t linear_red_field_position;
00222   l4_uint8_t linear_green_mask_size;
00224   l4_uint8_t linear_green_field_position;
00226   l4_uint8_t linear_blue_mask_size;
00228   l4_uint8_t linear_blue_field_position;
00230   l4_uint8_t linear_reserved_mask_size;
00232   l4_uint8_t linear_reserved_field_position;
00234   l4_uint32_t max_pixel_clock;
00235
00236   l4_uint8_t reserved3[190];
00238 } __attribute__ ((packed)) l4util_mb_vbe_mode_t;
00239 static_assert(sizeof(l4util_mb_vbe_mode_t) == 256, "Check l4util_mb_vbe_mode_t");
00240
00241
```

```
typedef struct
{
  l4_uint32_t flags;
  l4_uint32_t mem_lower;
  l4_uint32_t mem_upper;
  l4_uint32_t boot_device;
  l4_uint32_t cmdline;
  l4_uint32_t mods_count;
  l4_uint32_t mods_addr;
  union
  {
    struct
    {
      l4_uint32_t tabsize;
      l4_uint32_t strsize;
      l4_uint32_t addr;
      l4_uint32_t pad;
    }
    a;

    struct
    {
      l4_uint32_t num;
      l4_uint32_t size;
      l4_uint32_t addr;
      l4_uint32_t shndx;
    }
    e;
  }
  syms;

  l4_uint32_t mmap_length;
  l4_uint32_t mmap_addr;
  l4_uint32_t drives_length;
  l4_uint32_t drives_addr;
  l4_uint32_t config_table;
  l4_uint32_t boot_loader_name;
  l4_uint32_t apm_table;
  l4_uint32_t vbe_ctrl_info;
  l4_uint32_t vbe_mode_info;
  l4_uint16_t vbe_mode;
  l4_uint16_t vbe_interface_seg;
  l4_uint16_t vbe_interface_off;
  l4_uint16_t vbe_interface_len;
} l4util_mb_info_t;
static_assert(sizeof(l4util_mb_info_t) == 88, "Check l4util_mb_info_t");

static inline l4util_mb_addr_range_t *
l4util_mb_first_mmap_entry(l4util_mb_info_t *mbi)
{
  return (l4util_mb_addr_range_t *)(l4_addr_t)mbi->mmap_addr;
}

static inline l4util_mb_addr_range_t *
l4util_mb_next_mmap_entry(l4util_mb_addr_range_t *e)
{
  return (l4util_mb_addr_range_t *)((l4_addr_t)e + e->struct_size
                                    + sizeof(e->struct_size));
}

#define l4util_mb_for_each_mmap_entry(i, mbi)                                  \
  for (i = l4util_mb_first_mmap_entry(mbi);                                    \
       (unsigned long)i < (unsigned long)mbi->mmap_addr + mbi->mmap_length;    \
       i = l4util_mb_next_mmap_entry(i))

#endif /* ! __ASSEMBLY__ */

#define L4UTIL_MB_MEMORY    0x00000001

#define L4UTIL_MB_BOOTDEV   0x00000002

#define L4UTIL_MB_CMDLINE   0x00000004

#define L4UTIL_MB_MODS      0x00000008

/* These next two are mutually exclusive */
#define L4UTIL_MB_AOUT_SYMS   0x00000010

#define L4UTIL_MB_ELF_SHDR    0x00000020

#define L4UTIL_MB_MEM_MAP   0x00000040

#define L4UTIL_MB_DRIVE_INFO    0x00000080

#define L4UTIL_MB_CONFIG_TABLE    0x00000100

#define L4UTIL_MB_BOOT_LOADER_NAME  0x00000200
```

```
#define L4UTIL_MB_APM_TABLE   0x00000400

#define L4UTIL_MB_VIDEO_INFO    0x00000800


#define L4UTIL_MB_VALID     0x2BADB002UL
#define L4UTIL_MB_VALID_ASM   0x2BADB002


/*******************************************************************************
 * Multiboot2
 *******************************************************************************/

#ifndef __ASSEMBLY__

typedef struct
{
  l4_uint32_t total_size;
  l4_uint32_t reserved;
}  __attribute__((packed)) l4util_mb2_info_t;

typedef struct
{
  char string[0];
}  __attribute__((packed)) l4util_mb2_cmdline_tag_t;

typedef struct
{
  l4_uint32_t mod_start;
  l4_uint32_t mod_end;
  char string[];
}  __attribute__((packed)) l4util_mb2_module_tag_t;

typedef struct
{
  l4_uint64_t base_addr;
  l4_uint64_t length;
  l4_uint32_t type;
  l4_uint32_t reserved;
}  __attribute__((packed)) l4util_mb2_memmap_entry_t;

typedef struct
{
  l4_uint32_t entry_size;
  l4_uint32_t entry_version;
  l4util_mb2_memmap_entry_t entries[];
}  __attribute__((packed)) l4util_mb2_memmap_tag_t;

typedef struct
{
  char data[0];
} __attribute__((packed)) l4util_mb2_rsdp_tag_t;

typedef struct
{
  l4_uint32_t type;
  l4_uint32_t size;

  union
  {
    l4util_mb2_cmdline_tag_t cmdline;
    l4util_mb2_module_tag_t module;
    l4util_mb2_memmap_tag_t memmap;
    l4util_mb2_rsdp_tag_t rsdp;
  };
}  __attribute__((packed)) l4util_mb2_tag_t;

#endif  /* ! __ASSEMBLY__ */


#define L4UTIL_MB2_MAGIC    0xE85250D6
#define L4UTIL_MB2_ARCH_I386    0x0

#define L4UTIL_MB2_TERMINATOR_HEADER_TAG  0
#define L4UTIL_MB2_INFO_REQUEST_HEADER_TAG  1
#define L4UTIL_MB2_ENTRY_ADDRESS_HEADER_TAG 3
#define L4UTIL_MB2_RELOCATABLE_HEADER_TAG 10

#define L4UTIL_MB2_TAG_FLAG_REQUIRED    0

#define L4UTIL_MB2_TAG_ALIGN_SHIFT    3
#define L4UTIL_MB2_TAG_ALIGN      8

#define L4UTIL_MB2_TERMINATOR_INFO_TAG            0
#define L4UTIL_MB2_BOOT_CMDLINE_INFO_TAG          1
#define L4UTIL_MB2_MODULE_INFO_TAG                3
```

```
#define L4UTIL_MB2_MEMORY_MAP_INFO_TAG            6
#define L4UTIL_MB2_RSDP_OLD_INFO_TAG             14
#define L4UTIL_MB2_RSDP_NEW_INFO_TAG             15
#define L4UTIL_MB2_IMAGE_LOAD_BASE_PHYS_INFO_TAG 21

#define L4UTIL_MB2_RELO_PREFERED_NONE 0
#define L4UTIL_MB2_RELO_PREFERED_MIN  1
#define L4UTIL_MB2_RELO_PREFERED_MAX  2

#endif
```

16.435 l4/util/parse_cmd.h File Reference

comfortable command-line parsing

```
#include <stdarg.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for parse_cmd.h:

Typedefs

- typedef void(∗ parse_cmd_fn_t) (int)

  Function type for PARSE_CMD_FN.
- typedef void(∗ parse_cmd_fn_arg_t) (int, const char ∗, int)

  Function type for PARSE_CMD_FN_ARG.

Enumerations

- enum parse_cmd_type

  Types for parsing.

Functions

- int parse_cmdline (int ∗argc, const char ∗∗∗argv, char arg0,...)

  Parse the command-line for specified arguments and store the values into variables.

16.435.1 Detailed Description

comfortable command-line parsing

Date

2002

Author

Jork Loeser jork.loeser@inf.tu-dresden.de

Definition in file parse_cmd.h.

16.436 parse_cmd.h

```
00001
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016
00017 #ifndef __PARSE_CMD_H
00018 #define __PARSE_CMD_H
00019
00020 #include <stdarg.h>
00021 #include <l4/sys/compiler.h>
00022
00032 enum parse_cmd_type {
00033     PARSE_CMD_INT,
00034     PARSE_CMD_SWITCH,
00035     PARSE_CMD_STRING,
00036     PARSE_CMD_FN,
00037     PARSE_CMD_FN_ARG,
00038     PARSE_CMD_INC,
00039     PARSE_CMD_DEC,
00040 };
00041
00045 typedef L4_CV void (*parse_cmd_fn_t)(int);
00046
00050 typedef L4_CV void (*parse_cmd_fn_arg_t)(int, const char*, int);
00051
00052 EXTERN_C_BEGIN
00053
00140 L4_CV int parse_cmdline(int *argc, const char***argv, char arg0, ...);
00141 L4_CV int parse_cmdlinev(int *argc, const char***argv, char arg0, va_list va);
00142 L4_CV int parse_cmdline_extra(const char*argv0, const char*line, char delim,
00143             char arg0,...);
00144
00145 EXTERN_C_END
00148 #endif
00149
```

16.437 l4/util/rand.h File Reference

Simple Pseudo-Random Number Generator.

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
```

Include dependency graph for rand.h:

Functions

- l4_uint32_t l4util_rand (void)

  Deliver next random number.
- void l4util_srand (l4_uint32_t seed)

  Initialize random number generator.

16.437.1 Detailed Description

Simple Pseudo-Random Number Generator.

Date

1998

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file rand.h.

16.438 rand.h

```
00001
00008 /*
00009  * (c) 2008-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016 #ifndef __L4UTIL_RAND_H
00017 #define __L4UTIL_RAND_H
00018
00019 #define L4_RAND_MAX 65535
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/sys/l4int.h>
00023
00024 EXTERN_C_BEGIN
00025
00037 L4_CV l4_uint32_t
00038 l4util_rand(void);
00039
00046 L4_CV void
00047 l4util_srand (l4_uint32_t seed);
00048
00049 EXTERN_C_END
00050
00051 #endif /* __L4UTIL_RAND_H */
```

16.439 l4/util/splitlog2.h File Reference

Split a range in log2 aligned and size-aligned chunks.

```
#include <l4/sys/linkage.h>
#include <l4/sys/err.h>
#include <l4/util/bitops.h>
```

Include dependency graph for splitlog2.h:

Functions

- long l4util_splitlog2_hdl (l4_addr_t start, l4_addr_t end, long(∗handler)(l4_addr_t s, l4_addr_t e, int log2size))

  Split a range into log2 base and size aligned chunks.
- l4_addr_t l4util_splitlog2_size (l4_addr_t start, l4_addr_t end)

  Return log2 base and size aligned length of a range.

16.439.1 Detailed Description

Split a range in log2 aligned and size-aligned chunks.

Definition in file splitlog2.h.

16.440 splitlog2.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #ifndef __L4UTIL__INCLUDE__SPLITLOG2_H__
00013 #define __L4UTIL__INCLUDE__SPLITLOG2_H__
00014
00015 #include <l4/sys/linkage.h>
00016 #include <l4/sys/err.h>
00017 #include <l4/util/bitops.h>
00018
00019 EXTERN_C_BEGIN
00020
00033 L4_INLINE long
00034 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00035                      long (*handler)(l4_addr_t s, l4_addr_t e, int log2size));
00036
00045 L4_INLINE l4_addr_t
00046 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end);
00047
00048 EXTERN_C_END
00049
00050 /* Implementation */
00051
00052 L4_INLINE long
00053 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00054                      long (*handler)(l4_addr_t s, l4_addr_t e, int log2size))
00055 {
00056   if (end < start)
00057     return -L4_EINVAL;
00058
00059   while (start <= end)
00060     {
00061       long retval;
00062       int len2 = l4util_splitlog2_size(start, end);
00063       l4_addr_t len = 1UL « len2;
00064       if ((retval = handler(start, start + len - 1, len2)))
00065   return retval;
00066       start += len;
00067     }
00068   return 0;
00069 }
00070
00071 L4_INLINE l4_addr_t
00072 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end)
00073 {
00074   int start_bits = l4util_bsf(start);
00075   int len_bits = l4util_bsr(end - start + 1);
00076   if (start_bits != -1 && len_bits > start_bits)
00077     len_bits = start_bits;
00078
00079   return len_bits;
00080 }
00081
00082 #endif /* ! __L4UTIL__INCLUDE__SPLITLOG2_H__ */
```

16.441 l4/util/thread.h File Reference

Low-level Thread Functions.

```
#include <l4/sys/types.h>
#include <l4/sys/scheduler.h>
```

Include dependency graph for thread.h:

Macros

- #define __L4UTIL_THREAD_FUNC(name) void L4_NORETURN name(void)

  Defines a wrapper function that sets up the registers according to the calling conventions for the architecture.

16.441.1 Detailed Description

Low-level Thread Functions.

Date

1997

Author

Sebastian Schönberg

Definition in file thread.h.

16.441.2 Macro Definition Documentation

16.441.2.1 __L4UTIL_THREAD_FUNC

```
#define __L4UTIL_THREAD_FUNC(
              name ) void L4_NORETURN name(void)
```

Defines a wrapper function that sets up the registers according to the calling conventions for the architecture.

Use this as a function header when starting a low-level thread where only stack and instruction pointer are in a well-defined state.

Example:

L4UTIL_THREAD_FUNC(helper_thread) { for(;;); }

thread_cap->ex_regs((l4_umword_t)helper_thread, stack_addr);

Definition at line 72 of file thread.h.

16.442 thread.h

```
00001
00008 /*
00009  * (c) 2003-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016 #ifndef __L4_THREAD_H
00017 #define __L4_THREAD_H
00018
00019 #include <l4/sys/types.h>
00020 #include <l4/sys/scheduler.h>
00021
00022 EXTERN_C_BEGIN
00023
00046 L4_CV long
00047 l4util_create_thread(l4_cap_idx_t id, l4_utcb_t *thread_utcb,
00048                      l4_cap_idx_t factory,
00049                      l4_umword_t pc, l4_umword_t sp, l4_cap_idx_t pager,
00050                      l4_cap_idx_t task,
00051                      l4_cap_idx_t scheduler, l4_sched_param_t scp) L4_NOTHROW;
00052
00053 EXTERN_C_END
00054
00055 #ifndef L4UTIL_THREAD_FUNC
00072 #define __L4UTIL_THREAD_FUNC(name) void L4_NORETURN name(void)
00073 #define L4UTIL_THREAD_FUNC(name) __L4UTIL_THREAD_FUNC(name)
00074 #define __L4UTIL_THREAD_STATIC_FUNC(name) static L4_NORETURN void name(void)
00075 #define L4UTIL_THREAD_STATIC_FUNC(name) __L4UTIL_THREAD_STATIC_FUNC(name)
00076 #endif
00077
00078 #endif /* __L4_THREAD_H */
```

16.443 l4/sys/thread.h File Reference

Common thread related definitions.

```
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for thread.h:

This graph shows which files directly or indirectly include this file:

Enumerations

- enum L4_thread_ops {
  L4_THREAD_CONTROL_OP = 0UL , L4_THREAD_EX_REGS_OP = 1UL , L4_THREAD_SWITCH_OP =

2UL , L4_THREAD_STATS_OP = 3UL , L4_THREAD_VCPU_RESUME_OP = 4UL , L4_THREAD_REGISTER_DELETE_IRQ_OP = 5UL , L4_THREAD_MODIFY_SENDER_OP = 6UL , L4_THREAD_VCPU_CONTROL_OP = 7UL , **L4_THREAD_VCPU_CONTROL_EXT_OP** = L4_THREAD_VCPU_CONTROL_OP | 0x10000 , L4_THREAD_X86_GDT_OP = 0x10UL , L4_THREAD_ARM_TPIDRURO_OP = 0x10UL , L4_THREAD_AMD64_SET_SEGMENT_BASE_OP = 0x12UL , L4_THREAD_AMD64_GET_SEGMENT_INFO_OP = 0x13UL , L4_THREAD_OPCODE_MASK = 0xffff }

Operations on thread objects.

- enum L4_thread_control_flags { L4_THREAD_CONTROL_SET_PAGER = 0x0010000 , L4_THREAD_CONTROL_BIND_TASK = 0x0200000 , L4_THREAD_CONTROL_ALIEN = 0x0400000 , L4_THREAD_CONTROL_UX_NATIVE = 0x0800000 , L4_THREAD_CONTROL_SET_EXC_HANDLER = 0x1000000 }

  Flags for the thread control operation.
- enum L4_thread_control_mr_indices { L4_THREAD_CONTROL_MR_IDX_FLAGS = 0 , L4_THREAD_CONTROL_MR_IDX_PAGER = 1 , L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER = 2 , L4_THREAD_CONTROL_MR_IDX_FLAG_VALS = 4 , L4_THREAD_CONTROL_MR_IDX_BIND_UTCB = 5 , L4_THREAD_CONTROL_MR_IDX_BIND_TASK = 6 }

  Indices for the values in the message register for thread control.
- enum L4_thread_ex_regs_flags { L4_THREAD_EX_REGS_CANCEL = 0x10000UL , L4_THREAD_EX_REGS_TRIGGER_EXC = 0x20000UL }

  Flags for the thread ex-regs operation.

Functions

- l4_msgtag_t l4_thread_ex_regs (l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp, l4_umword_t flags) L4_NOTHROW

  Exchange basic thread registers.
- l4_msgtag_t l4_thread_ex_regs_u (l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp, l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW

  Exchange basic thread registers.
- l4_msgtag_t l4_thread_ex_regs_ret (l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp, l4_umword_t *flags) L4_NOTHROW

  Exchange basic thread registers and return previous values.
- l4_msgtag_t l4_thread_ex_regs_ret_u (l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp, l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW

  Exchange basic thread registers and return previous values.
- void l4_thread_control_start (void) L4_NOTHROW

  Start a thread control API sequence.
- void l4_thread_control_pager (l4_cap_idx_t pager) L4_NOTHROW

  Set the pager.
- void l4_thread_control_exc_handler (l4_cap_idx_t exc_handler) L4_NOTHROW

  Set the exception handler.
- void l4_thread_control_bind (l4_utcb_t *thread_utcb, l4_cap_idx_t task) L4_NOTHROW

  Bind the thread to a task.
- void l4_thread_control_alien (int on) L4_NOTHROW

  Enable alien mode.
- void l4_thread_control_ux_host_syscall (int on) L4_NOTHROW

  Enable pass through of native host (Linux) system calls.
- l4_msgtag_t l4_thread_control_commit (l4_cap_idx_t thread) L4_NOTHROW

  Commit the thread control parameters.

- l4_msgtag_t l4_thread_yield (void) L4_NOTHROW

  Yield current time slice.
- l4_msgtag_t l4_thread_switch (l4_cap_idx_t to_thread) L4_NOTHROW

  Switch to another thread (and donate the remaining time slice).
- l4_msgtag_t l4_thread_stats_time (l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW

  Get consumed time of thread in µs.
- l4_msgtag_t l4_thread_vcpu_resume_start (void) L4_NOTHROW

  vCPU return from event handler.
- l4_msgtag_t l4_thread_vcpu_resume_commit (l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW

  Commit vCPU resume.
- l4_msgtag_t l4_thread_vcpu_control (l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW

  Enable the vCPU feature for the thread.
- l4_msgtag_t l4_thread_vcpu_control_u (l4_cap_idx_t thread, l4_addr_t vcpu_state, l4_utcb_t *utcb) L4_NOTHROW

  Enable the vCPU feature for the thread.
- l4_msgtag_t l4_thread_vcpu_control_ext (l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW

  Enable the extended vCPU feature for the thread.
- l4_msgtag_t l4_thread_vcpu_control_ext_u (l4_cap_idx_t thread, l4_addr_t ext_vcpu_state, l4_utcb_t *utcb) L4_NOTHROW

  Enable the extended vCPU feature for the thread.
- l4_msgtag_t l4_thread_register_del_irq (l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW

  Register an IRQ that will trigger upon deletion events.
- l4_msgtag_t l4_thread_modify_sender_start (void) L4_NOTHROW

  Start a thread sender modification sequence.
- int l4_thread_modify_sender_add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits, l4_msgtag_t *tag) L4_NOTHROW

  Add a modification pattern to a sender modification sequence.
- l4_msgtag_t l4_thread_modify_sender_commit (l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW

  Apply (commit) a sender modification sequence.

16.443.1 Detailed Description

Common thread related definitions.

Definition in file thread.h.

16.444 thread.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction.  Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License.  This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
```

```
 * the GNU General Public License.
 */
#pragma once

#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

L4_INLINE l4_msgtag_t
l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
                  l4_umword_t flags) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_ex_regs_u(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
                    l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_ex_regs_ret(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
                      l4_umword_t *flags) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_ex_regs_ret_u(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
                        l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW;



L4_INLINE void
l4_thread_control_start(void) L4_NOTHROW;

L4_INLINE void
l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE void
l4_thread_control_pager(l4_cap_idx_t pager) L4_NOTHROW;

L4_INLINE void
l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE void
l4_thread_control_exc_handler(l4_cap_idx_t exc_handler) L4_NOTHROW;

L4_INLINE void
l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
                                l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE void
l4_thread_control_bind(l4_utcb_t *thread_utcb,
                       l4_cap_idx_t task) L4_NOTHROW;

L4_INLINE void
l4_thread_control_bind_u(l4_utcb_t *thread_utcb,
                         l4_cap_idx_t task, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE void
l4_thread_control_alien(int on) L4_NOTHROW;

L4_INLINE void
l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW;

L4_INLINE void
l4_thread_control_ux_host_syscall(int on) L4_NOTHROW;

L4_INLINE void
l4_thread_control_ux_host_syscall_u(l4_utcb_t *utcb, int on) L4_NOTHROW;



L4_INLINE l4_msgtag_t
l4_thread_control_commit(l4_cap_idx_t thread) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_yield(void) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb) L4_NOTHROW;



L4_INLINE l4_msgtag_t
l4_thread_stats_time(l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW;
```

```
L4_INLINE l4_msgtag_t
l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
                       l4_utcb_t *utcb) L4_NOTHROW;


L4_INLINE l4_msgtag_t
l4_thread_vcpu_resume_start(void) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
                             l4_msgtag_t tag) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
                               l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;


L4_INLINE l4_msgtag_t
l4_thread_vcpu_control(l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_vcpu_control_u(l4_cap_idx_t thread, l4_addr_t vcpu_state,
                         l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_vcpu_control_ext(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state,
                             l4_utcb_t *utcb) L4_NOTHROW;


L4_INLINE l4_msgtag_t
l4_thread_register_del_irq(l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
                             l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_modify_sender_start(void) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW;

L4_INLINE int
l4_thread_modify_sender_add(l4_umword_t match_mask,
                            l4_umword_t match,
                            l4_umword_t del_bits,
                            l4_umword_t add_bits,
                            l4_msgtag_t *tag) L4_NOTHROW;

L4_INLINE int
l4_thread_modify_sender_add_u(l4_umword_t match_mask,
                              l4_umword_t match,
                              l4_umword_t del_bits,
                              l4_umword_t add_bits,
                              l4_msgtag_t *tag, l4_utcb_t *u) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_modify_sender_commit(l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
                                 l4_utcb_t *u) L4_NOTHROW;

enum L4_thread_ops
{
  L4_THREAD_CONTROL_OP                = 0UL,
  L4_THREAD_EX_REGS_OP                = 1UL,
  L4_THREAD_SWITCH_OP                 = 2UL,
  L4_THREAD_STATS_OP                  = 3UL,
  L4_THREAD_VCPU_RESUME_OP            = 4UL,
  L4_THREAD_REGISTER_DELETE_IRQ_OP    = 5UL,
  L4_THREAD_MODIFY_SENDER_OP          = 6UL,
  L4_THREAD_VCPU_CONTROL_OP           = 7UL,
  L4_THREAD_VCPU_CONTROL_EXT_OP       = L4_THREAD_VCPU_CONTROL_OP | 0x10000,
  L4_THREAD_X86_GDT_OP                = 0x10UL,
  L4_THREAD_ARM_TPIDRURO_OP           = 0x10UL,
  L4_THREAD_AMD64_SET_SEGMENT_BASE_OP = 0x12UL,
  L4_THREAD_AMD64_GET_SEGMENT_INFO_OP = 0x13UL,
  L4_THREAD_OPCODE_MASK               = 0xffff,
};

```

```
enum L4_thread_control_flags
{
  L4_THREAD_CONTROL_SET_PAGER       = 0x0010000,
  L4_THREAD_CONTROL_BIND_TASK       = 0x0200000,
  L4_THREAD_CONTROL_ALIEN           = 0x0400000,
  L4_THREAD_CONTROL_UX_NATIVE       = 0x0800000,
  L4_THREAD_CONTROL_SET_EXC_HANDLER = 0x1000000,
};

enum L4_thread_control_mr_indices
{
  L4_THREAD_CONTROL_MR_IDX_FLAGS       = 0,
  L4_THREAD_CONTROL_MR_IDX_PAGER       = 1,
  L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER = 2,
  L4_THREAD_CONTROL_MR_IDX_FLAG_VALS   = 4,
  L4_THREAD_CONTROL_MR_IDX_BIND_UTCB   = 5,
  L4_THREAD_CONTROL_MR_IDX_BIND_TASK   = 6,
};

enum L4_thread_ex_regs_flags
{
  L4_THREAD_EX_REGS_CANCEL            = 0x10000UL,
  L4_THREAD_EX_REGS_TRIGGER_EXCEPTION = 0x20000UL,
};


/* IMPLEMENTATION -----------------------------------------------------------*/

#include <l4/sys/ipc.h>
#include <l4/sys/types.h>

L4_INLINE l4_msgtag_t
l4_thread_ex_regs_u(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
                    l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[0] = L4_THREAD_EX_REGS_OP | flags;
  v->mr[1] = ip;
  v->mr[2] = sp;
  return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 3, 0, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_thread_ex_regs_ret_u(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
                        l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  l4_msgtag_t ret = l4_thread_ex_regs_u(thread, *ip, *sp, *flags, utcb);
  if (l4_error_u(ret, utcb))
    return ret;

  *flags = v->mr[0];
  *ip    = v->mr[1];
  *sp    = v->mr[2];
  return ret;
}

L4_INLINE void
l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] = L4_THREAD_CONTROL_OP;
}

L4_INLINE void
l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_SET_PAGER;
  v->mr[L4_THREAD_CONTROL_MR_IDX_PAGER]  = pager;
}

L4_INLINE void
l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
                                l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS]       |= L4_THREAD_CONTROL_SET_EXC_HANDLER;
  v->mr[L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER]  = exc_handler;
}

L4_INLINE void
l4_thread_control_bind_u(l4_utcb_t *thread_utcb, l4_cap_idx_t task,
                         l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS]         |= L4_THREAD_CONTROL_BIND_TASK;
```

```
  v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_UTCB]      = (l4_addr_t)thread_utcb;
  v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK]      = L4_ITEM_MAP;
  v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK + 1]  = l4_obj_fpage(task, 0, L4_CAP_FPAGE_RWS).raw;
}

L4_INLINE void
l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS]     |= L4_THREAD_CONTROL_ALIEN;
  v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ? L4_THREAD_CONTROL_ALIEN : 0;
}

L4_INLINE void
l4_thread_control_ux_host_syscall_u(l4_utcb_t *utcb, int on) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS]     |= L4_THREAD_CONTROL_UX_NATIVE;
  v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ? L4_THREAD_CONTROL_UX_NATIVE : 0;
}

L4_INLINE l4_msgtag_t
l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW
{
  int items = 0;
  if (l4_utcb_mr_u(utcb)->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] & L4_THREAD_CONTROL_BIND_TASK)
    items = 1;
  return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 6, items, 0), L4_IPC_NEVER);
}


L4_INLINE l4_msgtag_t
l4_thread_yield(void) L4_NOTHROW
{
  l4_ipc_receive(L4_INVALID_CAP, NULL, L4_IPC_BOTH_TIMEOUT_0);
  return l4_msgtag(0, 0, 0, 0);
}

/* Preliminary, to be changed */
L4_INLINE l4_msgtag_t
l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[0] = L4_THREAD_SWITCH_OP;
  return l4_ipc_call(to_thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
}


L4_INLINE l4_msgtag_t
l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
                       l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  l4_msgtag_t res;

  v->mr[0] = L4_THREAD_STATS_OP;

  res = l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);

  if (l4_msgtag_has_error(res))
    return res;

  *us = v->mr64[l4_utcb_mr64_idx(0)];

  return res;
}

L4_INLINE l4_msgtag_t
l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[0] = L4_THREAD_VCPU_RESUME_OP;
  return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
}

L4_INLINE l4_msgtag_t
l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
                               l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
{
  return l4_ipc_call(thread, utcb, tag, L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
                  l4_umword_t flags) L4_NOTHROW
{
  return l4_thread_ex_regs_u(thread, ip, sp, flags, l4_utcb());
```

```
}

L4_INLINE l4_msgtag_t
l4_thread_ex_regs_ret(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
                      l4_umword_t *flags) L4_NOTHROW
{
  return l4_thread_ex_regs_ret_u(thread, ip, sp, flags, l4_utcb());
}

L4_INLINE void
l4_thread_control_start(void) L4_NOTHROW
{
  l4_thread_control_start_u(l4_utcb());
}

L4_INLINE void
l4_thread_control_pager(l4_cap_idx_t pager) L4_NOTHROW
{
  l4_thread_control_pager_u(pager, l4_utcb());
}

L4_INLINE void
l4_thread_control_exc_handler(l4_cap_idx_t exc_handler) L4_NOTHROW
{
  l4_thread_control_exc_handler_u(exc_handler, l4_utcb());
}


L4_INLINE void
l4_thread_control_bind(l4_utcb_t *thread_utcb, l4_cap_idx_t task) L4_NOTHROW
{
  l4_thread_control_bind_u(thread_utcb, task, l4_utcb());
}

L4_INLINE void
l4_thread_control_alien(int on) L4_NOTHROW
{
  l4_thread_control_alien_u(l4_utcb(), on);
}

L4_INLINE void
l4_thread_control_ux_host_syscall(int on) L4_NOTHROW
{
  l4_thread_control_ux_host_syscall_u(l4_utcb(), on);
}

L4_INLINE l4_msgtag_t
l4_thread_control_commit(l4_cap_idx_t thread) L4_NOTHROW
{
  return l4_thread_control_commit_u(thread, l4_utcb());
}



L4_INLINE l4_msgtag_t
l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW
{
  return l4_thread_switch_u(to_thread, l4_utcb());
}



L4_INLINE l4_msgtag_t
l4_thread_stats_time(l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW
{
  return l4_thread_stats_time_u(thread, us, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_thread_vcpu_resume_start(void) L4_NOTHROW
{
  return l4_thread_vcpu_resume_start_u(l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
                             l4_msgtag_t tag) L4_NOTHROW
{
  return l4_thread_vcpu_resume_commit_u(thread, tag, l4_utcb());
}


L4_INLINE l4_msgtag_t
l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
                             l4_utcb_t *u) L4_NOTHROW
```

```
{
  l4_msg_regs_t *m = l4_utcb_mr_u(u);
  m->mr[0] = L4_THREAD_REGISTER_DELETE_IRQ_OP;
  m->mr[1] = l4_map_obj_control(0,0);
  m->mr[2] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
  return l4_ipc_call(thread, u, l4_msgtag(L4_PROTO_THREAD, 1, 1, 0), L4_IPC_NEVER);

}

L4_INLINE l4_msgtag_t
l4_thread_register_del_irq(l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW
{
  return l4_thread_register_del_irq_u(thread, irq, l4_utcb());
}


L4_INLINE l4_msgtag_t
l4_thread_vcpu_control_u(l4_cap_idx_t thread, l4_addr_t vcpu_state,
                         l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[0] = L4_THREAD_VCPU_CONTROL_OP;
  v->mr[1] = vcpu_state;
  return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_thread_vcpu_control(l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW
{ return l4_thread_vcpu_control_u(thread, vcpu_state, l4_utcb()); }


L4_INLINE l4_msgtag_t
l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state,
                             l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[0] = L4_THREAD_VCPU_CONTROL_EXT_OP;
  v->mr[1] = ext_vcpu_state;
  return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_thread_vcpu_control_ext(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW
{ return l4_thread_vcpu_control_ext_u(thread, ext_vcpu_state, l4_utcb()); }

L4_INLINE l4_msgtag_t
l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *m = l4_utcb_mr_u(u);
  m->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
  return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
}

L4_INLINE int
l4_thread_modify_sender_add_u(l4_umword_t match_mask,
                              l4_umword_t match,
                              l4_umword_t del_bits,
                              l4_umword_t add_bits,
                              l4_msgtag_t *tag, l4_utcb_t *u) L4_NOTHROW
{
  l4_msg_regs_t *m = l4_utcb_mr_u(u);
  unsigned w = l4_msgtag_words(*tag);
  if (w >= L4_UTCB_GENERIC_DATA_SIZE - 4)
    return -L4_ENOMEM;

  m->mr[w]   = match_mask;
  m->mr[w+1] = match;
  m->mr[w+2] = del_bits;
  m->mr[w+3] = add_bits;

  *tag = l4_msgtag(l4_msgtag_label(*tag), w + 4, 0, 0);

  return 0;
}

L4_INLINE l4_msgtag_t
l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
                                 l4_utcb_t *u) L4_NOTHROW
{
  return l4_ipc_call(thread, u, tag, L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_thread_modify_sender_start(void) L4_NOTHROW
{
  return l4_thread_modify_sender_start_u(l4_utcb());
}
```

```
L4_INLINE int
l4_thread_modify_sender_add(l4_umword_t match_mask,
                            l4_umword_t match,
                            l4_umword_t del_bits,
                            l4_umword_t add_bits,
                            l4_msgtag_t *tag) L4_NOTHROW
{
  return l4_thread_modify_sender_add_u(match_mask, match,
                                       del_bits, add_bits, tag, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_thread_modify_sender_commit(l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW
{
  return l4_thread_modify_sender_commit_u(thread, tag, l4_utcb());
}
```

16.445 arm/l4/sys/thread.h File Reference

ARM-specific thread related definitions.

Functions

- l4_msgtag_t l4_thread_arm_set_tpidruro (l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW

  Set the TPIDRURO thread specific register.

16.445.1 Detailed Description

ARM-specific thread related definitions.

Definition in file thread.h.

16.446 thread.h

```
/*
 * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 *
 * As a special exception, you may use this file as part of a free software
 * library without restriction.  Specifically, if other files instantiate
 * templates or use macros or inline functions from this file, or you compile
 * this file and link it with other files to produce an executable, this
 * file does not by itself cause the resulting executable to be covered by
 * the GNU General Public License.  This exception does not however
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include_next <l4/sys/thread.h>

L4_INLINE l4_msgtag_t
l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
                             l4_utcb_t *utcb) L4_NOTHROW;

/* IMPLEMENTATION ----------------------------------------------------------*/

```

```
L4_INLINE l4_msgtag_t
l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
                             l4_utcb_t *utcb) L4_NOTHROW
{
  l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
  v->mr[0] = L4_THREAD_ARM_TPIDRURO_OP;
  v->mr[1] = tpidruro;
  return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
                     L4_IPC_NEVER);
}

L4_INLINE l4_msgtag_t
l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW
{
  return l4_thread_arm_set_tpidruro_u(thread, tpidruro, l4_utcb());
}
```

16.447 l4/vbus/vbus.h File Reference

Description of the vbus C API.

```
#include <l4/sys/compiler.h>
#include <l4/vbus/vbus_types.h>
#include <l4/sys/types.h>
```

Include dependency graph for vbus.h:

This graph shows which files directly or indirectly include this file:

Enumerations

- enum { L4VBUS_NULL = (l4_mword_t)-1 , L4VBUS_ROOT_BUS = 0 }

  Constants for device nodes.
- enum l4vbus_icu_src_types { L4VBUS_ICU_SRC_DEV_HANDLE = 1ULL << 63 }

  Flags that can be used with the ICU on a vbus device.
- enum L4vbus_dma_domain_assign_flags { L4VBUS_DMAD_UNBIND = 0 , L4VBUS_DMAD_BIND = 1 , L4VBUS_DMAD_L4RE_DMA_SPACE = 0 , L4VBUS_DMAD_KERNEL_DMA_SPACE = 2 }

  Flags for l4vbus_assign_dma_domain().

Functions

- int l4vbus_get_device_by_hid (l4_cap_idx_t vbus, l4vbus_device_handle_t parent, l4vbus_device_handle_t *child, char const *hid, int depth, l4vbus_device_t *devinfo)

  Find a device by the hardware interface identifier (HID).
- int l4vbus_get_next_device (l4_cap_idx_t vbus, l4vbus_device_handle_t parent, l4vbus_device_handle_t *child, int depth, l4vbus_device_t *devinfo)

  Find next child following `child`.
- int l4vbus_get_device (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, l4vbus_device_t *devinfo)

  Obtain detailed information about a Vbus device.
- int l4vbus_get_resource (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, int res_idx, l4vbus_resource_t *res)

  Obtain the resource description of an individual device resource.
- int l4vbus_is_compatible (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char const *cid)

  Check if the given device has a compatibility ID (CID) or HID that matches cid.
- int l4vbus_get_hid (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char *hid, unsigned long max_len)

  Get the HID (hardware identifier) of a device.
- int l4vbus_request_ioport (l4_cap_idx_t vbus, l4vbus_resource_t const *res)

  Request an IO port resource.
- int l4vbus_request_resource (l4_cap_idx_t vbus, l4vbus_resource_t const *res, int flags)

  Request a resource of a specific type.
- int l4vbus_assign_dma_domain (l4_cap_idx_t vbus, unsigned domain_id, unsigned flags, l4_cap_idx_t dma_space)

  Bind or unbind a kernel DMA space (L4::Task) or a L4Re::Dma_space to a DMA domain.

- int l4vbus_release_ioport (l4_cap_idx_t vbus, l4vbus_resource_t const *res)

  Release a previously requested IO port resource.
- int l4vbus_release_resource (l4_cap_idx_t vbus, l4vbus_resource_t const *res)

  Release a previously requested resource.
- int l4vbus_vicu_get_cap (l4_cap_idx_t vbus, l4vbus_device_handle_t icu, l4_cap_idx_t cap)

  Get capability of ICU.

16.447.1 Detailed Description

Description of the vbus C API.

Definition in file vbus.h.

16.447.2 Enumeration Type Documentation

16.447.2.1 anonymous enum

`anonymous enum`

Constants for device nodes.

Enumerator

L4VBUS_NULL	NULL device.
L4VBUS_ROOT_BUS	Root device on the vbus.

Definition at line 22 of file vbus.h.

16.447.2.2 l4vbus_icu_src_types

`enum l4vbus_icu_src_types`

Flags that can be used with the ICU on a vbus device.

Enumerator

L4VBUS_ICU_SRC_DEV_HANDLE	Flag to denote that the value should be interpreted as a device handle. This flag may be used in the `source` parameter in l4_icu_msi_info() to denote that the ICU should interpret the source ID as a device handle.

Definition at line 28 of file vbus.h.

16.448 vbus.h

```
/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *          Alexander Warg <warg@os.inf.tu-dresden.de>,
 *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#pragma once

#include <l4/sys/compiler.h>
#include <l4/vbus/vbus_types.h>
#include <l4/sys/types.h>

enum {
  L4VBUS_NULL = (l4_mword_t)-1,
  L4VBUS_ROOT_BUS = 0,
};

enum l4vbus_icu_src_types {
    L4VBUS_ICU_SRC_DEV_HANDLE = 1ULL << 63
};

__BEGIN_DECLS

int L4_CV
l4vbus_get_device_by_hid(l4_cap_idx_t vbus, l4vbus_device_handle_t parent,
                         l4vbus_device_handle_t *child, char const *hid,
                         int depth, l4vbus_device_t *devinfo);

int L4_CV
l4vbus_get_next_device(l4_cap_idx_t vbus, l4vbus_device_handle_t parent,
                       l4vbus_device_handle_t *child, int depth,
                       l4vbus_device_t *devinfo);

int L4_CV
l4vbus_get_device(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
                  l4vbus_device_t *devinfo);

int L4_CV
l4vbus_get_resource(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
                    int res_idx, l4vbus_resource_t *res);


int L4_CV
l4vbus_is_compatible(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
                     char const *cid);

int L4_CV
l4vbus_get_hid(l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char *hid,
               unsigned long max_len);

int L4_CV
l4vbus_request_ioport(l4_cap_idx_t vbus, l4vbus_resource_t const *res);

int L4_CV
L4_DEPRECATED("use l4vbus_request_ioport")
l4vbus_request_resource(l4_cap_idx_t vbus, l4vbus_resource_t const *res,
                        int flags);

enum L4vbus_dma_domain_assign_flags
{
  L4VBUS_DMAD_UNBIND = 0,
  L4VBUS_DMAD_BIND   = 1,
  L4VBUS_DMAD_L4RE_DMA_SPACE = 0,
  L4VBUS_DMAD_KERNEL_DMA_SPACE = 2,
};

int L4_CV
l4vbus_assign_dma_domain(l4_cap_idx_t vbus, unsigned domain_id,
                         unsigned flags, l4_cap_idx_t dma_space);

int L4_CV
l4vbus_release_ioport(l4_cap_idx_t vbus, l4vbus_resource_t const *res);

int L4_CV
L4_DEPRECATED("use l4vbus_release_ioport")
l4vbus_release_resource(l4_cap_idx_t vbus, l4vbus_resource_t const *res);

int L4_CV
l4vbus_vicu_get_cap(l4_cap_idx_t vbus, l4vbus_device_handle_t icu,
                    l4_cap_idx_t cap);

```

```
__END_DECLS
```

16.449 l4/vbus/vbus_interfaces.h File Reference

This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs.

`#include <l4/sys/types.h>`

Include dependency graph for vbus_interfaces.h:

This graph shows which files directly or indirectly include this file:

Typedefs

- typedef enum l4vbus_iface_type_t l4vbus_iface_type_t

  Different sub-interfaces a vbus device may support.

Enumerations

- enum l4vbus_iface_type_t

  Different sub-interfaces a vbus device may support.
- enum { L4VBUS_IFACE_SHIFT = 26 }

Functions

- unsigned l4vbus_subinterface (unsigned opcode)

  Return the ID of the vbus sub-interface.
- unsigned l4vbus_interface_opcode (unsigned opcode)

  Return the function opcode within the sub-interface of the vbus command.
- int l4vbus_subinterface_supported (l4_uint32_t dev_type, l4vbus_iface_type_t iface_type)

  Check if a vbus device supports a given sub-interface.

16.449.1 Detailed Description

This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs.

Definition in file vbus_interfaces.h.

16.449.2 Typedef Documentation

16.449.2.1 l4vbus_iface_type_t

```
typedef enum l4vbus_iface_type_t l4vbus_iface_type_t
```

Different sub-interfaces a vbus device may support.

The IPC interface of vbus devices is divided into functional groups of sub-interfaces. Every device must implement the generic interface which provides general device information. According to the type of device, additional functionality may be supported.

The sub-interface constants are first of all used to divide the function opcode space of the interface into these functional groups (see L4VBUS_IFACE_SHIFT). They also make up a bitmask that specify the type of the device, i.e. from the point of view of the client a device is defined by the kinds of sub-interfaces it supports.

16.449.3 Enumeration Type Documentation

16.449.3.1 anonymous enum

```
anonymous enum
```

Enumerator

L4VBUS_IFACE_SHIFT	Sub-interface ID shift. Divides the function opcode sent via IPC into a sub-interface ID and the actual function opcode within the sub-interface.

Definition at line 43 of file vbus_interfaces.h.

16.449.3.2 l4vbus_iface_type_t

```
enum l4vbus_iface_type_t
```

Different sub-interfaces a vbus device may support.

The IPC interface of vbus devices is divided into functional groups of sub-interfaces. Every device must implement the generic interface which provides general device information. According to the type of device, additional functionality may be supported.

The sub-interface constants are first of all used to divide the function opcode space of the interface into these functional groups (see L4VBUS_IFACE_SHIFT). They also make up a bitmask that specify the type of the device, i.e. from the point of view of the client a device is defined by the kinds of sub-interfaces it supports.

Definition at line 31 of file vbus_interfaces.h.

16.449.4 Function Documentation

16.449.4.1 l4vbus_subinterface_supported()

```
int l4vbus_subinterface_supported (
            l4_uint32_t dev_type,
            l4vbus_iface_type_t iface_type )  [inline]
```

Check if a vbus device supports a given sub-interface.

Parameters

dev_type	Device type as reported in l4vbus_device_t.
iface_type	Sub-interface type to check for.

Returns

True if the device supports the sub-interface.

Definition at line 94 of file vbus_interfaces.h.

16.450 vbus_interfaces.h

```
00001 /*
00002  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00003  *
00004  * This file is part of TUD:OS and distributed under the terms of the
```

```
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#pragma once

#include <l4/sys/types.h>

typedef enum l4vbus_iface_type_t
{
  L4VBUS_INTERFACE_ICU = 0,
  L4VBUS_INTERFACE_GPIO,
  L4VBUS_INTERFACE_PCI,
  L4VBUS_INTERFACE_PCIDEV,
  L4VBUS_INTERFACE_PM,
  L4VBUS_INTERFACE_BUS,
  L4VBUS_INTERFACE_GENERIC = 0x20
} l4vbus_iface_type_t;


enum {
  L4VBUS_IFACE_SHIFT = 26
};

L4_INLINE unsigned l4vbus_subinterface(unsigned opcode)
{
  return opcode » L4VBUS_IFACE_SHIFT;
}

L4_INLINE unsigned l4vbus_interface_opcode(unsigned opcode)
{
  return opcode & ((1 « L4VBUS_IFACE_SHIFT) - 1);
}

L4_INLINE int l4vbus_subinterface_supported(l4_uint32_t dev_type,
                                            l4vbus_iface_type_t iface_type)
{
  if (iface_type == L4VBUS_INTERFACE_GENERIC)
    return 1;

  return (dev_type & (1 « iface_type)) ? 1 : 0;
}
```

16.451 l4/vbus/vbus_types.h File Reference

This header file contains descriptions of vbus related data types and constants.

`#include <l4/sys/types.h>`

Include dependency graph for vbus_types.h:

This graph shows which files directly or indirectly include this file:

Data Structures

- struct l4vbus_resource_t

  Description of a single vbus resource.
- struct l4vbus_device_t

  Detailed information about a vbus device.

Enumerations

- enum l4vbus_resource_type_t { L4VBUS_RESOURCE_INVALID = 0 , L4VBUS_RESOURCE_IRQ , L4VBUS_RESOURCE_MEM , L4VBUS_RESOURCE_PORT , L4VBUS_RESOURCE_BUS , L4VBUS_RESOURCE_GPIO , L4VBUS_RESOURCE_DMA_DOMAIN , L4VBUS_RESOURCE_MAX }

  Description of vbus resource types.
- enum l4vbus_resource_flags_t { L4VBUS_RESOURCE_F_MEM_READ_ONLY = 0x1 , L4VBUS_RESOURCE_F_MEM_MMIO_ = 0x2000 , L4VBUS_RESOURCE_F_MEM_MMIO_WRITE = 0x4000 }

  Description of vbus resource flags.
- enum l4vbus_device_flags_t { L4VBUS_DEVICE_F_CHILDREN = 0x10 }

  Flags describing device properties, see l4vbus_device_t.

16.451.1 Detailed Description

This header file contains descriptions of vbus related data types and constants.

Definition in file vbus_types.h.

16.451.2 Enumeration Type Documentation

16.451.2.1 l4vbus_device_flags_t

```
enum l4vbus_device_flags_t
```

Flags describing device properties, see l4vbus_device_t.

Enumerator

L4VBUS_DEVICE_F_CHILDREN	Device has child devices.

Definition at line 78 of file vbus_types.h.

16.451.2.2 l4vbus_resource_flags_t

```
enum l4vbus_resource_flags_t
```

Description of vbus resource flags.

Enumerator

L4VBUS_RESOURCE_F_MEM_READ_ONLY	Memory resource is read-only.
L4VBUS_RESOURCE_F_MEM_MMIO_READ	Reading needs to be performed using the MMIO space protocol.
L4VBUS_RESOURCE_F_MEM_MMIO_WRITE	Writing needs to be performed using the MMIO space protocol.

Definition at line 51 of file vbus_types.h.

16.451.2.3 l4vbus_resource_type_t

```
enum l4vbus_resource_type_t
```

Description of vbus resource types.

Enumerator

L4VBUS_RESOURCE_INVALID	Invalid type.
L4VBUS_RESOURCE_IRQ	Interrupt resource.
L4VBUS_RESOURCE_MEM	I/O memory resource.
L4VBUS_RESOURCE_PORT	I/O port resource (x86 only)
L4VBUS_RESOURCE_BUS	Bus resource.
L4VBUS_RESOURCE_GPIO	Gpio resource.
L4VBUS_RESOURCE_DMA_DOMAIN	DMA domain.
L4VBUS_RESOURCE_MAX	Maximum resource id.

Definition at line 39 of file vbus_types.h.

16.452 vbus_types.h

```
00001 /*
```

```
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *          Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#pragma once

#include <l4/sys/types.h>

typedef l4_mword_t l4vbus_device_handle_t;
typedef l4_addr_t l4vbus_paddr_t;

typedef struct {
  l4_uint16_t    type;
  l4_uint16_t    flags;
  l4vbus_paddr_t start;
  l4vbus_paddr_t end;
  l4vbus_device_handle_t provider;
  l4_uint32_t id;
} l4vbus_resource_t;

enum l4vbus_resource_type_t {
  L4VBUS_RESOURCE_INVALID = 0,
  L4VBUS_RESOURCE_IRQ,
  L4VBUS_RESOURCE_MEM,
  L4VBUS_RESOURCE_PORT,
  L4VBUS_RESOURCE_BUS,
  L4VBUS_RESOURCE_GPIO,
  L4VBUS_RESOURCE_DMA_DOMAIN,
  L4VBUS_RESOURCE_MAX,
};

enum l4vbus_resource_flags_t {
  L4VBUS_RESOURCE_F_MEM_READ_ONLY = 0x1,
  L4VBUS_RESOURCE_F_MEM_MMIO_READ = 0x2000,
  L4VBUS_RESOURCE_F_MEM_MMIO_WRITE = 0x4000,
};

enum l4vbus_consts_t {
  L4VBUS_DEV_NAME_LEN = 64,
  L4VBUS_MAX_DEPTH = 100,
};

typedef struct {
  l4_uint32_t   type;
  char          name[L4VBUS_DEV_NAME_LEN];
  unsigned      num_resources;
  unsigned      flags;
} l4vbus_device_t;

enum l4vbus_device_flags_t {
  L4VBUS_DEVICE_F_CHILDREN = 0x10,
};
```

Chapter 17

Example Documentation

17.1 hello/server/src/main.c

This is the famous "Hello World!" program.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Frank Mehnert <fm3@os.inf.tu-dresden.de>,
 *               Lukas Grützmacher <lg2@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <unistd.h>
int
main(void)
{
  for (;;)
    {
      puts("Hello World!");
      sleep(1);
    }
}
```

17.2 examples/sys/ipc/ipc_example.c

This example shows how two threads can exchange data using the L4 IPC mechanism. One thread is sending an integer to the other thread which is returning the square of the integer. Both values are printed.

```
/*
 * (c) 2008-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>
#include <pthread-l4.h>
#include <unistd.h>
#include <stdio.h>
static pthread_t t2;
/* Thread1 is the initiator thread, i.e. it initiates the IPC calls. In
 * other words, it takes the client role. It uses L4 IPC mechanisms to send
 * an integer value to thread2 and received a calculation result back. */
static void *thread1_fn(void *arg)
{
  l4_msgtag_t tag;
  int ipc_error;
  unsigned long value = 1;
  (void)arg;
```

```
  while (1)
    {
      printf("Sending:  %ld\n", value);
      /* Store the value which we want to have squared in the first message
       * register of our UTCB. */
      l4_utcb_mr()->mr[0] = value;
      /* To an L4 IPC call, i.e. send a message to thread2 and wait for a
       * reply from thread2. The '1' in the msgtag denotes that we want to
       * transfer one word of our message registers (i.e. MR0). No timeout. */
      tag = l4_ipc_call(pthread_l4_cap(t2), l4_utcb(),
                        l4_msgtag(0, 1, 0, 0), L4_IPC_NEVER);
      /* Check for IPC error, if yes, print out the IPC error code, if not,
       * print the received result. */
      ipc_error = l4_ipc_error(tag, l4_utcb());
      if (ipc_error)
        fprintf(stderr, "thread1: IPC error: %x\n", ipc_error);
      else
        printf("Received: %ld\n", l4_utcb_mr()->mr[0]);
      /* Wait some time and increment our value. */
      sleep(1);
      value++;
    }
  return NULL;
}
/* Thread2 is in the server role, i.e. it waits for requests from others and
 * sends back the calculation results. */
static void *thread2_fn(void *arg)
{
  l4_msgtag_t tag;
  l4_umword_t label;
  int ipc_error;
  (void)arg;
  /* Wait for requests from any thread. No timeout, i.e. wait forever. */
  tag = l4_ipc_wait(l4_utcb(), &label, L4_IPC_NEVER);
  while (1)
    {
      /* Check if we had any IPC failure, if yes, print the error code
       * and just wait again. */
      ipc_error = l4_ipc_error(tag, l4_utcb());
      if (ipc_error)
        {
          fprintf(stderr, "thread2: IPC error: %x\n", ipc_error);
          tag = l4_ipc_wait(l4_utcb(), &label, L4_IPC_NEVER);
          continue;
        }
      /* So, the IPC was ok, now take the value out of message register 0
       * of the UTCB and store the square of it back to it. */
      l4_utcb_mr()->mr[0] = l4_utcb_mr()->mr[0] * l4_utcb_mr()->mr[0];
      /* Send the reply and wait again for new messages.
       * The '1' in the msgtag indicated that we want to transfer 1 word in
       * the message registers (i.e. MR0) */
      tag = l4_ipc_reply_and_wait(l4_utcb(), l4_msgtag(0, 1, 0, 0),
                                  &label, L4_IPC_NEVER);
    }
  return NULL;
}
int main(void)
{
  // We will have two threads, one is already running the main function, the
  // other (thread2) will be created using pthread_create.
  if (pthread_create(&t2, NULL, thread2_fn, NULL))
    {
      fprintf(stderr, "Thread creation failed\n");
      return 1;
    }
  // Just run thread1 in the main thread
  thread1_fn(NULL);
  return 0;
}
```

17.3 examples/sys/ipc/ipc.cfg

Sample configuration file for the IPC example.

```
# vim:se ft=lua:
local L4 = require("L4");
L4.default_loader:start({}, "rom/ex_ipc1");
```

17.4 examples/sys/start-with-exc/main.c

This example shows how to start a newly created thread with a defined set of CPU registers.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
 *               Frank Mehnert <fm3@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Start a thread with an exception reply. This example does only work on
 * the x86-32 and ARM architectures.
 */
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/ipc.h>
#include <l4/sys/utcb.h>
#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <stdlib.h>
#include <stdio.h>
/* Stack for the thread to be created. 8kB are enough. */
static char thread_stack[8 « 10];
/* The thread to be created. For illustration it will print out its
 * register set.
 */
static void L4_STICKY(thread_func(l4_umword_t *d))
{
  while (1)
    {
      printf("hey, I'm a thread\n");
      printf("got register values: %ld %ld %ld %ld %ld %ld %ld\n",
             d[7], d[6], d[5], d[4], d[2], d[1], d[0]);
      l4_sleep(800);
    }
}
/* Startup trick for this example. Put all the CPU registers on the stack so
 * that the C function above can get it on the stack. */
asm(
".global thread     \n\t"
"thread:      \n\t"
#ifdef ARCH_x86
" pusha     \n\t"
" push %esp   \n\t"
" call thread_func  \n\t"
#endif
#ifdef ARCH_arm
"       push {r0-r7}            \n\t"
"       mov r0, sp              \n\t"
"       bl thread_func          \n\t"
#endif
);
extern void thread(void);
/* Our main function */
int main(void)
{
  /* Get a capability slot for our new thread. */
  l4_cap_idx_t t1 = l4re_util_cap_alloc();
  l4_utcb_t *u = l4_utcb();
  l4_exc_regs_t *e = l4_utcb_exc_u(u);
  l4_msgtag_t tag;
  int err;
  printf("Example showing how to start a thread with an exception.\n");
  /* We do not want to implement a pager here, take the shortcut. */
  printf("Make sure to start this program with ldr-flags=eager_map\n");
  if (l4_is_invalid_cap(t1))
    return 1;
  /* Create the thread using our default factory */
  tag = l4_factory_create_thread(l4re_env()->factory, t1);
  if (l4_error(tag))
    return 1;
  /* Setup the thread by setting the pager and task. */
  l4_thread_control_start();
  l4_thread_control_pager(l4re_env()->main_thread);
  l4_thread_control_exc_handler(l4re_env()->main_thread);
  l4_thread_control_bind((l4_utcb_t *)l4re_env()->first_free_utcb,
                         L4RE_THIS_TASK_CAP);
  tag = l4_thread_control_commit(t1);
  if (l4_error(tag))
```

```
    return 2;
  /* Start the thread by finally setting instruction and stack pointer */
  tag = l4_thread_ex_regs(t1,
                          (l4_umword_t)thread,
                          (l4_umword_t)thread_stack + sizeof(thread_stack),
                          L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);
  if (l4_error(tag))
    return 3;
  l4_sched_param_t sp = l4_sched_param(1, 0);
  tag = l4_scheduler_run_thread(l4re_env()->scheduler, t1, &sp);
  if (l4_error(tag))
    return 4;
  /* Receive initial exception from just started thread */
  tag = l4_ipc_receive(t1, u, L4_IPC_NEVER);
  if ((err = l4_ipc_error(tag, u)))
    {
      printf("Umm, ipc error: %x\n", err);
      return 1;
    }
  /* We expect an exception IPC */
  if (!l4_msgtag_is_exception(tag))
    {
      printf("PF?: %lx %lx (not prepared to handle this) %ld\n",
       l4_utcb_mr_u(u)->mr[0], l4_utcb_mr_u(u)->mr[1], l4_msgtag_label(tag));
      return 1;
    }
  /* Fill out the complete register set of the new thread */
  e->sp = (l4_umword_t)(thread_stack + sizeof(thread_stack));
#ifdef ARCH_x86
  e->ip = (l4_umword_t)thread;
  e->edi = 0;
  e->esi = 1;
  e->ebp = 2;
  e->ebx = 4;
  e->edx = 5;
  e->ecx = 6;
  e->eax = 7;
#endif
#ifdef ARCH_arm
  e->pc = (l4_umword_t)thread;
  e->r[0] = 0;
  e->r[1] = 1;
  e->r[2] = 2;
  e->r[3] = 3;
  e->r[4] = 4;
  e->r[5] = 5;
  e->r[6] = 6;
  e->r[7] = 7;
#endif
  /* Send a complete exception */
  tag = l4_msgtag(0, L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
  /* Send reply and start the thread with the defined CPU register set */
  tag = l4_ipc_send(t1, u, tag, L4_IPC_NEVER);
  if ((err = l4_ipc_error(tag, u)))
    printf("Error sending IPC: %x\n", err);
  /* Idle around */
  while (1)
    l4_sleep(10000);
  return 0;
}
```

17.5 examples/sys/singlestep/main.c

This example shows how a thread can be single stepped on the x86 architecture.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Single stepping example for the x86-32 architecture.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/factory.h>
#include <l4/sys/thread.h>
#include <l4/sys/utcb.h>
```

```c
#include <l4/sys/kdebug.h>
#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
static char thread_stack[8 « 10];
static void thread_func(void)
{
  while (1)
    {
      unsigned long d = 0;
      /* Enable single stepping  */
      asm volatile("pushf; pop %0; or $256,%0; push %0; popf\n"
                   : "=r" (d) : "r" (d));
      /* Some instructions */
      asm volatile("nop");
      asm volatile("nop");
      asm volatile("nop");
      asm volatile("mov $0x12345000, %%edx" : : : "edx"); // a non-existent cap
      asm volatile("int $0x30\n");
      asm volatile("nop");
      asm volatile("nop");
      asm volatile("nop");
      /* Disabled single stepping */
      asm volatile("pushf; pop %0; and $~256,%0; push %0; popf\n"
                   : "=r" (d) : "r" (d));
      /* You won't see those */
      asm volatile("nop");
      asm volatile("nop");
      asm volatile("nop");
    }
}
int main(void)
{
  l4_msgtag_t tag;
  int ipc_stat = 0;
  l4_cap_idx_t th = l4re_util_cap_alloc();
  l4_exc_regs_t exc;
  l4_umword_t mr0, mr1;
  l4_utcb_t *u = l4_utcb();
  printf("Singlestep testing\n");
  if (l4_is_invalid_cap(th))
    return 1;
  l4_touch_rw(thread_stack, sizeof(thread_stack));
  l4_touch_ro(thread_func, 1);
  tag = l4_factory_create_thread(l4re_env()->factory, th);
  if (l4_error(tag))
    return 1;
  l4_thread_control_start();
  l4_thread_control_pager(l4re_env()->main_thread);
  l4_thread_control_exc_handler(l4re_env()->main_thread);
  l4_thread_control_bind((l4_utcb_t *)l4re_env()->first_free_utcb,
                         L4RE_THIS_TASK_CAP);
  l4_thread_control_alien(1);
  tag = l4_thread_control_commit(th);
  if (l4_error(tag))
    return 2;
  tag = l4_thread_ex_regs(th, (l4_umword_t)thread_func,
                          (l4_umword_t)thread_stack + sizeof(thread_stack),
                          0);
  if (l4_error(tag))
    return 3;
  l4_sched_param_t sp = l4_sched_param(1, 0);
  tag = l4_scheduler_run_thread(l4re_env()->scheduler, th, &sp);
  if (l4_error(tag))
    return 4;
  /* Pager/Exception loop */
  if (l4_msgtag_has_error(tag = l4_ipc_receive(th, u, L4_IPC_NEVER)))
    {
      printf("l4_ipc_receive failed");
      return 5;
    }
  memcpy(&exc, l4_utcb_exc(), sizeof(exc));
  mr0 = l4_utcb_mr()->mr[0];
  mr1 = l4_utcb_mr()->mr[1];
  for (;;)
    {
      if (l4_msgtag_is_exception(tag))
        {
          printf("PC = %08lx Trap = %08lx Err = %08lx, SP = %08lx SC-Nr: %lx\n",
                 l4_utcb_exc_pc(&exc), exc.trapno, exc.err,
                 exc.sp, exc.err » 3);
          if (exc.err » 3)
            {
              if (!(exc.err & 4))
```

```
                {
                  tag = l4_msgtag(L4_PROTO_ALLOW_SYSCALL,
                                  L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
                  if (ipc_stat)
                    enter_kdebug("Should not be 1");
                }
              else
                {
                  tag = l4_msgtag(L4_PROTO_NONE,
                                  L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
                  if (!ipc_stat)
                    enter_kdebug("Should not be 0");
                }
              ipc_stat = !ipc_stat;
            }
          l4_sleep(100);
        }
      else
        printf("Umm, non-handled request: %ld, %08lx %08lx\n",
               l4_msgtag_label(tag), mr0, mr1);
      memcpy(l4_utcb_exc(), &exc, sizeof(exc));
      /* Reply and wait */
      if (l4_msgtag_has_error(tag = l4_ipc_call(th, u, tag, L4_IPC_NEVER)))
        {
          printf("l4_ipc_call failed\n");
          return 5;
        }
      memcpy(&exc, l4_utcb_exc(), sizeof(exc));
      mr0 = l4_utcb_mr()->mr[0];
      mr1 = l4_utcb_mr()->mr[1];
    }
  return 0;
}
```

17.6 examples/sys/aliens/main.c

This example shows how system call tracing can be done.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Example to show syscall tracing.
 */
#if defined(ARCH_x86) || defined(ARCH_amd64)
// MEASURE only works on x86/amd64
//#define MEASURE
#endif
#include <l4/sys/ipc.h>
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/utcb.h>
#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/util/kumem_alloc.h>
#include <l4/sys/debugger.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
/* Architecture specifics */
#if defined(ARCH_x86) || defined(ARCH_amd64)
static int
is_alien_after_call(l4_exc_regs_t const *exc)
{
#if defined(ARCH_x86)
  return exc->err & 4;
#else
  return exc->err == 1;
#endif
}
static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
  printf("PC=%08lx SP=%08lx Err=%08lx Trap=%lx, %s syscall, SC-Nr: %lx\n",
```

```
        l4_utcb_exc_pc(exc), exc->sp, exc->err,
        exc->trapno, is_alien_after_call(exc) ? " after" : "before",
        exc->err » 3);
}
#elif defined(ARCH_arm)
static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->err & 0x40; } // TODO: Should change this to (1 « 16)
static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
  printf("PC=%08lx SP=%08lx ULR=%08lx CPSR=%08lx Err=%lx/%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->ulr, exc->cpsr,
        exc->err, exc->err » 26,
        is_alien_after_call(exc) ? " after" : "before");
}
#elif defined(ARCH_arm64)
static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->err & (1ul « 16); }
static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
  printf("PC=%08lx SP=%08lx PSTATE=%08lx Err=%lx/%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->pstate,
        exc->err, exc->err » 26,
        is_alien_after_call(exc) ? " after" : "before");
}
#elif defined(ARCH_mips)
static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return 0; }
static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
  printf("PC=%08lx SP=%08lx Cause=%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->cause,
        is_alien_after_call(exc) ? " after" : "before");
}
#else
static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->err & 1; }
static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
  printf("PC=%08lx SP=%08lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp,
        is_alien_after_call(exc) ? " after" : "before");
}
#endif
/* Measurement mode specifics.
 *
 * In measurement mode the code is less verbose and uses RDTSC for alient exception
 * performance measurement.
 */
#ifdef MEASURE
#include <l4/util/rdtsc.h>
static inline void
calibrate_timer(void)
{
  l4_calibrate_tsc(l4re_kip());
}
static inline void
print_timediff(l4_cpu_time_t start)
{
  e = l4_rdtsc();
  printf("time %lld\n", l4_tsc_to_ns(e - start));
}
static inline void
alien_sleep(void)
{
  l4_sleep(0);
}
static inline void
print_exc_state(l4_exc_regs_t const *exc)
{
  if (0)
    _print_exc_state(exc);
}
#else
static inline void
calibrate_timer(void)
{
}
static inline void
print_timediff(l4_cpu_time_t start)
```

```
{
  (void)start;
}
static inline l4_cpu_time_t
l4_rdtsc(void)
{
  return 0;
}
static inline void
alien_sleep(void)
{
  l4_sleep(1000);
}
static inline void
print_exc_state(l4_exc_regs_t const *exc)
{
  _print_exc_state(exc);
}
#endif
static char alien_thread_stack[8 « 10];
static l4_cap_idx_t alien;
static void alien_thread(void)
{
  while (1)
    {
      l4_ipc_call(0x1234 « L4_CAP_SHIFT, l4_utcb(),
                  l4_msgtag(0, 0, 0, 0), L4_IPC_NEVER);
      alien_sleep();
    }
}
int main(void)
{
  l4_msgtag_t tag;
  l4_cpu_time_t s;
  l4_utcb_t *u = l4_utcb();
  l4_exc_regs_t exc;
  l4_umword_t mr0, mr1;
  printf("Alien feature testing\n");
  l4_debugger_set_object_name(l4re_env()->main_thread, "alientest");
  /* Start alien thread */
  if (l4_is_invalid_cap(alien = l4re_util_cap_alloc()))
    return 1;
  l4_touch_rw(alien_thread_stack, sizeof(alien_thread_stack));
  tag = l4_factory_create_thread(l4re_env()->factory, alien);
  if (l4_error(tag))
    return 2;
  l4_debugger_set_object_name(alien, "alienth");
  l4_addr_t kumem;
  if (l4re_util_kumem_alloc(&kumem, 0, L4RE_THIS_TASK_CAP, l4re_env()->rm))
    return 3;
  l4_thread_control_start();
  l4_thread_control_pager(l4re_env()->main_thread);
  l4_thread_control_exc_handler(l4re_env()->main_thread);
  l4_thread_control_bind((l4_utcb_t *)kumem, L4RE_THIS_TASK_CAP);
  l4_thread_control_alien(1);
  tag = l4_thread_control_commit(alien);
  if (l4_error(tag))
    return 4;
  tag = l4_thread_ex_regs(alien,
                          (l4_umword_t)alien_thread,
                          (l4_umword_t)alien_thread_stack + sizeof(alien_thread_stack),
                          0);
  if (l4_error(tag))
    return 5;
  l4_sched_param_t sp = l4_sched_param(1, 0);
  tag = l4_scheduler_run_thread(l4re_env()->scheduler, alien, &sp);
  if (l4_error(tag))
    return 6;
  calibrate_timer();
  /* Pager/Exception loop */
  if (l4_msgtag_has_error(tag = l4_ipc_receive(alien, u, L4_IPC_NEVER)))
    {
      printf("l4_ipc_receive failed");
      return 7;
    }
  memcpy(&exc, l4_utcb_exc(), sizeof(exc));
  mr0 = l4_utcb_mr()->mr[0];
  mr1 = l4_utcb_mr()->mr[1];
  for (;;)
    {
      s = l4_rdtsc();
      if (l4_msgtag_is_exception(tag))
        {
          print_exc_state(&exc);
          tag = l4_msgtag(is_alien_after_call(&exc)
                           ? 0 : L4_PROTO_ALLOW_SYSCALL,
                          L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
```

```
        }
      else
        printf("Umm, non-handled request (like PF): %lx %lx\n", mr0, mr1);
      memcpy(l4_utcb_exc(), &exc, sizeof(exc));
      /* Reply and wait */
      if (l4_msgtag_has_error(tag = l4_ipc_call(alien, u, tag, L4_IPC_NEVER)))
        {
          printf("l4_ipc_call failed\n");
          return 8;
        }
      memcpy(&exc, l4_utcb_exc(), sizeof(exc));
      mr0 = l4_utcb_mr()->mr[0];
      mr1 = l4_utcb_mr()->mr[1];
      print_timediff(s);
    }
  return 0;
}
```

17.7 examples/sys/utcb-ipc/main.c

This example shows how to send IPC using the UTCB to store payload.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/utcb.h>
#include <l4/sys/task.h>
#include <l4/sys/vcon.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/util/kumem_alloc.h>
#include <l4/util/thread.h>
#include <stdio.h>
#include <string.h>
static unsigned char stack2[8 « 10] __attribute__((aligned(8)));
static l4_cap_idx_t thread1_cap, thread2_cap;
static void vlogprintn(const char *s, int l)
{
  if (l > L4_VCON_WRITE_SIZE)
    l = L4_VCON_WRITE_SIZE;
  l4_vcon_send(L4_BASE_LOG_CAP, s, l);
}
static void vlogprint(const char *s)
{
  vlogprintn(s, strlen(s));
}
static void vlogprintc(const char c)
{
  vlogprintn(&c, 1);
}
static void thread1(void)
{
  l4_msg_regs_t *mr = l4_utcb_mr();
  l4_msgtag_t tag;
  int i, j;
  printf("Thread1 up (%p)\n", l4_utcb());
  for (i = 0; i < 10; i++)
    {
      for (j = 0; j < L4_UTCB_GENERIC_DATA_SIZE; j++)
        mr->mr[j] = 'A' + (i + j) % ('~' - 'A' + 1);
      tag = l4_msgtag(0, L4_UTCB_GENERIC_DATA_SIZE, 0, 0);
      if (l4_msgtag_has_error(l4_ipc_send(thread2_cap, l4_utcb(), tag, L4_IPC_NEVER)))
  printf("IPC-send error\n");
    }
  mr->mr[0] = 1;
  if (l4_msgtag_has_error(l4_ipc_send(thread2_cap, l4_utcb(), tag, L4_IPC_NEVER)))
    printf("IPC-send error\n");
  printf("Thread1 done\n");
}
L4UTIL_THREAD_STATIC_FUNC(thread2)
{
  l4_msgtag_t tag;
```

```
  l4_msg_regs_t mr;
  unsigned i;
  // No printf() here because this would require a working pthread environment!
  vlogprint("Thread2 up\n");
  while (1)
    {
      if (l4_msgtag_has_error(tag = l4_ipc_receive(thread1_cap, l4_utcb(), L4_IPC_NEVER)))
        vlogprint("IPC receive error\n");
      memcpy(&mr, l4_utcb_mr(), sizeof(mr));
      if (mr.mr[0] == 1) // exit notification
        break;
      vlogprint("Thread2 receive: ");
      for (i = 0; i < l4_msgtag_words(tag); i++)
        vlogprintc((char)mr.mr[i]);
      vlogprint("\n");
    }
  vlogprint("Thread2 done, switching to thread1\n");
  if (l4_msgtag_has_error(l4_ipc_send(thread1_cap, l4_utcb(),
                                      tag, L4_IPC_NEVER)))
    vlogprint("IPC-send error\n");
  // In theory this could hit if the above IPC send operation doesn't switch
  // to the other thread.
  __builtin_trap();
}
int main(void)
{
  l4_msgtag_t tag;
  thread1_cap = l4re_env()->main_thread;
  thread2_cap = l4re_util_cap_alloc();
  if (l4_is_invalid_cap(thread2_cap))
    {
      printf("Cannot allocate thread2 capability\n");
      return 1;
    }
  tag = l4_factory_create_thread(l4re_env()->factory, thread2_cap);
  if (l4_error(tag))
    {
      printf("Cannot create thread2\n");
      return 2;
    }
  l4_addr_t kumem;
  if (l4re_util_kumem_alloc(&kumem, 0, L4RE_THIS_TASK_CAP, l4re_env()->rm))
    {
      printf("Cannot allocate UTCB for thread2\n");
      return 3;
    }
  l4_thread_control_start();
  l4_thread_control_pager(l4re_env()->rm);
  l4_thread_control_exc_handler(l4re_env()->rm);
  l4_thread_control_bind((l4_utcb_t *)kumem, L4RE_THIS_TASK_CAP);
  tag = l4_thread_control_commit(thread2_cap);
  if (l4_error(tag))
    {
      printf("Cannot set thread2 thread parameters\n");
      return 4;
    }
  tag = l4_thread_ex_regs(thread2_cap,
                          (l4_umword_t)thread2,
                          (l4_umword_t)(stack2 + sizeof(stack2)), 0);
  if (l4_error(tag))
    {
      printf("Cannot set thread2 IP/SP\n");
      return 5;
    }
  l4_sched_param_t sp = l4_sched_param(1, 0);
  tag = l4_scheduler_run_thread(l4re_env()->scheduler, thread2_cap, &sp);
  if (l4_error(tag))
    {
      printf("Cannot start thread2\n");
      return 6;
    }
  thread1();
  if (l4_msgtag_has_error(l4_ipc_receive(thread2_cap, l4_utcb(),
                                         L4_IPC_NEVER)))
    printf("IPC-receive error\n");
  l4_task_unmap(L4RE_THIS_TASK_CAP,
                l4_obj_fpage(thread2_cap, 0, L4_FPAGE_RWX),
                L4_FP_ALL_SPACES);
  printf("Terminated thread2. Terminating.\n");
  return 0;
}
```

17.8 examples/sys/ux-vhw/main.c

This example shows how to iterate the virtual hardware descriptors under Fiasco-UX.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/vhw.h>
#include <l4/util/util.h>
#include <l4/util/kip.h>
#include <l4/re/env.h>
#include <stdlib.h>
#include <stdio.h>
static void print_entry(struct l4_vhw_entry *e)
{
  printf("type: %d  mem start: %08lx  end: %08lx\n"
         "irq: %d pid %d\n",
   e->type, e->mem_start, e->mem_size,
   e->irq_no, e->provider_pid);
}
int main(void)
{
  l4_kernel_info_t *kip = l4re_kip();
  struct l4_vhw_descriptor *vhw;
  int i;
  if (!kip)
    {
      printf("KIP not available!\n");
      return 1;
    }
  if (!l4util_kip_kernel_is_ux(kip))
    {
      printf("This example is for Fiasco-UX only.\n");
      return 1;
    }
  vhw = l4_vhw_get(kip);
  printf("kip at %p, vhw at %p\n", kip, vhw);
  printf("magic: %08x, version: %08x, count: %02d\n",
         vhw->magic, vhw->version, vhw->count);
  for (i = 0; i < vhw->count; i++)
    print_entry(l4_vhw_get_entry(vhw, i));
  return 0;
}
```

17.9 examples/sys/isr/main.c

Example of an interrupt service routine.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * This example shall show how to connect to an interrupt, receive interrupt
 * events and detach again. As the interrupt source we'll use the virtual
 * key interrupt. The interrupt number of the virtual key interrupt can be
 * found in the kernel info page.
 */
#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/namespace.h>
#include <l4/sys/utcb.h>
#include <l4/sys/irq.h>
#include <l4/sys/factory.h>
#include <l4/sys/icu.h>
#include <stdio.h>
int main(void)
{
  int irqno = 1;
```

```
  l4_cap_idx_t irqcap, icucap;
  l4_msgtag_t tag;
  int err;
  icucap = l4re_env_get_cap("icu");
  /* Get a free capability slot for the ICU capability */
  if (l4_is_invalid_cap(icucap))
    {
      printf("Did not find an ICU\n");
      return 1;
    }
  /* Get another free capaiblity slot for the corresponding IRQ object*/
  if (l4_is_invalid_cap(irqcap = l4re_util_cap_alloc()))
    return 1;
  /* Create IRQ object */
  if (l4_error(tag = l4_factory_create_irq(l4re_global_env->factory, irqcap)))
    {
      printf("Could not create IRQ object: %lx\n", l4_error(tag));
      return 1;
    }
  /*
   * Bind the recently allocated IRQ object to the IRQ number irqno
   * as provided by the ICU.
   */
  if (l4_error(l4_icu_bind(icucap, irqno, irqcap)))
    {
      printf("Binding IRQ%d to the ICU failed\n", irqno);
      return 1;
    }
  /* Bind ourselves to the IRQ */
  tag = l4_rcv_ep_bind_thread(irqcap, l4re_env()->main_thread, 0xDEAD);
  if ((err = l4_error(tag)))
    {
      printf("Error binding to IRQ %d: %d\n", irqno, err);
      return 1;
    }
  printf("Attached to key IRQ %d\nPress keys now, Shift-Q to exit\n", irqno);
  /* IRQ receive loop */
  while (1)
    {
      unsigned long label = 0;
      /* Wait for the interrupt to happen */
      tag = l4_irq_receive(irqcap, L4_IPC_NEVER);
      if ((err = l4_ipc_error(tag, l4_utcb())))
        printf("Error on IRQ receive: %d\n", err);
      else
        {
          /* Process the interrupt -- may do a 'break' */
          printf("Got IRQ with label 0x%lX\n", label);
        }
    }
  /* We're done, detach from the interrupt. */
  tag = l4_irq_detach(irqcap);
  if ((err = l4_error(tag)))
    printf("Error detach from IRQ: %d\n", err);
  return 0;
}
```

17.10 examples/clntsrv/server.cc

Client/Server example using C++ infrastructure – Server implementation.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/re/util/br_manager>
#include <l4/sys/cxx/ipc_epiface>
#include "shared.h"
static L4Re::Util::Registry_server<L4Re::Util::Br_manager_hooks> server;
class Calculation_server : public L4::Epiface_t<Calculation_server, Calc>
{
public:
  int op_sub(Calc::Rights, l4_uint32_t a, l4_uint32_t b, l4_uint32_t &res)
```

```
  {
    res = a - b;
    return 0;
  }
  int op_neg(Calc::Rights, l4_uint32_t a, l4_uint32_t &res)
  {
    res = -a;
    return 0;
  }
};
int
main()
{
  static Calculation_server calc;
  // Register calculation server
  if (!server.registry()->register_obj(&calc, "calc_server").is_valid())
    {
      printf("Could not register my service, is there a 'calc_server' in the caps table?\n");
      return 1;
    }
  printf("Welcome to the calculation server!\n"
         "I can do subtractions and negations.\n");
  // Wait for client requests
  server.loop();
  return 0;
}
```

17.11 examples/clntsrv/client.cc

Client/Server example using C++ infrastructure – Client implementation.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/err.h>
#include <l4/sys/types.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <stdio.h>
#include "shared.h"
int
main()
{
  L4::Cap<Calc> server = L4Re::Env::env()->get_cap<Calc>("calc_server");
  if (!server.is_valid())
    {
      printf("Could not get server capability!\n");
      return 1;
    }
  l4_uint32_t val1 = 8;
  l4_uint32_t val2 = 5;
  printf("Asking for %d - %d\n", val1, val2);
  if (server->sub(val1, val2, &val1))
    {
      printf("Error talking to server\n");
      return 1;
    }
  printf("Result of subtract call: %d\n", val1);
  printf("Asking for -%d\n", val1);
  if (server->neg(val1, &val1))
    {
      printf("Error talking to server\n");
      return 1;
    }
  printf("Result of negate call: %d\n", val1);
  return 0;
}
```

17.12 examples/clntsrv/clntsrv.cfg

Sample configuration file for the client/server example.

```
-- vim:set ft=lua:
-- Include L4 functionality
local L4 = require("L4");
-- Some shortcut for less typing
local ld = L4.default_loader;
-- Channel for the two programs to talk to each other.
local calc_server = ld:new_channel();
-- The server program, getting the channel in server mode.
ld:start({ caps = { calc_server = calc_server:svr() },
           log = { "server", "blue" } },
         "rom/ex_clntsrv-server");
-- The client program, getting the 'calc_server' channel to be able to talk
-- to the server. The client will be started with a green log output.
ld:start({ caps = { calc_server = calc_server },
           log = { "client", "green" } },
         "rom/ex_clntsrv-client");
```

17.13 examples/libs/l4re/c/ma+rm.c

Coarse grained memory allocation, in C.

```
/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/re/c/mem_alloc.h>
#include <l4/re/c/rm.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/sys/err.h>
#include <stdio.h>
#include <string.h>
static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
                        void **virt_addr)
{
  int r;
  l4re_ds_t ds;
  /* Allocate a free capability index for our data space */
  ds = l4re_util_cap_alloc();
  if (l4_is_invalid_cap(ds))
    return -L4_ENOMEM;
  size_in_bytes = l4_trunc_page(size_in_bytes);
  /* Allocate memory via a dataspace */
  if ((r = l4re_ma_alloc(size_in_bytes, ds, flags)))
    return r;
  /* Make the dataspace visible in our address space */
  *virt_addr = 0;
  if ((r = l4re_rm_attach(virt_addr, size_in_bytes,
                          L4RE_RM_F_SEARCH_ADDR | L4RE_RM_F_RWX, ds, 0,
                          flags & L4RE_MA_SUPER_PAGES
                             ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
    {
      /* Free dataspace again */
      l4re_util_cap_free_um(ds);
      return r;
    }
  /* Done, virtual address is in virt_addr */
  return 0;
}
static int free_mem(void *virt_addr)
{
  int r;
  l4re_ds_t ds;
  /* Detach memory from our address space */
  if ((r = l4re_rm_detach_ds(virt_addr, &ds)))
    return r;
  /* Free memory at our memory allocator */
  l4re_util_cap_free_um(ds);
  /* All went ok */
  return 0;
}
int main(void)
{
  void *virt;
  /* Allocate memory: 16k Bytes (usually) */
  if (allocate_mem(4 * L4_PAGESIZE, 0, &virt))
    return 1;
  printf("Allocated memory.\n");
  /* Do something with the memory */
```

```
  memset(virt, 0x12, 4 * L4_PAGESIZE);
  printf("Touched memory.\n");
  /* Free memory */
  if (free_mem(virt))
    return 2;
  printf("Freed and done. Bye.\n");
  return 0;
}
```

17.14 examples/libs/l4re/c++/mem_alloc/ma+rm.cc

Coarse grained memory allocation, in C++.

```
/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/re/mem_alloc>
#include <l4/re/rm>
#include <l4/re/env>
#include <l4/re/dataspace>
#include <l4/re/util/cap_alloc>
#include <l4/sys/err.h>
#include <cstdio>
#include <cstring>
static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
                        void **virt_addr)
{
  int r;
  L4::Cap<L4Re::Dataspace> d;
  /* Allocate a free capability index for our data space */
  d = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
  if (!d.is_valid())
    return -L4_ENOMEM;
  size_in_bytes = l4_trunc_page(size_in_bytes);
  /* Allocate memory via a dataspace */
  if ((r = L4Re::Env::env()->mem_alloc()->alloc(size_in_bytes, d, flags)))
    return r;
  /* Make the dataspace visible in our address space */
  *virt_addr = 0;
  if ((r = L4Re::Env::env()->rm()->attach(virt_addr, size_in_bytes,
                                          L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                          L4::Ipc::make_cap_rw(d), 0,
                                          flags & L4Re::Mem_alloc::Super_pages
                                            ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
    return r;
  /* Done, virtual address is in virt_addr */
  return 0;
}
static int free_mem(void *virt_addr)
{
  int r;
  L4::Cap<L4Re::Dataspace> ds;
  /* Detach memory from our address space */
  if ((r = L4Re::Env::env()->rm()->detach(virt_addr, &ds)))
    return r;
  /* Release and return capability slot to allocator */
  L4Re::Util::cap_alloc.free(ds, L4Re::Env::env()->task().cap());
  /* All went ok */
  return 0;
}
int main(void)
{
  void *virt;
  /* Allocate memory: 16k Bytes (usually) */
  if (allocate_mem(4 * L4_PAGESIZE, 0, &virt))
    return 1;
  printf("Allocated memory.\n");
  /* Do something with the memory */
  memset(virt, 0x12, 4 * L4_PAGESIZE);
  printf("Touched memory.\n");
  /* Free memory */
  if (free_mem(virt))
    return 2;
  printf("Freed and done. Bye.\n");
  return 0;
}
```

17.15 examples/libs/l4re/c++/shared_ds/ds_clnt.cc

Sharing memory between applications, client side.

```
/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *          Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/re/util/cap_alloc> // L4::Cap
#include <l4/re/dataspace>      // L4Re::Dataspace
#include <l4/re/rm>             // L4::Rm
#include <l4/re/env>            // L4::Env
#include <l4/sys/cache.h>
#include <cstring>
#include <cstdio>
#include <unistd.h>
#include "interface.h"
int main()
{
  /*
   * Try to get server interface cap.
   */
  L4::Cap<My_interface> svr = L4Re::Env::env()->get_cap<My_interface>("shm");
  if (!svr.is_valid())
    {
      printf("Could not get the server capability\n");
      return 1;
    }
  /*
   * Alloc data space cap slot
   */
  L4::Cap<L4Re::Dataspace> ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
  if (!ds.is_valid())
    {
      printf("Could not get capability slot!\n");
      return 1;
    }
  /*
   * Alloc server notifier IRQ cap slot
   */
  L4::Cap<L4::Irq> irq = L4Re::Util::cap_alloc.alloc<L4::Irq>();
  if (!irq.is_valid())
    {
      printf("Could not get capability slot!\n");
      return 1;
    }
  /*
   * Request shared data-space cap.
   */
  if (svr->get_shared_buffer(ds, irq))
    {
      printf("Could not get shared memory dataspace!\n");
      return 1;
    }
  /*
   * Attach to arbitrary region
   */
  char *addr = 0;
  int err = L4Re::Env::env()->rm()->attach(&addr, ds->size(),
                                           L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                           L4::Ipc::make_cap_rw(ds));
  if (err < 0)
    {
      printf("Error attaching data space: %s\n", l4sys_errtostr(err));
      return 1;
    }
  printf("Content: %s\n", addr);
  // wait a bit for the demo effect
  printf("Sleeping a bit...\n");
  sleep(1);
  /*
   * Fill in new stuff
   */
  memset(addr, 0, ds->size());
  char const * const msg = "Hello from client, too!";
  printf("Setting new content in shared memory\n");
  snprintf(addr, strlen(msg)+1, msg);
  l4_cache_clean_data((unsigned long)addr,
                      (unsigned long)addr + strlen(msg) + 1);
  // notify the server
  irq->trigger();
```

```
  /*
   * Detach region containing addr, result should be Detached_ds (other results
   * only apply if we split regions etc.).
   */
  err = L4Re::Env::env()->rm()->detach(addr, 0);
  if (err)
    printf("Failed to detach region\n");
  /* Free objects and capabilties, just for completeness. */
  L4Re::Util::cap_alloc.free(ds, L4Re::This_task);
  L4Re::Util::cap_alloc.free(irq, L4Re::This_task);
  return 0;
}
```

17.16 examples/libs/l4re/c++/shared_ds/ds_srv.cc

Sharing memory between applications, server/creator side.

```
/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *          Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/re/env>
#include <l4/re/namespace>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/re/dataspace>
#include <l4/cxx/ipc_server>
#include <l4/sys/typeinfo_svr>
#include <cstring>
#include <cstdio>
#include <unistd.h>
#include "interface.h"
class My_server_obj : public L4::Server_object_t<L4::Kobject>
{
private:
  L4::Cap<L4Re::Dataspace> _shm;
  L4::Cap<L4::Irq> _irq;
public:
  explicit My_server_obj(L4::Cap<L4Re::Dataspace> shm, L4::Cap<L4::Irq> irq)
  : _shm(shm), _irq(irq)
  {}
  int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};
int My_server_obj::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
{
  // we don't care about the original object reference, however
  // we could read out the access rights from the lowest 2 bits
  (void) obj;
  l4_msgtag_t t;
  ios » t; // extract the tag
  switch (t.label())
    {
    case L4::Meta::Protocol:
      // handle the meta protocol requests, implementing the
      // runtime dynamic type system for L4 objects.
      return L4::Util::handle_meta_request<My_interface>(ios);
    case 0:
      // since we have just one operation we have no opcode dispatch,
      // and just return the data-space and the notifier IRQ capabilities
      ios « _shm « _irq;
      return 0;
    default:
      // every other protocol is not supported.
      return -L4_EBADPROTO;
    }
}
class Shm_observer : public L4::Irq_handler_object
{
private:
  char *_shm;
public:
  explicit Shm_observer(char *shm)
  : _shm(shm)
  {}
  int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};
int Shm_observer::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
```

```
{
  // We don't care about the original object reference, however
  // we could read out the access rights from the lowest 2 bits
  (void)obj;
  // Since we end up here in this function, we got a 'message' from the IRQ
  // that is bound to us. The 'ios' stream won't contain any valuable info.
  (void)ios;
  printf("Client sent us: %s\n", _shm);
  return 0;
}
static L4Re::Util::Registry_server<> server;
enum
{
  DS_SIZE = 4 « 12,
};
static char *get_ds(L4::Cap<L4Re::Dataspace> *_ds)
{
  *_ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
  if (!(*_ds).is_valid())
    {
      printf("Dataspace allocation failed.\n");
      return 0;
    }
  int err =  L4Re::Env::env()->mem_alloc()->alloc(DS_SIZE, *_ds, 0);
  if (err < 0)
    {
      printf("mem_alloc->alloc() failed.\n");
      L4Re::Util::cap_alloc.free(*_ds);
      return 0;
    }
  /*
   * Attach DS to local address space
   */
  char *_addr = 0;
  err =  L4Re::Env::env()->rm()->attach(&_addr, (*_ds)->size(),
                                        L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                        L4::Ipc::make_cap_rw(*_ds));
  if (err < 0)
    {
      printf("Error attaching data space: %s\n", l4sys_errtostr(err));
      L4Re::Util::cap_alloc.free(*_ds);
      return 0;
    }
  /*
   * Success! Write something to DS.
   */
  printf("Attached DS\n");
  static char const * const msg = "[DS] Hello from server!";
  snprintf(_addr, strlen(msg) + 1, msg);
  return _addr;
}
int main()
{
  L4::Cap<L4Re::Dataspace> ds;
  char *addr;
  if (!(addr = get_ds(&ds)))
    return 2;
  // First the IRQ handler, because we need it in the My_server_obj object
  Shm_observer observer(addr);
  // Registering the observer as an IRQ handler, this allocates an
  // IRQ object using the factory of our server.
  L4::Cap<L4::Irq> irq = server.registry()->register_irq_obj(&observer);
  // Now the initial server object shared with the client via our parent.
  // it provides the data-space and the IRQ capabilities to a client.
  My_server_obj server_obj(ds, irq);
  // Registering the server object to the capability 'shm' in our the L4Re::Env.
  // This capability must be provided by the parent. (see the shared_ds.lua)
  server.registry()->register_obj(&server_obj, "shm");
  // Run our server loop.
  server.loop();
  return 0;
}
```

17.17 examples/libs/l4re/c++/shared_ds/shared_ds.cfg

Sharing memory between applications, configuration file.

```
-- Include L4 functionality
local L4 = require("L4");
-- Create a channel from the client to the server
local channel = L4.default_loader:new_channel();
-- Start the server, giving the channel with full server rights.
```

```
-- The server will have a yellow log output.
L4.default_loader:start(
  {
    caps = { shm = channel:svr() },
    log  = { "server", "yellow" }
  },
  "rom/ex_l4re_ds_srv"
);
-- Start the client, giving it the channel with read only rights. The
-- log output will be green.
L4.default_loader:start(
  {
    caps = { shm = channel },
    log  = { "client", "green"  },
    l4re_dbg = L4.Dbg.Warn
  },
  "rom/ex_l4re_ds_clnt"
);
```

17.18 examples/libs/l4re/streammap/server.cc

Client/Server example showing how to map a page to another task – Server implementation. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/cxx/ipc_server>
#include "shared.h"
static char page_to_map[L4_PAGESIZE] __attribute__((aligned(L4_PAGESIZE)));
static L4Re::Util::Registry_server<> server;
class Smap_server : public L4::Server_object_t<Mapper>
{
public:
  int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};
int
Smap_server::dispatch(l4_umword_t, L4::Ipc::Iostream &ios)
{
  l4_msgtag_t t;
  ios » t;
  // We're only talking the Map_example protocol
  if (t.label() != Mapper::Protocol)
    return -L4_EBADPROTO;
  L4::Opcode opcode;
  ios » opcode;
  switch (opcode)
    {
    case Mapper::Do_map:
      l4_addr_t snd_base;
      ios » snd_base;
      // put something into the page to read it out at the other side
      snprintf(page_to_map, sizeof(page_to_map), "Hello from the server!");
      printf("Sending to client\n");
      // send page
      ios « L4::Ipc::Snd_fpage::mem((l4_addr_t)page_to_map, L4_PAGESHIFT,
                                    L4_FPAGE_RO, snd_base);
      return L4_EOK;
    default:
      return -L4_ENOSYS;
    }
}
int
main()
{
  static Smap_server smap;
  // Register server
  if (!server.registry()->register_obj(&smap, "smap").is_valid())
    {
      printf("Could not register my service, read-only namespace?\n");
      return 1;
    }
```

```
  printf("Welcome to the memory map example server!\n");
  // Wait for client requests
  server.loop();
  return 0;
}
```

17.19 examples/libs/l4re/streammap/client.cc

Client/Server example showing how to map a page to another task – Client implementation. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/err.h>
#include <l4/sys/types.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/cxx/ipc_stream>
#include <stdio.h>
#include "shared.h"
static int
func_smap_call(L4::Cap<void> const &server)
{
  L4::Ipc::Iostream s(l4_utcb());
  l4_addr_t addr = 0;
  int err;
  if ((err = L4Re::Env::env()->rm()->reserve_area(&addr, L4_PAGESIZE,
                                                  L4Re::Rm::F::Search_addr)))
    {
      printf("The reservation of one page within our virtual memory failed with %d\n", err);
      return 1;
    }
  s « L4::Opcode(Mapper::Do_map)
    « (l4_addr_t)addr;
  s « L4::Ipc::Rcv_fpage::mem((l4_addr_t)addr, L4_PAGESHIFT, 0);
  int r = l4_error(s.call(server.cap(), Mapper::Protocol));
  if (r)
    return r; // failure
  printf("String sent by server: %s\n", (char *)addr);
  return 0; // ok
}
int
main()
{
  L4::Cap<void> server = L4Re::Env::env()->get_cap<void>("smap");
  if (!server.is_valid())
    {
      printf("Could not get capability slot!\n");
      return 1;
    }
  printf("Asking for page from server\n");
  if (func_smap_call(server))
    {
      printf("Error talking to server\n");
      return 1;
    }
  printf("It worked!\n");
  L4Re::Util::cap_alloc.free(server, L4Re::This_task);
  return 0;
}
```

17.20 examples/libs/l4re/streammap/streammap.cfg

Sample configuration file for the client/server map example.

```
-- vim:set ft=lua:
-- Include L4 functionality
local L4 = require("L4");
-- Channel for the communication between the server and the client.
local smap_channel = L4.default_loader:new_channel();
```

```
-- The server program, using the 'smap' channel in server
-- mode. The log prefix will be 'server', colored yellow.
L4.default_loader:start({ caps = { smap = smap_channel:svr() },
                          log = { "server", "yellow" }},
                        "rom/ex_smap-server");
-- The client program.
-- It is given the 'smap' channel to be able to talk to the server.
-- The log prefix will be 'client', colored green.
L4.default_loader:start({ caps = { smap = smap_channel },
                          log = { "client", "green" } },
                        "rom/ex_smap-client");
```

17.21 examples/libs/libirq/loop.c

libirq usage example using a self-created thread.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/irq/irq.h>
#include <l4/util/util.h>
#include <stdio.h>
#include <pthread.h>
enum { IRQ_NO = 17 };
static void isr_handler(void)
{
  printf("Got IRQ %d\n", IRQ_NO);
}
static void *isr_thread(void *data)
{
  l4irq_t *irq;
  (void)data;
  if (!(irq = l4irq_attach(IRQ_NO)))
    return NULL;
  while (1)
    {
      if (l4irq_wait(irq))
        continue;
      isr_handler();
    }
  return NULL;
}
int main(void)
{
  pthread_t thread;
  if (pthread_create(&thread, NULL, isr_thread, NULL))
    return 1;
  l4_sleep_forever();
  return 0;
}
```

17.22 examples/libs/libirq/async_isr.c

libirq usage example using asynchronous ISR handler functionality.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * This example shall show how to use the libirq.
 */
#include <l4/irq/irq.h>
#include <l4/util/util.h>
#include <stdio.h>
enum { IRQ_NO = 17 };
static void isr_handler(void *data)
{
```

```
  (void)data;
  printf("Got IRQ %d\n", IRQ_NO);
}
int main(void)
{
  const int seconds = 5;
  l4irq_t *irqdesc;
  if (!(irqdesc = l4irq_request(IRQ_NO, isr_handler, 0, 0xff, 0)))
    {
      printf("Requesting IRQ %d failed\n", IRQ_NO);
      return 1;
    }
  printf("Attached to key IRQ %d\nPress keys now, will terminate in %d seconds\n",
         IRQ_NO, seconds);
  l4_sleep(seconds * 1000);
  if (l4irq_release(irqdesc))
    {
      printf("Failed to release IRQ\n");
      return 1;
    }
  printf("Bye\n");
  return 0;
}
```

17.23 examples/sys/migrate/thread_migrate.cc

Thread migration example.

```
/*
 * (c) 2008-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/scheduler>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <pthread-l4.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
enum { NR_THREADS = 12 };
static L4::Cap<L4::Thread> threads[NR_THREADS];
static l4_umword_t         cpu_map, cpu_nrs;
/* Function for the threads. The content is not really relevant, so lets
 * just sleep around a bit. */
static void *thread_fn(void *)
{
  while (1)
    sleep(1);
  return 0;
}
/* Check how many CPUs we have available.
 */
static int check_cpus(void)
{
  l4_sched_cpu_set_t cs = l4_sched_cpu_set(0, 0);
  if (l4_error(L4Re::Env::env()->scheduler()->info(&cpu_nrs, &cs)) < 0)
    return 1;
  cpu_map = cs.map;
  printf("%ld maximal supported CPUs.\n", cpu_nrs);
  if (cpu_nrs >= L4_MWORD_BITS)
    {
      printf("Will only handle %ld CPUs.\n", cpu_nrs);
      cpu_nrs = L4_MWORD_BITS;
    }
  else if (cpu_nrs == 1)
    printf("Only found 1 CPU.\n");
  return cpu_nrs < 2;
}
/* Create a couple of threads and store their capabilities in an array */
static int create_threads(void)
{
  unsigned i;
  for (i = 0; i < NR_THREADS; ++i)
    {
      pthread_t t;
      if (pthread_create(&t, NULL, thread_fn, NULL))
        return 1;
      threads[i] = L4::Cap<L4::Thread>(pthread_l4_cap(t));
```

```
    }
  printf("Created %d threads.\n", NR_THREADS);
  return 0;
}
/* Helper function to get the next CPU */
static unsigned get_next_cpu(unsigned c)
{
  unsigned x = c;
  for (;;)
    {
      x = (x + 1) % cpu_nrs;
      if (L4Re::Env::env()->scheduler()->is_online(x))
        return x;
      if (x == c)
        return c;
    }
}
/* Function that shuffles the threads on the available CPUs */
static void shuffle(void)
{
  unsigned start = 0;
  while (1)
    {
      unsigned t;
      unsigned c = start;
      for (t = 0; t < NR_THREADS; ++t)
        {
          l4_sched_param_t sp = l4_sched_param(20);
          c = get_next_cpu(c);
          sp.affinity = l4_sched_cpu_set(c, 0);
          if (l4_error(L4Re::Env::env()->scheduler()->run_thread(threads[t], sp)))
            printf("Error migrating thread%02d to CPU%02d\n", t, c);
          printf("Migrated Thread%02d -> CPU%02d\n", t, c);
        }
      start++;
      if (start == cpu_nrs)
        start = 0;
      sleep(1);
    }
}
int main(void)
{
  if (check_cpus())
    return 1;
  if (create_threads())
    return 1;
  shuffle();
  return 0;
}
```

17.24 examples/sys/migrate/thread_migrate.cfg

Sample configuration file for the thread migration example.

```
-- vim:set ft=lua:
local L4 = require("L4");
-- The log prefix will be 'migrate', colored green.
L4.default_loader:start({ log = { "migrate", "green" } },
                        "rom/ex_thread_migrate");
```

17.25 tmpfs/lib/src/fs.cc

Example file system for L4Re::Vfs.

```
/*
 * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *          Alexander Warg <warg@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */
#include <l4/l4re_vfs/backend>
#include <l4/cxx/string>
#include <l4/cxx/avl_tree>
#include <sys/stat.h>
#include <sys/ioctl.h>
```

```
#include <dirent.h>
#include <cstdio>
#include <cstdlib>
#include <cstring>
namespace {
using namespace L4Re::Vfs;
using cxx::Ref_ptr;
class File_data
{
public:
  File_data() : _buf(0), _size(0) {}
  unsigned long put(unsigned long offset,
                    unsigned long bufsize, void *srcbuf);
  unsigned long get(unsigned long offset,
                    unsigned long bufsize, void *dstbuf);
  unsigned long size(unsigned long offset);
  unsigned long size() const { return _size; }
  ~File_data() throw() { free(_buf); }
private:
  void *_buf;
  unsigned long _size;
};
unsigned long
File_data::put(unsigned long offset, unsigned long bufsize, void *srcbuf)
{
  if (offset + bufsize > _size)
    size(offset + bufsize);
  if (!_buf)
    return 0;
  memcpy((char *)_buf + offset, srcbuf, bufsize);
  return bufsize;
}
unsigned long
File_data::get(unsigned long offset, unsigned long bufsize, void *dstbuf)
{
  unsigned long s = bufsize;
  if (offset > _size)
    return 0;
  if (offset + bufsize > _size)
    s = _size - offset;
  memcpy(dstbuf, (char *)_buf + offset, s);
  return s;
}
unsigned long
File_data::size(unsigned long offset)
{
  if (offset != _size)
    {
      _size = offset;
      _buf = realloc(_buf, _size);
    }
  if (_buf)
    return 0;
  return -ENOSPC;
}
class Node : public cxx::Avl_tree_node
{
public:
  Node(const char *path, mode_t mode)
    : _ref_cnt(0), _path(strdup(path))
  {
    memset(&_info, 0, sizeof(_info));
    _info.st_mode = mode;
  }
  const char *path() const { return _path; }
  struct stat64 *info() { return &_info; }
  void add_ref() throw() { ++_ref_cnt; }
  int remove_ref() throw() { return --_ref_cnt; }
  bool is_dir() const { return S_ISDIR(_info.st_mode); }
  virtual ~Node() { free(_path); }
private:
  int           _ref_cnt;
  char         *_path;
  struct stat64 _info;
};
struct Node_get_key
{
  typedef cxx::String Key_type;
  static Key_type key_of(Node const *n)
  { return n->path(); }
};
struct Path_avl_tree_compare
{
  bool operator () (const char *l, const char *r) const
  { return strcmp(l, r) < 0; }
  bool operator () (const cxx::String l, const cxx::String r) const
  {
```

```
    int v = strncmp(l.start(), r.start(), cxx::min(l.len(), r.len()));
    return v < 0 || (v == 0 && l.len() < r.len());
  }
};
class Pers_file : public Node
{
public:
  Pers_file(const char *name, mode_t mode)
    : Node(name, (mode & 0777) | __S_IFREG) {}
  File_data const &data() const { return _data; }
  File_data &data() { return _data; }
private:
  File_data     _data;
};
class Pers_dir : public Node
{
private:
  typedef cxx::Avl_tree<Node, Node_get_key, Path_avl_tree_compare> Tree;
  Tree _tree;
public:
  Pers_dir(const char *name, mode_t mode)
    : Node(name, (mode & 0777) | __S_IFDIR) {}
  Ref_ptr<Node> find_path(cxx::String);
  bool add_node(Ref_ptr<Node> const &);
  typedef Tree::Const_iterator Const_iterator;
  Const_iterator begin() const { return _tree.begin(); }
  Const_iterator end() const { return _tree.end(); }
};
Ref_ptr<Node> Pers_dir::find_path(cxx::String path)
{
  return cxx::ref_ptr(_tree.find_node(path));
}
bool Pers_dir::add_node(Ref_ptr<Node> const &n)
{
  bool e = _tree.insert(n.ptr()).second;
  if (e)
    n->add_ref();
  return e;
}
class Tmpfs_dir : public Be_file
{
public:
  explicit Tmpfs_dir(Ref_ptr<Pers_dir> const &d) throw()
    : _dir(d), _getdents_state(false) {}
  int get_entry(const char *, int, mode_t, Ref_ptr<File> *) throw();
  ssize_t getdents(char *, size_t) throw();
  int fstat64(struct stat64 *buf) const throw();
  int utime(const struct utimbuf *) throw();
  int fchmod(mode_t) throw();
  int mkdir(const char *, mode_t) throw();
  int unlink(const char *) throw();
  int rename(const char *, const char *) throw();
  int faccessat(const char *, int, int) throw();
private:
  int walk_path(cxx::String const &_s,
                Ref_ptr<Node> *ret, cxx::String *remaining = 0);
  Ref_ptr<Pers_dir> _dir;
  bool _getdents_state;
  Pers_dir::Const_iterator _getdents_iter;
};
class Tmpfs_file : public Be_file_pos
{
public:
  explicit Tmpfs_file(Ref_ptr<Pers_file> const &f) throw()
    : Be_file_pos(), _file(f) {}
  off64_t size() const throw();
  int fstat64(struct stat64 *buf) const throw();
  int ftruncate64(off64_t p) throw();
  int ioctl(unsigned long, va_list) throw();
  int utime(const struct utimbuf *) throw();
  int fchmod(mode_t) throw();
private:
  ssize_t preadv(const struct iovec *v, int iovcnt, off64_t p) throw();
  ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t p) throw();
  Ref_ptr<Pers_file> _file;
};
ssize_t Tmpfs_file::preadv(const struct iovec *v, int iovcnt, off64_t p) throw()
{
  if (iovcnt < 0)
    return -EINVAL;
  ssize_t sum = 0;
  for (int i = 0; i < iovcnt; ++i)
    {
      sum  += _file->data().get(p, v[i].iov_len, v[i].iov_base);
      p += v[i].iov_len;
    }
  return sum;
```

```
}
ssize_t Tmpfs_file::pwritev(const struct iovec *v, int iovcnt, off64_t p) throw()
{
  if (iovcnt < 0)
    return -EINVAL;
  ssize_t sum = 0;
  for (int i = 0; i < iovcnt; ++i)
    {
      sum  += _file->data().put(p, v[i].iov_len, v[i].iov_base);
      p += v[i].iov_len;
    }
  return sum;
}
int Tmpfs_file::fstat64(struct stat64 *buf) const throw()
{
  _file->info()->st_size = _file->data().size();
  memcpy(buf, _file->info(), sizeof(*buf));
  return 0;
}
int Tmpfs_file::ftruncate64(off64_t p) throw()
{
  if (p < 0)
      return -EINVAL;
  if (_file->data().size(p) == 0)
      return 0;
  return -EIO; // most likely ENOSPC, but can't report that
}
off64_t Tmpfs_file::size() const throw()
{ return _file->data().size(); }
int
Tmpfs_file::ioctl(unsigned long v, va_list args) throw()
{
  switch (v)
    {
    case FIONREAD: // return amount of data still available
      int *available = va_arg(args, int *);
      *available = _file->data().size() - pos();
      return 0;
    };
  return -EINVAL;
}
int
Tmpfs_file::utime(const struct utimbuf *times) throw()
{
  _file->info()->st_atime = times->actime;
  _file->info()->st_mtime = times->modtime;
  return 0;
}
int
Tmpfs_file::fchmod(mode_t m) throw()
{
  _file->info()->st_mode = m;
  return 0;
}
int
Tmpfs_dir::faccessat(const char *path, int mode, int) throw()
{
  Ref_ptr<Node> node;
  cxx::String name = path;
  int err = walk_path(name, &node, &name);
  if (err < 0)
    return err;
  if (mode == F_OK) // existence check
    return 0;
  struct stat64 *stats = node->info();
  if ((mode & R_OK) && !(stats->st_mode & S_IRUSR))
    return -EACCES;
  if ((mode & W_OK) && !(stats->st_mode & S_IWUSR))
    return -EACCES;
  if ((mode & X_OK) && !(stats->st_mode & S_IXUSR))
    return -EACCES;
  return 0;
}
int
Tmpfs_dir::get_entry(const char *name, int flags, mode_t mode,
                     Ref_ptr<File> *file) throw()
{
  Ref_ptr<Node> path;
  if (!*name)
    {
      *file = cxx::ref_ptr(this);
      return 0;
    }
  cxx::String n = name;
  int e = walk_path(n, &path, &n);
  if (e == -ENOTDIR)
    return e;
```

```
  if (!(flags & O_CREAT) && e < 0)
    return e;
  if ((flags & O_CREAT) && e == -ENOENT)
    {
      Ref_ptr<Node> node(new Pers_file(n.start(), mode));
      // when ENOENT is return, path is always a directory
      bool e = cxx::ref_ptr_static_cast<Pers_dir>(path)->add_node(node);
      if (!e)
        return -ENOMEM;
      path = node;
    }
  if (path->is_dir())
    *file = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr_static_cast<Pers_dir>(path)));
  else
    *file = cxx::ref_ptr(new Tmpfs_file(cxx::ref_ptr_static_cast<Pers_file>(path)));
  if (!*file)
    return -ENOMEM;
  return 0;
}
ssize_t
Tmpfs_dir::getdents(char *buf, size_t sz) throw()
{
  struct dirent64 *d = (struct dirent64 *)buf;
  ssize_t ret = 0;
  if (!_getdents_state)
    {
      _getdents_iter = _dir->begin();
      _getdents_state = true;
    }
  else if (_getdents_iter == _dir->end())
    {
      _getdents_state = false;
      return 0;
    }
  for (; _getdents_iter != _dir->end(); ++_getdents_iter)
    {
      unsigned l = strlen(_getdents_iter->path()) + 1;
      if (l > sizeof(d->d_name))
        l = sizeof(d->d_name);
      unsigned n = offsetof (struct dirent64, d_name) + l;
      n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);
      if (n > sz)
        break;
      d->d_ino = 1;
      d->d_off = 0;
      memcpy(d->d_name, _getdents_iter->path(), l);
      d->d_reclen = n;
      d->d_type   = DT_REG;
      ret += n;
      sz  -= n;
      d    = (struct dirent64 *)((unsigned long)d + n);
    }
  return ret;
}
int
Tmpfs_dir::fstat64(struct stat64 *buf) const throw()
{
  memcpy(buf, _dir->info(), sizeof(*buf));
  return 0;
}
int
Tmpfs_dir::utime(const struct utimbuf *times) throw()
{
  _dir->info()->st_atime = times->actime;
  _dir->info()->st_mtime = times->modtime;
  return 0;
}
int
Tmpfs_dir::fchmod(mode_t m) throw()
{
  _dir->info()->st_mode = m;
  return 0;
}
int
Tmpfs_dir::walk_path(cxx::String const &_s,
                     Ref_ptr<Node> *ret, cxx::String *remaining)
{
  Ref_ptr<Pers_dir> p = _dir;
  cxx::String s = _s;
  Ref_ptr<Node> n;
  while (1)
    {
      if (s.len() == 0)
        {
          *ret = p;
          return 0;
        }
```

```
      cxx::String::Index sep = s.find("/");
      if (sep - s.start() == 1 && *s.start() == '.')
        {
          s = s.substr(s.start() + 2);
          continue;
        }
      n = p->find_path(s.head(sep - s.start()));
      if (!n)
        {
          *ret = p;
          if (remaining)
            *remaining = s.head(sep - s.start());
          return -ENOENT;
        }
      if (sep == s.end())
        {
          *ret = n;
          return 0;
        }
      if (!n->is_dir())
        return -ENOTDIR;
      s = s.substr(sep + 1);
      p = cxx::ref_ptr_static_cast<Pers_dir>(n);
    }
  *ret = n;
  return 0;
}
int
Tmpfs_dir::mkdir(const char *name, mode_t mode) throw()
{
  Ref_ptr<Node> node = _dir;
  cxx::String p = cxx::String(name);
  cxx::String path, last = p;
  cxx::String::Index s = p.rfind("/");
  // trim /'s at the end
  while (p.len() && s == p.end() - 1)
    {
      p.len(p.len() - 1);
      s = p.rfind("/");
    }
  //printf("MKDIR '%s' p=%p %p\n", name, p.start(), s);
  if (s != p.end())
    {
      path = p.head(s);
      last = p.substr(s + 1, p.end() - s);
      int e = walk_path(path, &node);
      if (e < 0)
        return e;
    }
  if (!node->is_dir())
    return -ENOTDIR;
  // due to path walking we can end up with an empty name
  if (p.len() == 0 || p == cxx::String("."))
    return 0;
  Ref_ptr<Pers_dir> dnode = cxx::ref_ptr_static_cast<Pers_dir>(node);
  Ref_ptr<Pers_dir> dir(new Pers_dir(last.start(), mode));
  return dnode->add_node(dir) ? 0 : -EEXIST;
}
int
Tmpfs_dir::unlink(const char *name) throw()
{
  cxx::Ref_ptr<Node> n;
  int e = walk_path(name, &n);
  if (e < 0)
    return -ENOENT;
  printf("Unimplemented (if file exists): %s(%s)\n", __func__, name);
  return -ENOMEM;
}
int
Tmpfs_dir::rename(const char *old, const char *newn) throw()
{
  printf("Unimplemented: %s(%s, %s)\n", __func__, old, newn);
  return -ENOMEM;
}
class Tmpfs_fs : public Be_file_system
{
public:
  Tmpfs_fs() : Be_file_system("tmpfs") {}
  int mount(char const *source, unsigned long mountflags,
            void const *data, cxx::Ref_ptr<File> *dir) throw()
  {
    (void)mountflags;
    (void)source;
    (void)data;
    *dir = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr(new Pers_dir("root", 0777))));
    if (!*dir)
      return -ENOMEM;
```

```
    return 0;
  }
};
static Tmpfs_fs _tmpfs L4RE_VFS_FILE_SYSTEM_ATTRIBUTE;
}
```

17.26 examples/libs/shmc/prodcons.c

Simple shared memory example.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * This example uses shared memory between two threads, one producer, one
 * consumer.
 */
#include <l4/shmc/shmc.h>
#include <l4/util/util.h>
#include <stdio.h>
#include <string.h>
#include <pthread-l4.h>
#include <l4/sys/thread.h>
// a small helper
#define CHK(func) if (func) { printf("failure: %d\n", __LINE__); return (void *)-1; }
static const char some_data[] = "Hi consumer!";
static void *thread_producer(void *d)
{
  (void)d;
  l4shmc_chunk_t p_one;
  l4shmc_signal_t s_one, s_done;
  l4shmc_area_t shmarea;
  // attach this thread to the shm object
  CHK(l4shmc_attach("testshm", &shmarea));
  // add a chunk
  CHK(l4shmc_add_chunk(&shmarea, "one", 1024, &p_one));
  // add a signal
  CHK(l4shmc_add_signal(&shmarea, "prod", &s_one));
  CHK(l4shmc_attach_signal_to(&shmarea, "done",
                              pthread_l4_cap(pthread_self()), 10000, &s_done));
  // connect chunk and signal
  CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));
  printf("PRODUCER: ready\n");
  while (1)
    {
      while (l4shmc_chunk_try_to_take(&p_one))
        printf("Uh, should not happen!\n"); //l4_thread_yield();
      memcpy(l4shmc_chunk_ptr(&p_one), some_data, sizeof(some_data));
      CHK(l4shmc_chunk_ready_sig(&p_one, sizeof(some_data)));
      printf("PRODUCER: Sent data\n");
      CHK(l4shmc_wait_signal(&s_done));
    }
  l4_sleep_forever();
  return NULL;
}
static void *thread_consume(void *d)
{
  (void)d;
  l4shmc_area_t shmarea;
  l4shmc_chunk_t p_one;
  l4shmc_signal_t s_one, s_done;
  // attach to shared memory area
  CHK(l4shmc_attach("testshm", &shmarea));
  // get chunk 'one'
  CHK(l4shmc_get_chunk(&shmarea, "one", &p_one));
  // add a signal
  CHK(l4shmc_add_signal(&shmarea, "done", &s_done));
  // attach signal to this thread
  CHK(l4shmc_attach_signal_to(&shmarea, "prod",
                              pthread_l4_cap(pthread_self()), 10000, &s_one));
  // connect chunk and signal
  CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));
  while (1)
    {
      CHK(l4shmc_wait_chunk(&p_one));
      printf("CONSUMER: Received from chunk one: %s\n",
             (char *)l4shmc_chunk_ptr(&p_one));
```

```
      memset(l4shmc_chunk_ptr(&p_one), 0, l4shmc_chunk_size(&p_one));
      CHK(l4shmc_chunk_consumed(&p_one));
      CHK(l4shmc_trigger(&s_done));
    }
  return NULL;
}
int main(void)
{
  pthread_t one, two;
  // create new shared memory area, 8K in size
  if (l4shmc_create("testshm", 8192))
    return 1;
  // create two threads, one for producer, one for consumer
  pthread_create(&one, 0, thread_producer, 0);
  pthread_create(&two, 0, thread_consume, 0);
  // now sleep, the two threads are doing the work
  l4_sleep_forever();
  return 0;
}
```

Index

__Iface
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, 1099
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, 1103
__Iface_list
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, 1099
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, 1103
__L4UTIL_THREAD_FUNC
thread.h, 2372
__check_protocols__
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, 1100
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, 1104
__strcpy_maxlen
debugger.h, 2175
_bus
L4vbus::Device, 1557
~Be_file_system
L4Re::Vfs::Be_file_system, 1459
~H_list_item_t
cxx::H_list_item_t< ELEM_TYPE >, 782

a
L4Re::Video::Pixel_info, 1506
Access_width
L4Re::Mmio_space, 1354
acquire
L4Re::Inhibitor, 1342
add
L4::Ipc_svr::Timeout_queue, 1055
L4::Thread::Modify_senders, 1196
L4vcpu::State, 1599
L4virtio::Svr::Driver_mem_list_t< DATA >, 1697
add_block
L4virtio::Driver::Block_device, 1622
add_ku_mem
L4::Task, 1175
add_timeout
L4::Ipc_svr::Server_iface, 1047
L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >, 1063
alien
L4::Thread::Attr, 1193
align_to
L4::Ipc::Msg, 630
all
L4::Kip::Mem_desc, 1087, 1088
alloc
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 687
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 695
cxx::List_alloc, 790
cxx::Slab< Type, Slab_size, Max_free, Alloc >, 817
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >, 822
L4Re::Cap_alloc, 1296
L4Re::Mem_alloc, 1350
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, 1402, 1404
alloc_buffer_demand
L4::Ipc_svr::Br_manager_no_buffers, 1031
L4::Ipc_svr::Server_iface, 1047
L4Re::Util::Br_manager, 1391
alloc_descriptor
L4virtio::Driver::Virtqueue, 1644
alloc_dynamic_entry
L4Re::Util::Names::Name_space, 1431
alloc_fd
L4Re::Vfs::Fs, 1472
alloc_max
cxx::List_alloc, 790
allocate
L4Re::Dataspace, 1303
L4Re::Util::Dataspace_svr, 1409
amd64 Virtual Registers (UTCB), 592
amd64/l4/sys/cache.h, 2119, 2120
amd64/l4/sys/consts.h, 2134
amd64/l4/sys/l4int.h, 2227, 2228
amd64/l4/sys/linkage.h, 1828
amd64/l4/sys/segment.h, 1809, 1812
amd64/l4/sys/utcb.h, 2289, 2290
amd64/l4/util/apic.h, 1763, 1764
amd64/l4/util/bitops_arch.h, 1834
amd64/l4/util/cpu.h, 1842, 1843
amd64/l4/util/idt.h, 1774, 1775
amd64/l4/util/irq.h, 1933, 1935
amd64/l4/util/l4_macros.h, 1846, 1847
amd64/l4/util/mbi_argv.h, 1850, 1851
amd64/l4/util/perform.h, 1777, 1778
amd64/l4/util/port_io.h, 1819
amd64/l4/util/rdtsc.h, 1789, 1791
amd64/l4/util/spin.h, 1798
amd64/l4/util/util.h, 1800, 1802

amd64/l4f/l4/sys/segment.h, 1806, 1808
amd64/l4f/l4/util/port_io.h, 1818, 1819
Arch
L4::Kip::Mem_desc, 1087
Area
L4Re::Rm, 1374
ARM Virtual Registers (UTCB), 109
arm/l4/sys/atomic.h, 2318, 2319
arm/l4/sys/cache.h, 2116, 2118
arm/l4/sys/consts.h, 2133
arm/l4/sys/l4int.h, 2227
arm/l4/sys/linkage.h, 1827
arm/l4/sys/mem_op.h, 1830, 1831
arm/l4/sys/thread.h, 2382
arm/l4/sys/utcb.h, 2286, 2288
arm/l4/sys/vm.h, 1832
arm/l4/util/bitops_arch.h, 1833, 1834
arm/l4/util/cpu.h, 1841
arm/l4/util/irq.h, 1931, 1932
arm/l4/util/l4_macros.h, 1846
arm/l4/util/mbi_argv.h, 1849
arm/l4f/l4/sys/ipc.h, 2206, 2207
arm/l4f/l4/sys/syscall_defs.h, 1853
arm_smccc.h
l4_arm_smccc_call, 2112
assert.h
l4_assert, 2310
assign_dma_domain
L4vbus::Vbus, 1590, 1591
associate
L4Re::Dma_space, 1317
AT_BASE
ELF binary format, 193
AT_EGID
ELF binary format, 193
AT_ENTRY
ELF binary format, 193
AT_EUID
ELF binary format, 193
AT_EXECFD
ELF binary format, 193
AT_FLAGS
ELF binary format, 193
AT_GID
ELF binary format, 193
AT_IGNORE
ELF binary format, 193
AT_L4_AUX
ELF binary format, 193
AT_L4_ENV
ELF binary format, 193
AT_NOTELF
ELF binary format, 193
AT_NULL
ELF binary format, 193
AT_PAGESZ
ELF binary format, 193
AT_PHDR
ELF binary format, 193
AT_PHENT
ELF binary format, 193
AT_PHNUM
ELF binary format, 193
AT_UID
ELF binary format, 193
Atomic Instructions, 110
l4util_add16, 112
l4util_add16_res, 112
l4util_add32, 112
l4util_add32_res, 113
l4util_add8, 113
l4util_add8_res, 113
l4util_and16, 114
l4util_and16_res, 114
l4util_and32, 114
l4util_and32_res, 115
l4util_and8, 115
l4util_and8_res, 115
l4util_atomic_add, 116
l4util_atomic_inc, 116
l4util_cmpxchg, 116
l4util_cmpxchg16, 117
l4util_cmpxchg32, 118
l4util_cmpxchg64, 118
l4util_cmpxchg8, 119
l4util_dec16, 119
l4util_dec16_res, 120
l4util_dec32, 120
l4util_dec32_res, 120
l4util_dec8, 121
l4util_dec8_res, 121
l4util_inc16, 121
l4util_inc16_res, 121
l4util_inc32, 122
l4util_inc32_res, 122
l4util_inc8, 122
l4util_inc8_res, 123
l4util_or16, 123
l4util_or16_res, 123
l4util_or32, 124
l4util_or32_res, 124
l4util_or8, 124
l4util_or8_res, 125
l4util_sub16, 125
l4util_sub16_res, 125
l4util_sub32, 126
l4util_sub32_res, 126
l4util_sub8, 127
l4util_sub8_res, 127
l4util_xchg, 127
l4util_xchg16, 128
l4util_xchg32, 128
l4util_xchg8, 128
attach
L4Re::Rm, 1375, 1376
L4Re::Util::Event_buffer_t< PAYLOAD >, 1419

Attach_flags
L4Re::Rm::F, 1384
Attach_mask
L4Re::Rm::F, 1385
Attr
L4::Thread::Attr, 1192
Attribute
L4Re::Dma_space, 1316
Attributes
L4Re::Dma_space, 1315
atype
Elf32_Auxv, 836
Elf64_Auxv, 847
Auto_ptr
cxx::Auto_ptr< T >, 666, 667
Auxiliary data, 129
avail
cxx::List_alloc, 791
avail_align
L4virtio::Virtqueue, 1729
avail_size
L4virtio::Virtqueue, 1729
Avl_map
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, 671
ax
l4_vcpu_regs_t, 1282

b
L4Re::Video::Pixel_info, 1507
backtrace.h
l4util_backtrace, 2321
Bad_address
L4virtio::Svr::Bad_descriptor, 1655
Bad_descriptor
L4virtio::Svr::Bad_descriptor, 1655
Bad_flags
L4virtio::Svr::Bad_descriptor, 1655
Bad_next
L4virtio::Svr::Bad_descriptor, 1655
Bad_rights
L4virtio::Svr::Bad_descriptor, 1655
Bad_size
L4virtio::Svr::Bad_descriptor, 1655
Base API, 130
Base_avl_set
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 733
Basic Macros, 132
l4_align_stack_for_direct_fncall, 135
L4_ALWAYS_INLINE, 134
L4_HIDDEN, 134
L4_NOTHROW, 134
Be_file_system
L4Re::Vfs::Be_file_system, 1458
begin
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 734, 735
cxx::Bits::Bst< Node, Get_key, Compare >, 752, 753
Bidirectional
L4Re::Dma_space, 1316
bind
L4::Icu, 927
L4::Iommu, 942
L4::Thread::Attr, 1193
bind_notification_irq
L4virtio::Driver::Device, 1631
bind_thread
L4::Rcv_endpoint, 1134
bit
cxx::Bitmap_base, 720
Bit Manipulation, 135
l4util_bsf, 136
l4util_bsr, 137
l4util_btc, 137
l4util_btr, 138
l4util_bts, 139
l4util_clear_bit, 139
l4util_complement_bit, 140
l4util_find_first_set_bit, 140
l4util_find_first_zero_bit, 141
l4util_next_power2, 141
l4util_set_bit, 142
l4util_test_bit, 142
bit_index
cxx::Bitmap_base, 721
Bitmap graphics and fonts, 143
bitmap.h
gfxbitmap_bmap, 1946
gfxbitmap_color_pix_t, 1945
gfxbitmap_color_t, 1946
gfxbitmap_convert_color, 1947
gfxbitmap_copy, 1947
gfxbitmap_fill, 1947
gfxbitmap_set, 1948
pSLIM_BMAP_START_LSB, 1945
Bits
cxx::Bitfield< T, LSB, MSB >, 700
bits_per_pixel
L4Re::Video::Pixel_info, 1507
Bits_type
cxx::Bitfield< T, LSB, MSB >, 699
Block_dev_base
L4virtio::Svr::Block_dev_base< Ds_data >, 1660
Bootloader
L4::Kip::Mem_desc, 1087
bp
l4_vcpu_regs_t, 1283
Br_bytes
L4::Ipc::Msg, 629
buf
L4Re::Util::Event_buffer_t< PAYLOAD >, 1420
buf_cp_in
L4::Ipc, 622
buf_cp_out

L4::Ipc, 623
buf_in
L4::Ipc, 623
buffer
L4Re::Util::Event_t< PAYLOAD >, 1424
Buffer Registers (BRs), 143
L4_BDR_IO_SHIFT, 144
L4_BDR_MEM_SHIFT, 144
L4_BDR_OBJ_SHIFT, 144
l4_buffer_desc_consts_t, 144
Bufferable
L4Re::Dataspace::F, 1310
bus_cap
L4vbus::Device, 1551
bx
l4_vcpu_regs_t, 1283
bytes_per_pixel
L4Re::Video::Pixel_info, 1508

c
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, 1100
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, 1104
C++ Exceptions, 145
C++ IPC Interface Definition., 146
C_bits
cxx::Bitmap_base, 719
Cache Consistency, 149
l4_cache_clean_data, 150
l4_cache_coherent, 150
l4_cache_dma_coherent, 151
l4_cache_flush_data, 151
l4_cache_inv_data, 152
Cache_buffered
L4Re::Rm::F, 1385
Cache_normal
L4Re::Rm::F, 1385
Cache_uncached
L4Re::Rm::F, 1385
Cacheable
L4Re::Dataspace::F, 1310
Caching_mask
L4Re::Dataspace::F, 1310
L4Re::Rm::F, 1385
Caching_shift
L4Re::Dataspace::F, 1310
L4Re::Rm, 1375
call
L4::Arm_smccc, 859
L4::Ipc::Iostream, 965
Cap
L4::Cap< T >, 870, 871
L4::Ipc::Cap< T >, 956
cap
L4::Cap_base, 877
L4::Invalid_capability, 936
L4::Kobject, 1095
cap_alloc
L4Re Capability API, 368
Cap_base
L4::Cap_base, 876
cap_cast
L4, 614
cap_dynamic_cast
L4, 615
cap_equal
L4::Task, 1176
Cap_mask
L4::Ipc::Cap< T >, 956
cap_received
L4::Ipc::Gen_fpage< T >, 959
cap_reinterpret_cast
L4, 616, 617
Cap_type
L4::Cap_base, 875
cap_valid
L4::Task, 1176
Capabilities, 152
L4_BASE_ARM_SMCCC_CAP, 154
L4_BASE_CAPS_LAST, 154
L4_BASE_DEBUGGER_CAP, 154
L4_BASE_FACTORY_CAP, 154
L4_BASE_ICU_CAP, 154
L4_BASE_IOMMU_CAP, 154
L4_BASE_LOG_CAP, 154
L4_BASE_PAGER_CAP, 154
L4_BASE_SCHEDULER_CAP, 154
L4_BASE_TASK_CAP, 154
L4_BASE_THREAD_CAP, 154
l4_cap_consts_t, 153
L4_CAP_MASK, 154
L4_CAP_SHIFT, 154
L4_CAP_SIZE, 154
l4_capability_equal, 155
l4_default_caps_t, 154
L4_INVALID_CAP, 154
l4_is_invalid_cap, 155
l4_is_valid_cap, 156
capability
L4_DISABLE_COPY, 2123
Capability allocator, 156
l4re_util_cap_last, 157
Caps
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, 1207
cast
L4vcpu::Vcpu, 1604
cfg_read
L4vbus::Pci_dev, 1578
L4vbus::Pci_host_bridge, 1583
cfg_write
L4vbus::Pci_dev, 1578
L4vbus::Pci_host_bridge, 1584
chain
L4::Irq_mux, 1079
change_queue_config

L4virtio::Svr::Dev_config, 1674
check_size
L4::Ipc::Msg, 631, 632
chkcap
L4Re, 641
chkipc
L4Re, 642
chksys
L4Re, 643, 645, 646
Chunks, 157
l4shmc_add_chunk, 158
l4shmc_chunk_capacity, 159
l4shmc_chunk_ptr, 159
l4shmc_chunk_signal, 160
l4shmc_get_chunk, 160
l4shmc_get_chunk_to, 160
l4shmc_iterate_chunk, 161
Class
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, 1099
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, 1103
clear
cxx::Bits::Basic_list< POLICY >, 748
L4::Types::Flags< BITS_ENUM, UNDERLYING >, 1233
L4Re::Dataspace, 1304
L4Re::Util::Dataspace_svr, 1410
L4vcpu::State, 1600
clear_bit
cxx::Bitmap_base, 721
Coherent
L4Re::Dma_space, 1317
Color_component
L4Re::Video::Color_component, 1492
Com_error
L4::Com_error, 886
Comfortable Command Line Parsing, 162
parse_cmdline, 162
config_get
L4vbus::Gpio_pin, 1567
config_pad
L4vbus::Gpio_module, 1560
L4vbus::Gpio_pin, 1568
config_pull
L4vbus::Gpio_pin, 1568
config_queue
L4virtio::Device, 1615
L4virtio::Driver::Device, 1632
Console API, 164
consumed
L4virtio::Svr::Virtqueue, 1720
Consumer, 165, 168
l4shmc_chunk_consumed, 165
l4shmc_chunk_size, 166
l4shmc_enable_chunk, 166
l4shmc_enable_signal, 169
l4shmc_is_chunk_ready, 167
l4shmc_wait_any, 169
l4shmc_wait_any_to, 170
l4shmc_wait_any_try, 170
l4shmc_wait_chunk, 167
l4shmc_wait_chunk_to, 167
l4shmc_wait_chunk_try, 168
l4shmc_wait_signal, 171
l4shmc_wait_signal_to, 171
l4shmc_wait_signal_try, 171
contains
L4virtio::Svr::Driver_mem_region_t< DATA >, 1706
Continuous
L4Re::Mem_alloc, 1350
contrib/libio-io/l4/io/io.h, 1853, 1856
control
L4::Thread, 1184
Conventional
L4::Kip::Mem_desc, 1087
copy
L4::Cap< T >, 872
L4::Cap_base, 877
L4Re::Util::Dataspace_svr, 1410
copy_in
L4Re::Dataspace, 1304
copy_receive_cap
L4Re::Util::Names::Name_space, 1431
copy_to
L4virtio::Svr::Data_buffer, 1669
count
L4::Kip::Mem_desc, 1089
Counting_cap_alloc
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, 1402
CPU related functions, 146
l4util_cpu_capabilities, 147
l4util_cpu_capabilities_nocheck, 147
l4util_cpu_has_cpuid, 148
cpu_allow_shutdown
L4::Platform_control, 1125
cpu_disable
L4::Platform_control, 1125
cpu_enable
L4::Platform_control, 1125
create
L4::Factory, 912, 913
create_buffer
L4Re::Video::Goos, 1499
create_factory
L4::Factory, 914
create_gate
L4::Factory, 915
create_task
L4::Factory, 916
create_view
L4Re::Video::Goos, 1499
current_flags
L4virtio::Svr::Request_processor, 1712

cx
l4_vcpu_regs_t, 1283
cxx, 607
cxx::Auto_ptr< T >, 665
Auto_ptr, 666, 667
get, 667
operator=, 667
release, 668
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, 669
Avl_map, 671
insert, 672
operator[], 673
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >, 674
cxx::Avl_tree< Node, Get_key, Compare >, 677
insert, 681
Iterator, 681
remove, 681
cxx::Avl_tree_node, 682
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 685
alloc, 687
free, 688
free_objects, 689
max_free_slabs, 687
object_size, 687
objects_per_slab, 687
slab_size, 687
total_objects, 690
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i, 691
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 692
alloc, 695
free, 695
free_objects, 696
max_free_slabs, 695
object_size, 695
objects_per_slab, 695
slab_size, 695
total_objects, 697
cxx::Bitfield< T, LSB, MSB >, 698
Bits, 700
Bits_type, 699
get, 700
get_unshifted, 701
Low_mask, 700
Lsb, 700
Mask, 700
Masks, 700
Msb, 700
set, 702
set_dirty, 702
set_unshifted, 703
set_unshifted_dirty, 704
Shift_type, 699
val, 705
val_dirty, 707
val_unshifted, 708
cxx::Bitfield< T, LSB, MSB >::Value< TT >, 709
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >, 710
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >, 712
cxx::Bitmap< BITS >, 713
scan_zero, 716
cxx::Bitmap_base, 717
bit, 720
bit_index, 721
C_bits, 719
clear_bit, 721
operator[], 722
scan_zero, 723
set_bit, 724
W_bits, 719
word_index, 724
cxx::Bitmap_base::Bit, 725
cxx::Bitmap_base::Char< BITS >, 726
cxx::Bitmap_base::Word< BITS >, 726
cxx::Bits, 609
cxx::Bits::Avl_map_get_key< KEY_TYPE >, 728
cxx::Bits::Avl_set_get_key< KEY_TYPE >, 728
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 729
Base_avl_set, 733
begin, 734, 735
E_exist, 733
E_inval, 733
E_noent, 733
E_nomem, 733
end, 736
erase, 737
find_node, 738
insert, 738
lower_bound_node, 739
rbegin, 740
remove, 741
rend, 742
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, 743
operator*, 744
operator->, 744
valid, 745
cxx::Bits::Basic_list< POLICY >, 745
clear, 748
iter, 748
cxx::Bits::Bst< Node, Get_key, Compare >, 749
begin, 752, 753
dir, 754
end, 755
find, 756
find_node, 757
lower_bound_node, 758
rbegin, 759
remove_all, 760
remove_tree, 761
rend, 762

cxx::Bits::Bst_node, 763
cxx::Bits::Direction, 765
Direction_e, 767
L, 767
N, 767
operator!, 767
R, 767
cxx::Bits::Smart_ptr_list< ITEM >, 768
pop_front, 770
cxx::Bits::Smart_ptr_list_item< T, STORE_T >, 770
cxx::H_list< T, POLICY >, 772
erase, 774
insert, 775
insert_after, 776
insert_before, 777
iter, 777
pop_front, 778
remove, 779
replace, 780
cxx::H_list_item_t< ELEM_TYPE >, 780
~H_list_item_t, 782
H_list_item_t, 782
cxx::H_list_t< T >, 783
cxx::List< D, Alloc >, 786
operator[], 787
cxx::List< D, Alloc >::Iter, 788
cxx::List_alloc, 789
alloc, 790
alloc_max, 790
avail, 791
free, 791
List_alloc, 789
cxx::List_item, 792
push_back, 793
push_front, 794
remove, 794
cxx::List_item::Iter, 795
cxx::List_item::T_iter< T, Poly >, 797
cxx::Lt_functor< Obj >, 800
cxx::New_allocator< _Type >, 800
cxx::Nothrow, 802
cxx::Pair< First, Second >, 802
Pair, 803
cxx::Pair_first_compare< Cmp, Typ >, 804
operator(), 805
Pair_first_compare, 805
cxx::Ref_obj_list_item< T >, 806
cxx::Ref_ptr< T, CNT >, 807
get, 810
ptr, 810
Ref_ptr, 809, 810
release, 811
cxx::S_list< T, POLICY >, 811
pop_front, 814
cxx::Slab< Type, Slab_size, Max_free, Alloc >, 815
alloc, 817
free, 817
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >, 818
alloc, 822
cxx::static_vector< T, IDX >, 823
cxx::String, 824
find, 827
from_dec, 828
from_hex, 829
starts_with, 830
String, 827
cxx::Weak_ref< T >, 831
cxx::Weak_ref_base, 833

d_tag
Elf32_Dyn, 837
Elf64_Dyn, 848
data
L4::Ipc::Varg, 1016
Data_buffer
L4virtio::Svr::Data_buffer, 1669
data_size
L4virtio::Svr::Block_request< Ds_data >, 1666
data_space
L4Re::Vfs::Be_file, 1456
L4Re::Vfs::Regular_file, 1485
Dataspace interface, 176
l4re_ds_allocate, 178
l4re_ds_clear, 178
l4re_ds_copy_in, 178
L4RE_DS_F_BUFFERABLE, 177
L4RE_DS_F_CACHEABLE, 177
L4RE_DS_F_CACHING_MASK, 177
L4RE_DS_F_CACHING_SHIFT, 177
L4RE_DS_F_NORMAL, 177
L4RE_DS_F_UNCACHEABLE, 177
l4re_ds_flags, 178
l4re_ds_info, 179
l4re_ds_map_flags, 177
l4re_ds_size, 179
debug
L4Re::Debug_obj, 1314
Debug interface, 179
l4re_debug_obj_debug, 180
debugger.h
__strcpy_maxlen, 2175
l4_debugger_get_object_name, 2176
l4_debugger_query_log_name, 2176
l4_debugger_query_log_typeid, 2177
l4_debugger_switch_log, 2177
Debugging API, 180
dec_refcnt
L4::Kobject, 1096
Dedicated
L4::Kip::Mem_desc, 1087
delete_buffer
L4Re::Video::Goos, 1500
delete_obj
L4::Task, 1177
delete_view

L4Re::Video::Goos, 1500
Demand
L4::Kobject_typeid< T >, 1108
L4::Type_info::Demand, 1203
demand
L4::Kobject_typeid< T >, 1108
desc
L4virtio::Driver::Virtqueue, 1645
L4virtio::Svr::Virtqueue, 1721
L4virtio::Svr::Virtqueue::Head_desc, 1724
desc_align
L4virtio::Virtqueue, 1730
desc_avail
L4virtio::Svr::Virtqueue, 1721
desc_size
L4virtio::Virtqueue, 1730
detach
L4::Irq, 1068
L4Re::Rm, 1377–1379
L4Re::Util::Event_buffer_t< PAYLOAD >, 1420
Detach_again
L4Re::Rm, 1375
Detach_exact
L4Re::Rm, 1375
Detach_flags
L4Re::Rm, 1374
Detach_free
L4Re::Rm::F, 1385
Detach_keep
L4Re::Rm, 1375
Detach_overlap
L4Re::Rm, 1375
Detach_result
L4Re::Rm, 1375
Detached_ds
L4Re::Rm, 1375
Dev_config
L4virtio::Svr::Dev_config, 1673, 1674
dev_handle
L4vbus::Device, 1552
device
L4vbus::Device, 1552
device_by_hid
L4vbus::Device, 1553
device_config
L4virtio::Device, 1616
device_error
L4virtio::Svr::Device_t< DATA >, 1689
device_notification_irq
L4virtio::Device, 1616
device_notify_irq
L4virtio::Svr::Device_t< DATA >, 1690
DF_1_CONFALT
ELF binary format, 195
DF_1_DIRECT
ELF binary format, 195
DF_1_DISPRELDNE
ELF binary format, 195
DF_1_DISPRELPND
ELF binary format, 195
DF_1_ENDFILTEE
ELF binary format, 195
DF_1_GLOBAL
ELF binary format, 195
DF_1_GROUP
ELF binary format, 195
DF_1_INITFIRST
ELF binary format, 195
DF_1_INTERPOSE
ELF binary format, 195
DF_1_LOADFLTR
ELF binary format, 195
DF_1_NODEFLIB
ELF binary format, 195
DF_1_NODELETE
ELF binary format, 195
DF_1_NODUMP
ELF binary format, 195
DF_1_NOOPEN
ELF binary format, 195
DF_1_NOW
ELF binary format, 194
DF_1_ORIGIN
ELF binary format, 195
DF_BIND_NOW
ELF binary format, 196
DF_ORIGIN
ELF binary format, 196
DF_P1_GROUPPERM
ELF binary format, 195
DF_P1_LAZYLOAD
ELF binary format, 195
DF_STATIC_TLS
ELF binary format, 196
DF_SYMBOLIC
ELF binary format, 196
DF_TEXTREL
ELF binary format, 196
di
l4_vcpu_regs_t, 1283
dir
cxx::Bits::Bst< Node, Get_key, Compare >, 754
Direction
L4Re::Dma_space, 1316
Direction_e
cxx::Bits::Direction, 767
disable
L4virtio::Virtqueue, 1731
disable_notify
L4virtio::Svr::Virtqueue, 1722
disassociate
L4Re::Dma_space, 1317
dispatch
L4::Basic_registry, 864
L4::Epiface, 898
L4::Epiface_t< Derived, IFACE, BASE, bool >, 903

L4::Irqep_t< Derived, BASE, bool >, 1083
L4::Server_object, 1159
dispatch_meta_request
L4::Server_object_t< IFACE, BASE >, 1163
DMA space, 176
DMA Space Interface, 172
l4re_dma_space_associate, 173
l4re_dma_space_disassociate, 174
l4re_dma_space_map, 174
l4re_dma_space_t, 173
l4re_dma_space_unmap, 175
dma_space.h
L4RE_DMA_SPACE_BIDIRECTIONAL, 1962
L4RE_DMA_SPACE_COHERENT, 1963
l4re_dma_space_direction, 1962
L4RE_DMA_SPACE_FROM_DEVICE, 1962
L4RE_DMA_SPACE_NONE, 1962
L4RE_DMA_SPACE_PHYS_SPACE, 1963
l4re_dma_space_space_attribs, 1962
L4RE_DMA_SPACE_TO_DEVICE, 1962
done
L4virtio::Svr::Data_buffer, 1670
down
L4::Semaphore, 1150
drive_cylinders
l4util_mb_drive_t, 1541
drive_mode
l4util_mb_drive_t, 1541
drive_number
l4util_mb_drive_t, 1541
driver_acknowledge
L4virtio::Driver::Device, 1633
driver_connect
L4virtio::Driver::Device, 1633
Driver_mem_region_t
L4virtio::Svr::Driver_mem_region_t< DATA >, 1706
drv_base
L4virtio::Svr::Driver_mem_region_t< DATA >, 1707
ds
L4virtio::Svr::Dev_config, 1675
L4virtio::Svr::Driver_mem_region_t< DATA >, 1707
Ds_map_mask
L4Re::Rm::F, 1385
ds_offset
L4virtio::Svr::Driver_mem_region_t< DATA >, 1708
DT_BIND_NOW
ELF binary format, 196
DT_DEBUG
ELF binary format, 196
DT_ENCODING
ELF binary format, 197
DT_FINI
ELF binary format, 196
DT_FINI_ARRAY
ELF binary format, 197
DT_FINI_ARRAYSZ
ELF binary format, 197
DT_FLAGS
ELF binary format, 197
DT_HASH
ELF binary format, 196
DT_HIOS
ELF binary format, 197
DT_HIPROC
ELF binary format, 197
DT_INIT
ELF binary format, 196
DT_INIT_ARRAY
ELF binary format, 197
DT_INIT_ARRAYSZ
ELF binary format, 197
DT_JMPREL
ELF binary format, 196
DT_LOOS
ELF binary format, 197
DT_LOPROC
ELF binary format, 197
DT_NEEDED
ELF binary format, 196
DT_NULL
ELF binary format, 196
DT_NUM
ELF binary format, 197
DT_PLTGOT
ELF binary format, 196
DT_PLTRELSZ
ELF binary format, 196
DT_PREINIT_ARRAY
ELF binary format, 197
DT_PREINIT_ARRAYSZ
ELF binary format, 197
DT_PTRREL
ELF binary format, 196
DT_REL
ELF binary format, 196
DT_RELA
ELF binary format, 196
DT_RELAENT
ELF binary format, 196
DT_RELASZ
ELF binary format, 196
DT_RELENT
ELF binary format, 196
DT_RELSZ
ELF binary format, 196
DT_RPATH
ELF binary format, 196
DT_RUNPATH
ELF binary format, 197
DT_SONAME
ELF binary format, 196
DT_STRSZ

ELF binary format, 196
DT_STRTAB
ELF binary format, 196
DT_SYMBOLIC
ELF binary format, 196
DT_SYMENT
ELF binary format, 196
DT_SYMTAB
ELF binary format, 196
DT_TEXTREL
ELF binary format, 196
dump
L4Re::Video::Color_component, 1492
L4Re::Video::Pixel_info, 1508
L4virtio::Virtqueue, 1732
dx
l4_vcpu_regs_t, 1284

E_exist
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 733
e_flags
Elf32_Ehdr, 839
Elf64_Ehdr, 850
E_inval
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 733
e_machine
Elf32_Ehdr, 839
Elf64_Ehdr, 850
E_noent
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 733
E_nomem
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 733
e_type
Elf32_Ehdr, 839
Elf64_Ehdr, 850
e_version
Elf32_Ehdr, 840
Elf64_Ehdr, 851
Eager_map
L4Re::Rm::F, 1385
EDID parsing functionality, 181
Libedid_block_size, 182
libedid_check_header, 182
libedid_checksum, 182
Libedid_consts, 181
libedid_dump, 182
libedid_dump_standard_timings, 183
libedid_num_ext_blocks, 183
libedid_pnp_id, 183
libedid_prefered_resolution, 184
libedid_revision, 184
libedid_version, 184
EF_ARM_ALIGN8
ELF binary format, 197
EI_ABIVERSION
ELF binary format, 198
EI_CLASS
ELF binary format, 198
EI_DATA
ELF binary format, 198
EI_MAG0
ELF binary format, 198
EI_MAG1
ELF binary format, 198
EI_MAG2
ELF binary format, 198
EI_MAG3
ELF binary format, 198
EI_NIDENT
ELF binary format, 193
EI_OSABI
ELF binary format, 198
EI_PAD
ELF binary format, 198
EI_VERSION
ELF binary format, 198
ELF binary format, 185
AT_BASE, 193
AT_EGID, 193
AT_ENTRY, 193
AT_EUID, 193
AT_EXECFD, 193
AT_FLAGS, 193
AT_GID, 193
AT_IGNORE, 193
AT_L4_AUX, 193
AT_L4_ENV, 193
AT_NOTELF, 193
AT_NULL, 193
AT_PAGESZ, 193
AT_PHDR, 193
AT_PHENT, 193
AT_PHNUM, 193
AT_UID, 193
DF_1_CONFALT, 195
DF_1_DIRECT, 195
DF_1_DISPRELDNE, 195
DF_1_DISPRELPND, 195
DF_1_ENDFILTEE, 195
DF_1_GLOBAL, 195
DF_1_GROUP, 195
DF_1_INITFIRST, 195
DF_1_INTERPOSE, 195
DF_1_LOADFLTR, 195
DF_1_NODEFLIB, 195
DF_1_NODELETE, 195
DF_1_NODUMP, 195
DF_1_NOOPEN, 195
DF_1_NOW, 194
DF_1_ORIGIN, 195
DF_BIND_NOW, 196
DF_ORIGIN, 196
DF_P1_GROUPPERM, 195

DF_P1_LAZYLOAD, 195
DF_STATIC_TLS, 196
DF_SYMBOLIC, 196
DF_TEXTREL, 196
DT_BIND_NOW, 196
DT_DEBUG, 196
DT_ENCODING, 197
DT_FINI, 196
DT_FINI_ARRAY, 197
DT_FINI_ARRAYSZ, 197
DT_FLAGS, 197
DT_HASH, 196
DT_HIOS, 197
DT_HIPROC, 197
DT_INIT, 196
DT_INIT_ARRAY, 197
DT_INIT_ARRAYSZ, 197
DT_JMPREL, 196
DT_LOOS, 197
DT_LOPROC, 197
DT_NEEDED, 196
DT_NULL, 196
DT_NUM, 197
DT_PLTGOT, 196
DT_PLTRELSZ, 196
DT_PREINIT_ARRAY, 197
DT_PREINIT_ARRAYSZ, 197
DT_PTRREL, 196
DT_REL, 196
DT_RELA, 196
DT_RELAENT, 196
DT_RELASZ, 196
DT_RELENT, 196
DT_RELSZ, 196
DT_RPATH, 196
DT_RUNPATH, 197
DT_SONAME, 196
DT_STRSZ, 196
DT_STRTAB, 196
DT_SYMBOLIC, 196
DT_SYMENT, 196
DT_SYMTAB, 196
DT_TEXTREL, 196
EF_ARM_ALIGN8, 197
EI_ABIVERSION, 198
EI_CLASS, 198
EI_DATA, 198
EI_MAG0, 198
EI_MAG1, 198
EI_MAG2, 198
EI_MAG3, 198
EI_NIDENT, 193
EI_OSABI, 198
EI_PAD, 198
EI_VERSION, 198
ELF32_R_TYPE, 191
ELF32_ST_BIND, 191
ELF32_ST_TYPE, 191
ELF64_R_TYPE, 191
ELF64_ST_BIND, 192
ELF64_ST_TYPE, 192
Elf_ARM_SBs, 193
Elf_ATs, 193
Elf_CIASSs, 194
Elf_DATAs, 194
Elf_DF_1s, 194
Elf_DF_P1s, 195
Elf_DFs, 195
Elf_DTs, 196
Elf_EF_ARM_s, 197
Elf_EIs, 197
Elf_EMs, 198
Elf_ETs, 200
Elf_EVs, 200
Elf_MAGs, 200
Elf_NTs_core, 201
Elf_NTs_obj, 201
Elf_OSABIs, 202
ELF_PFs, 202
Elf_PTs, 203
Elf_R_386_s, 203
Elf_R_AARCH64_s, 205
Elf_R_ARM_s, 206
Elf_R_X86_64_s, 206
Elf_SHF_s_ARM, 207
Elf_SHFs, 207
Elf_SHNs, 208
Elf_SHTs, 208
Elf_STBs, 209
Elf_STTs, 210
ELFCLASS32, 194
ELFCLASS64, 194
ELFCLASSNONE, 194
ELFCLASSNUM, 194
ELFDATA2LSB, 194
ELFDATA2MSB, 194
ELFDATANONE, 194
ELFDATANUM, 194
ELFMAG0, 201
ELFMAG1, 201
ELFMAG2, 201
ELFMAG3, 201
ELFOSABI_AIX, 202
ELFOSABI_ARM, 202
ELFOSABI_FREEBSD, 202
ELFOSABI_HPUX, 202
ELFOSABI_IRIX, 202
ELFOSABI_LINUX, 202
ELFOSABI_MODESTO, 202
ELFOSABI_NETBSD, 202
ELFOSABI_NONE, 202
ELFOSABI_OPENBSD, 202
ELFOSABI_SOLARIS, 202
ELFOSABI_STANDALONE, 202
ELFOSABI_SYSV, 202
ELFOSABI_TRU64, 202

EM_386, 198
EM_68HC05, 199
EM_68HC08, 199
EM_68HC11, 199
EM_68HC12, 199
EM_68HC16, 199
EM_68K, 198
EM_860, 198
EM_88K, 198
EM_960, 198
EM_AARCH64, 199
EM_ALPHA, 198
EM_ALTERA_NIOS2, 199
EM_ARC, 199
EM_ARC_A5, 199
EM_ARM, 198
EM_AVR, 199
EM_COLDFIRE, 199
EM_CRIS, 199
EM_D10V, 199
EM_D30V, 199
EM_FIREPATH, 199
EM_FR20, 198
EM_FR30, 199
EM_FX66, 199
EM_H8_300, 199
EM_H8_300H, 199
EM_H8_500, 199
EM_H8S, 199
EM_HUANY, 199
EM_IA_64, 199
EM_JAVELIN, 199
EM_M32, 198
EM_M32R, 199
EM_MICROBLAZE, 199
EM_MIPS, 198
EM_MIPS_RS4_BE, 198
EM_MIPS_X, 199
EM_MMIX, 199
EM_MN10200, 199
EM_MN10300, 199
EM_NONE, 198
EM_OPENRISC, 199
EM_PARISC, 198
EM_PDSP, 199
EM_PJ, 199
EM_PPC, 198
EM_PRISM, 199
EM_RCE, 198
EM_RH32, 198
EM_SH, 199
EM_SPARC, 198
EM_SPARC32PLUS, 198
EM_SPARC64, 198
EM_SPARCV9, 199
EM_ST19, 199
EM_ST7, 199
EM_ST9PLUS, 199
EM_SVX, 199
EM_TILEGX, 199
EM_TILEPRO, 199
EM_TRICORE, 199
EM_V800, 198
EM_V850, 199
EM_VAX, 199
EM_VPP500, 198
EM_X86_64, 199
EM_XTENSA, 199
EM_ZSP, 199
ET_CORE, 200
ET_DYN, 200
ET_EXEC, 200
ET_HIPROC, 200
ET_LOPROC, 200
ET_NONE, 200
ET_REL, 200
EV_CURRENT, 200
EV_NONE, 200
NT_ASRS, 201
NT_AUXV, 201
NT_FPREGSET, 201
NT_GWINDOWS, 201
NT_LWPSINFO, 201
NT_LWPSTATUS, 201
NT_PLATFORM, 201
NT_PRCRED, 201
NT_PRFPXREG, 201
NT_PRPSINFO, 201
NT_PRSTATUS, 201
NT_PRXREG, 201
NT_PSINFO, 201
NT_PSTATUS, 201
NT_TASKSTRUCT, 201
NT_UTSNAME, 201
NT_VERSION, 202
PF_ARM_SB, 193
PF_MASKOS, 203
PF_MASKPROC, 203
PF_R, 203
PF_W, 203
PF_X, 203
PT_DYNAMIC, 203
PT_GNU_EH_FRAME, 203
PT_GNU_RELRO, 203
PT_GNU_STACK, 203
PT_HIOS, 203
PT_HIPROC, 203
PT_INTERP, 203
PT_L4_AUX, 203
PT_L4_KIP, 203
PT_L4_STACK, 203
PT_LOAD, 203
PT_LOOS, 203
PT_LOPROC, 203
PT_NOTE, 203
PT_NULL, 203

PT_NUM, 203
PT_PHDR, 203
PT_SHLIB, 203
PT_TLS, 203
R_386_32, 205
R_386_COPY, 205
R_386_GLOB_DAT, 205
R_386_GOT32, 205
R_386_GOTOFF, 205
R_386_GOTPC, 205
R_386_JMP_SLOT, 205
R_386_NONE, 205
R_386_NUM, 205
R_386_PC32, 205
R_386_PLT32, 205
R_386_RELATIVE, 205
R_386_TLS_DTPMOD32, 205
R_386_TLS_DTPOFF32, 205
R_386_TLS_GD, 205
R_386_TLS_GD_32, 205
R_386_TLS_GD_CALL, 205
R_386_TLS_GD_POP, 205
R_386_TLS_GD_PUSH, 205
R_386_TLS_GOTIE, 205
R_386_TLS_IE, 205
R_386_TLS_IE_32, 205
R_386_TLS_LDM, 205
R_386_TLS_LDM_32, 205
R_386_TLS_LDM_CALL, 205
R_386_TLS_LDM_POP, 205
R_386_TLS_LDM_PUSH, 205
R_386_TLS_LDO_32, 205
R_386_TLS_LE, 205
R_386_TLS_LE_32, 205
R_386_TLS_TPOFF, 205
R_386_TLS_TPOFF32, 205
R_AARCH64_NONE, 205
R_ARM_ABS12, 206
R_ARM_ABS16, 206
R_ARM_ABS32, 206
R_ARM_ABS8, 206
R_ARM_COPY, 206
R_ARM_GLOB_DAT, 206
R_ARM_GOT32, 206
R_ARM_GOTOFF, 206
R_ARM_GOTPC, 206
R_ARM_JUMP_SLOT, 206
R_ARM_NONE, 206
R_ARM_NUM, 206
R_ARM_PC24, 206
R_ARM_PLT32, 206
R_ARM_REL32, 206
R_ARM_RELATIVE, 206
R_ARM_THM_PC11, 206
R_ARM_THM_PC9, 206
R_X86_64_16, 207
R_X86_64_32, 207
R_X86_64_32S, 207
R_X86_64_64, 206
R_X86_64_8, 207
R_X86_64_COPY, 207
R_X86_64_DTPMOD64, 207
R_X86_64_DTPOFF32, 207
R_X86_64_DTPOFF64, 207
R_X86_64_GLOB_DAT, 207
R_X86_64_GOT32, 206
R_X86_64_GOTPCREL, 207
R_X86_64_GOTTPOFF, 207
R_X86_64_JUMP_SLOT, 207
R_X86_64_NONE, 206
R_X86_64_PC16, 207
R_X86_64_PC32, 206
R_X86_64_PC8, 207
R_X86_64_PLT32, 206
R_X86_64_RELATIVE, 207
R_X86_64_TLSGD, 207
R_X86_64_TLSLD, 207
R_X86_64_TPOFF32, 207
R_X86_64_TPOFF64, 207
SHF_ALLOC, 208
SHF_ARM_COMDEF, 207
SHF_ARM_ENTRYSECT, 207
SHF_EXECINSTR, 208
SHF_GROUP, 208
SHF_INFO_LINK, 208
SHF_LINK_ORDER, 208
SHF_MASKOS, 208
SHF_MASKPROC, 208
SHF_MERGE, 208
SHF_OS_NONCONFORMING, 208
SHF_STRINGS, 208
SHF_TLS, 208
SHF_WRITE, 208
SHN_ABS, 208
SHN_COMMON, 208
SHN_HIPROC, 208
SHN_HIRESERVE, 208
SHN_LOPROC, 208
SHN_LORESERVE, 208
SHN_UNDEF, 208
SHT_DYNAMIC, 209
SHT_DYNSYM, 209
SHT_FINI_ARRAY, 209
SHT_GROUP, 209
SHT_HASH, 209
SHT_HIOS, 209
SHT_HIPROC, 209
SHT_HIUSER, 209
SHT_INIT_ARRAY, 209
SHT_LOOS, 209
SHT_LOPROC, 209
SHT_LOUSER, 209
SHT_NOBITS, 209
SHT_NOTE, 209
SHT_NULL, 209
SHT_NUM, 209

SHT_PREINIT_ARRAY, 209
SHT_PROGBITS, 209
SHT_REL, 209
SHT_RELA, 209
SHT_SHLIB, 209
SHT_STRTAB, 209
SHT_SYMTAB, 209
SHT_SYMTAB_SHNDX, 209
STB_GLOBAL, 209
STB_HIOS, 209
STB_HIPROC, 209
STB_LOCAL, 209
STB_LOOS, 209
STB_LOPROC, 209
STB_WEAK, 209
STT_FILE, 210
STT_FUNC, 210
STT_HIOS, 210
STT_HIPROC, 210
STT_LOOS, 210
STT_LOPROC, 210
STT_NOTYPE, 210
STT_OBJECT, 210
STT_SECTION, 210
Elf32_Auxv, 836
atype, 836
Elf32_Dyn, 837
d_tag, 837
Elf32_Ehdr, 838
e_flags, 839
e_machine, 839
e_type, 839
e_version, 840
Elf32_Phdr, 840
p_flags, 841
p_type, 841
ELF32_R_TYPE
ELF binary format, 191
Elf32_Rel, 842
Elf32_Rela, 843
Elf32_Shdr, 843
sh_flags, 845
sh_type, 845
ELF32_ST_BIND
ELF binary format, 191
ELF32_ST_TYPE
ELF binary format, 191
Elf32_Sym, 845
Elf64_Auxv, 846
atype, 847
Elf64_Dyn, 847
d_tag, 848
Elf64_Ehdr, 848
e_flags, 850
e_machine, 850
e_type, 850
e_version, 851
Elf64_Phdr, 851
p_flags, 852
p_type, 852
ELF64_R_TYPE
ELF binary format, 191
Elf64_Rel, 853
Elf64_Rela, 854
Elf64_Shdr, 854
sh_flags, 856
sh_type, 856
ELF64_ST_BIND
ELF binary format, 192
ELF64_ST_TYPE
ELF binary format, 192
Elf64_Sym, 856
Elf_ARM_SBs
ELF binary format, 193
Elf_ATs
ELF binary format, 193
Elf_CIASSs
ELF binary format, 194
Elf_DATAs
ELF binary format, 194
Elf_DF_1s
ELF binary format, 194
Elf_DF_P1s
ELF binary format, 195
Elf_DFs
ELF binary format, 195
Elf_DTs
ELF binary format, 196
Elf_EF_ARM_s
ELF binary format, 197
Elf_EIs
ELF binary format, 197
Elf_EMs
ELF binary format, 198
Elf_ETs
ELF binary format, 200
Elf_EVs
ELF binary format, 200
Elf_MAGs
ELF binary format, 200
Elf_NTs_core
ELF binary format, 201
Elf_NTs_obj
ELF binary format, 201
Elf_OSABIs
ELF binary format, 202
ELF_PFs
ELF binary format, 202
Elf_PTs
ELF binary format, 203
Elf_R_386_s
ELF binary format, 203
Elf_R_AARCH64_s
ELF binary format, 205
Elf_R_ARM_s
ELF binary format, 206

Elf_R_X86_64_s
ELF binary format, 206
Elf_SHF_s_ARM
ELF binary format, 207
Elf_SHFs
ELF binary format, 207
Elf_SHNs
ELF binary format, 208
Elf_SHTs
ELF binary format, 208
Elf_STBs
ELF binary format, 209
Elf_STTs
ELF binary format, 210
ELFCLASS32
ELF binary format, 194
ELFCLASS64
ELF binary format, 194
ELFCLASSNONE
ELF binary format, 194
ELFCLASSNUM
ELF binary format, 194
ELFDATA2LSB
ELF binary format, 194
ELFDATA2MSB
ELF binary format, 194
ELFDATANONE
ELF binary format, 194
ELFDATANUM
ELF binary format, 194
ELFMAG0
ELF binary format, 201
ELFMAG1
ELF binary format, 201
ELFMAG2
ELF binary format, 201
ELFMAG3
ELF binary format, 201
ELFOSABI_AIX
ELF binary format, 202
ELFOSABI_ARM
ELF binary format, 202
ELFOSABI_FREEBSD
ELF binary format, 202
ELFOSABI_HPUX
ELF binary format, 202
ELFOSABI_IRIX
ELF binary format, 202
ELFOSABI_LINUX
ELF binary format, 202
ELFOSABI_MODESTO
ELF binary format, 202
ELFOSABI_NETBSD
ELF binary format, 202
ELFOSABI_NONE
ELF binary format, 202
ELFOSABI_OPENBSD
ELF binary format, 202
ELFOSABI_SOLARIS
ELF binary format, 202
ELFOSABI_STANDALONE
ELF binary format, 202
ELFOSABI_SYSV
ELF binary format, 202
ELFOSABI_TRU64
ELF binary format, 202
EM_386
ELF binary format, 198
EM_68HC05
ELF binary format, 199
EM_68HC08
ELF binary format, 199
EM_68HC11
ELF binary format, 199
EM_68HC12
ELF binary format, 199
EM_68HC16
ELF binary format, 199
EM_68K
ELF binary format, 198
EM_860
ELF binary format, 198
EM_88K
ELF binary format, 198
EM_960
ELF binary format, 198
EM_AARCH64
ELF binary format, 199
EM_ALPHA
ELF binary format, 198
EM_ALTERA_NIOS2
ELF binary format, 199
EM_ARC
ELF binary format, 199
EM_ARC_A5
ELF binary format, 199
EM_ARM
ELF binary format, 198
EM_AVR
ELF binary format, 199
EM_COLDFIRE
ELF binary format, 199
EM_CRIS
ELF binary format, 199
EM_D10V
ELF binary format, 199
EM_D30V
ELF binary format, 199
EM_FIREPATH
ELF binary format, 199
EM_FR20
ELF binary format, 198
EM_FR30
ELF binary format, 199
EM_FX66
ELF binary format, 199

EM_H8_300
ELF binary format, 199
EM_H8_300H
ELF binary format, 199
EM_H8_500
ELF binary format, 199
EM_H8S
ELF binary format, 199
EM_HUANY
ELF binary format, 199
EM_IA_64
ELF binary format, 199
EM_JAVELIN
ELF binary format, 199
EM_M32
ELF binary format, 198
EM_M32R
ELF binary format, 199
EM_MICROBLAZE
ELF binary format, 199
EM_MIPS
ELF binary format, 198
EM_MIPS_RS4_BE
ELF binary format, 198
EM_MIPS_X
ELF binary format, 199
EM_MMIX
ELF binary format, 199
EM_MN10200
ELF binary format, 199
EM_MN10300
ELF binary format, 199
EM_NONE
ELF binary format, 198
EM_OPENRISC
ELF binary format, 199
EM_PARISC
ELF binary format, 198
EM_PDSP
ELF binary format, 199
EM_PJ
ELF binary format, 199
EM_PPC
ELF binary format, 198
EM_PRISM
ELF binary format, 199
EM_RCE
ELF binary format, 198
EM_RH32
ELF binary format, 198
EM_SH
ELF binary format, 199
EM_SPARC
ELF binary format, 198
EM_SPARC32PLUS
ELF binary format, 198
EM_SPARC64
ELF binary format, 198
EM_SPARCV9
ELF binary format, 199
EM_ST19
ELF binary format, 199
EM_ST7
ELF binary format, 199
EM_ST9PLUS
ELF binary format, 199
EM_SVX
ELF binary format, 199
EM_TILEGX
ELF binary format, 199
EM_TILEPRO
ELF binary format, 199
EM_TRICORE
ELF binary format, 199
EM_V800
ELF binary format, 198
EM_V850
ELF binary format, 199
EM_VAX
ELF binary format, 199
EM_VPP500
ELF binary format, 198
EM_X86_64
ELF binary format, 199
EM_XTENSA
ELF binary format, 199
EM_ZSP
ELF binary format, 199
empty
L4virtio::Svr::Driver_mem_region_t< DATA >, 1708
enable_notify
L4virtio::Svr::Virtqueue, 1722
end
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 736
cxx::Bits::Bst< Node, Get_key, Compare >, 755
L4::Kip::Mem_desc, 1090
enqueue_descriptor
L4virtio::Driver::Virtqueue, 1646
entry_ip
L4vcpu::Vcpu, 1604
entry_sp
L4vcpu::Vcpu, 1605
env
L4Re::Env, 1322
env.h
l4re_env_t, 2012
erase
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 737
cxx::H_list< T, POLICY >, 774
err_no
L4::Runtime_error, 1141
Error
L4virtio::Svr::Bad_descriptor, 1655

Error codes, 216
L4_E2BIG, 217
L4_EACCESS, 217
L4_EADDRNOTAVAIL, 217
L4_EAGAIN, 217
L4_EBADPROTO, 217
L4_EBUSY, 217
L4_EEXIST, 217
L4_EFAULT, 217
L4_EINVAL, 217
L4_EIO, 217
L4_EIPC_HI, 217
L4_EIPC_LO, 217
L4_EMSGMISSARG, 217
L4_EMSGTOOLONG, 217
L4_EMSGTOOSHORT, 217
L4_ENAMETOOLONG, 217
L4_ENODEV, 217
L4_ENOENT, 217
L4_ENOMEM, 217
L4_ENOREPLY, 217
L4_ENOSPC, 217
L4_ENOSYS, 217
L4_ENXIO, 217
L4_EOK, 217
L4_EPERM, 217
L4_ERANGE, 217
L4_ERRNOMAX, 217
l4_error_code_t, 216
Error Handling, 210
l4_error, 212
L4_IPC_ENOT_EXISTENT, 211
l4_ipc_error, 213
l4_ipc_error_code, 214
L4_IPC_ERROR_MASK, 211
l4_ipc_is_rcv_error, 215
l4_ipc_is_snd_error, 215
L4_IPC_REABORTED, 212
L4_IPC_RECANCELED, 212
L4_IPC_REMAPFAILED, 212
L4_IPC_REMSGCUT, 212
L4_IPC_RERCVPFTO, 212
L4_IPC_RESNDPFTO, 212
L4_IPC_RETIMEOUT, 211
L4_IPC_SEABORTED, 212
L4_IPC_SECANCELED, 212
L4_IPC_SEMAPFAILED, 212
L4_IPC_SEMSGCUT, 212
L4_IPC_SERCVPFTO, 212
L4_IPC_SESNDPFTO, 212
L4_IPC_SETIMEOUT, 212
L4_IPC_SND_ERR_MASK, 211
l4_ipc_tcr_error_t, 211
ET_CORE
ELF binary format, 200
ET_DYN
ELF binary format, 200
ET_EXEC
ELF binary format, 200
ET_HIPROC
ELF binary format, 200
ET_LOPROC
ELF binary format, 200
ET_NONE
ELF binary format, 200
ET_REL
ELF binary format, 200
EV_CURRENT
ELF binary format, 200
EV_NONE
ELF binary format, 200
Event API, 217
Event interface, 218
l4re_event_get_axis_info, 219
l4re_event_get_buffer, 219
l4re_event_get_num_streams, 220
l4re_event_get_stream_info, 220
l4re_event_get_stream_info_for_id, 221
Event_buffer_t
L4Re::Event_buffer_t< PAYLOAD >, 1337
ex_regs
L4::Thread, 1184, 1185
exc_handler
L4::Thread::Attr, 1194
Exception registers, 221
l4_utcb_exc, 222
l4_utcb_exc_is_ex_regs_exception, 222
l4_utcb_exc_is_pf, 223
l4_utcb_exc_pc, 223
l4_utcb_exc_pc_set, 224
execute
L4Re::Ned::Cmd_control, 1363
expired
L4::Ipc_svr::Timeout, 1053
ext_alloc
L4vcpu::Vcpu, 1605
Extended vCPU support, 224
l4vcpu_ext_alloc, 225
extra_str
L4::Runtime_error, 1141

F_above
L4Re::Video::View, 1514
F_auto_refresh
L4Re::Video::Goos, 1499
F_dyn_allocated
L4Re::Video::View, 1514
F_dynamic_buffers
L4Re::Video::Goos, 1499
F_dynamic_views
L4Re::Video::Goos, 1499
F_flags_mask
L4Re::Video::View, 1514
F_fully_dynamic
L4Re::Video::View, 1514
F_l4re_video_goos_auto_refresh
Video API, 567

F_l4re_video_goos_dynamic_buffers
Video API, 567
F_l4re_video_goos_dynamic_views
Video API, 567
F_l4re_video_goos_pointer
Video API, 567
F_l4re_video_view_above
Video API, 567
F_l4re_video_view_dyn_allocated
Video API, 567
F_l4re_video_view_flags_mask
Video API, 567
F_l4re_video_view_none
Video API, 567
F_l4re_video_view_set_background
Video API, 567
F_l4re_video_view_set_buffer
Video API, 567
F_l4re_video_view_set_buffer_offset
Video API, 567
F_l4re_video_view_set_bytes_per_line
Video API, 567
F_l4re_video_view_set_flags
Video API, 567
F_l4re_video_view_set_pixel
Video API, 567
F_l4re_video_view_set_position
Video API, 567
F_none
L4Re::Video::View, 1513
F_pointer
L4Re::Video::Goos, 1499
F_set_background
L4Re::Video::View, 1514
F_set_buffer
L4Re::Video::View, 1513
F_set_buffer_offset
L4Re::Video::View, 1513
F_set_bytes_per_line
L4Re::Video::View, 1513
F_set_flags
L4Re::Video::View, 1514
F_set_pixel
L4Re::Video::View, 1514
F_set_position
L4Re::Video::View, 1514
faccessat
L4Re::Vfs::Directory, 1462
Factory, 225
l4_factory_create, 227
l4_factory_create_factory, 227
l4_factory_create_gate, 228
l4_factory_create_irq, 228
l4_factory_create_task, 229
l4_factory_create_thread, 230
l4_factory_create_vm, 230
factory
L4Re::Env, 1322, 1323
fchmod
L4Re::Vfs::Generic_file, 1476
fdatasync
L4Re::Vfs::Regular_file, 1485
features
l4_icu_info_t, 1263
Fiasco extensions, 231
fiasco_gdt_get_entry_offset, 232
fiasco_gdt_set, 232
fiasco_ldt_set, 233
fiasco_tbuf_log, 234
fiasco_tbuf_log_3val, 234
fiasco_tbuf_log_binary, 235
Fiasco real time scheduling extensions, 236
Fiasco-UX Virtual devices, 236
L4_TYPE_VHW_FRAMEBUFFER, 237
L4_TYPE_VHW_INPUT, 237
L4_TYPE_VHW_NET, 237
L4_TYPE_VHW_NONE, 237
l4_vhw_entry_type, 237
fiasco_amd64_segment_info
segment.h, 1811
fiasco_amd64_set_fs
segment.h, 1807, 1811
fiasco_amd64_set_segment_base
segment.h, 1807, 1812
fiasco_gdt_get_entry_offset
Fiasco extensions, 232
fiasco_gdt_set
Fiasco extensions, 232
fiasco_ldt_set
Fiasco extensions, 233
fiasco_tbuf_log
Fiasco extensions, 234
fiasco_tbuf_log_3val
Fiasco extensions, 234
fiasco_tbuf_log_binary
Fiasco extensions, 235
finalize_request
L4virtio::Svr::Block_dev_base< Ds_data >, 1660
find
cxx::Bits::Bst< Node, Get_key, Compare >, 756
cxx::String, 827
L4::Basic_registry, 864
L4Re::Rm, 1379
L4virtio::Svr::Driver_mem_list_t< DATA >, 1698
find_next_used
L4virtio::Driver::Virtqueue, 1646
find_node
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 738
cxx::Bits::Bst< Node, Get_key, Compare >, 757
first
L4::Kip::Mem_desc, 1091
first_free_cap
L4Re::Env, 1323
first_free_utcb
L4Re::Env, 1324

Flags
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, 1207
L4::Types::Flags< BITS_ENUM, UNDERLYING >, 1232
L4Re::Dataspace::F, 1310
L4Re::Video::Goos, 1498
L4Re::Video::View, 1513
flags
l4_exc_regs_t, 1260
l4_msgtag_t, 1269
L4Re::Dataspace, 1306
l4re_env_cap_entry_t, 1525
L4virtio::Svr::Driver_mem_region_t< DATA >, 1708
Flex pages, 237
L4_CAP_FPAGE_D, 241
L4_CAP_FPAGE_R, 241
L4_cap_fpage_rights, 241
L4_CAP_FPAGE_RO, 241
L4_CAP_FPAGE_RS, 241
L4_CAP_FPAGE_RSD, 242
L4_CAP_FPAGE_RW, 241
L4_CAP_FPAGE_RWD, 242
L4_CAP_FPAGE_RWS, 242
L4_CAP_FPAGE_RWSD, 242
L4_CAP_FPAGE_S, 241
L4_CAP_FPAGE_W, 241
l4_fpage, 244
L4_FPAGE_ADDR_BITS, 243
L4_FPAGE_ADDR_SHIFT, 243
l4_fpage_all, 245
L4_FPAGE_BUFFERABLE, 242
L4_FPAGE_C_IPCGATE_SVR, 244
L4_FPAGE_C_NO_REF_CNT, 244
L4_FPAGE_C_OBJ_RIGHT1, 244
L4_FPAGE_C_OBJ_RIGHT2, 244
L4_FPAGE_C_OBJ_RIGHT3, 244
L4_FPAGE_C_OBJ_RIGHTS, 244
L4_FPAGE_C_REF_CNT, 244
L4_FPAGE_CACHE_OPT, 242
l4_fpage_cacheability_opt_t, 242
L4_FPAGE_CACHEABLE, 242
l4_fpage_consts, 242
l4_fpage_contains, 245
L4_fpage_control, 243
L4_FPAGE_CONTROL_MASK, 243
L4_FPAGE_CONTROL_OFFSET_SHIFT, 243
l4_fpage_invalid, 246
L4_FPAGE_IO, 244
l4_fpage_ioport, 246
l4_fpage_max_order, 247
l4_fpage_memaddr, 248
L4_FPAGE_MEMORY, 244
L4_FPAGE_OBJ, 244
l4_fpage_obj, 248
l4_fpage_page, 249
L4_fpage_rights, 243
l4_fpage_rights, 249
L4_FPAGE_RIGHTS_BITS, 243
L4_FPAGE_RIGHTS_MASK, 243
L4_FPAGE_RIGHTS_SHIFT, 243
L4_FPAGE_RO, 243
L4_FPAGE_RW, 243
L4_FPAGE_RWX, 243
L4_FPAGE_RX, 243
l4_fpage_set_rights, 250
l4_fpage_size, 251
L4_FPAGE_SIZE_BITS, 243
L4_FPAGE_SIZE_SHIFT, 243
L4_FPAGE_SPECIAL, 244
L4_fpage_type, 244
l4_fpage_type, 251
L4_FPAGE_TYPE_BITS, 243
L4_FPAGE_TYPE_SHIFT, 243
L4_FPAGE_UNCACHEABLE, 242
L4_FPAGE_W, 243
L4_FPAGE_X, 243
l4_iofpage, 252
L4_IOPORT_MAX, 240
l4_is_fpage_writable, 252
l4_obj_fpage, 253
L4_obj_fpage_ctl, 244
L4_WHOLE_ADDRESS_SPACE, 240
L4_WHOLE_IOADDRESS_SPACE, 240
font.h
gfxbitmap_font_data, 1951
gfxbitmap_font_get, 1952
gfxbitmap_font_height, 1952
gfxbitmap_font_init, 1952
gfxbitmap_font_text, 1953
gfxbitmap_font_text_scale, 1953
gfxbitmap_font_width, 1954
foreach_available_event
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, 1416
fpage
L4::Cap_base, 878
free
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 688
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 695
cxx::List_alloc, 791
cxx::Slab< Type, Slab_size, Max_free, Alloc >, 817
L4Re::Cap_alloc, 1297
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, 1404
free_area
L4Re::Rm, 1380
free_capability
L4Re::Util::Names::Name_space, 1431
free_descriptor
L4virtio::Driver::Virtqueue, 1647
free_dynamic_entry

L4Re::Util::Names::Name_space, 1432
free_epiface
L4Re::Util::Names::Name_space, 1432
free_fd
L4Re::Vfs::Fs, 1472
free_objects
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 689
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 696
from_ci
L4::Ipc::Cap< T >, 957
from_dec
cxx::String, 828
From_device
L4Re::Dma_space, 1316
from_hex
cxx::String, 829
from_raw
L4::Types::Flags< BITS_ENUM, UNDERLYING >, 1233
fstat64
L4Re::Vfs::Be_file, 1456
L4Re::Vfs::Generic_file, 1477
fsync
L4Re::Vfs::Regular_file, 1485
ftruncate64
L4Re::Vfs::Regular_file, 1485
full
L4virtio::Svr::Driver_mem_list_t< DATA >, 1698
Functions for rendering bitmap data in frame buffers, 254
Functions for rendering bitmap fonts to frame buffers, 254
Functions to manipulate the local IDT, 255

g
L4Re::Video::Pixel_info, 1509
get
cxx::Auto_ptr< T >, 667
cxx::Bitfield< T, LSB, MSB >, 700
cxx::Ref_ptr< T, CNT >, 810
L4::Ipc::Iostream, 966, 967
L4::Ipc::Istream, 974, 975
L4Re::Env, 1324
L4Re::Video::Color_component, 1492
L4vbus::Gpio_module, 1561
L4vbus::Gpio_pin, 1569
L4virtio::Ptr< T >, 1652
get_areas
L4Re::Rm, 1381
get_attr
L4::Vcon, 1249
get_avail_idx
L4virtio::Virtqueue, 1732
get_buffer
L4Re::Event, 1335
get_buffer_demand
L4::Epiface, 899
L4::Server_object_t< IFACE, BASE >, 1163
get_cap
L4Re::Env, 1325
get_cap_alloc
L4Re::Cap_alloc, 1298
get_cmd
L4virtio::Svr::Dev_config, 1675
get_epiface
L4Re::Util::Names::Name_space, 1432
get_fb
L4Re::Util::Video::Goos_svr, 1450
get_file
L4Re::Vfs::Fs, 1472
get_infos
L4::Ipc_gate, 1027
get_lock
L4Re::Vfs::Regular_file, 1486
get_object_name
L4::Debugger, 889
get_random
L4Re::Random, 1370
get_rcv_cap
L4::Ipc_svr::Server_iface, 1048
L4Re::Util::Br_manager, 1392
get_regions
L4Re::Rm, 1381
get_resource
L4vbus::Device, 1554
get_static_buffer
L4Re::Video::Goos, 1500
get_status_flags
L4Re::Vfs::Generic_file, 1477
get_tail_avail_idx
L4virtio::Virtqueue, 1733
get_unshifted
cxx::Bitfield< T, LSB, MSB >, 701
get_value
L4::Ipc::Varg, 1016
get_writeback
L4virtio::Svr::Block_dev_base< Ds_data >, 1661
gfxbitmap_bmap
bitmap.h, 1946
gfxbitmap_color_pix_t
bitmap.h, 1945
gfxbitmap_color_t
bitmap.h, 1946
gfxbitmap_convert_color
bitmap.h, 1947
gfxbitmap_copy
bitmap.h, 1947
gfxbitmap_fill
bitmap.h, 1947
gfxbitmap_font_data
font.h, 1951
gfxbitmap_font_get
font.h, 1952
gfxbitmap_font_height
font.h, 1952
gfxbitmap_font_init

font.h, 1952
gfxbitmap_font_text
font.h, 1953
gfxbitmap_font_text_scale
font.h, 1953
gfxbitmap_font_width
font.h, 1954
gfxbitmap_offset, 857
gfxbitmap_set
bitmap.h, 1948
global_id
L4::Debugger, 890
gran_offset
l4_sched_cpu_set_t, 1272
granularity
l4_sched_cpu_set_t, 1270
guest_features
L4virtio::Svr::Dev_config, 1676

H_list_item_t
cxx::H_list_item_t< ELEM_TYPE >, 782
handle_expired_timeouts
L4::Ipc_svr::Timeout_queue, 1056
handle_mem_cmd_write
L4virtio::Svr::Device_t< DATA >, 1690
has_alpha
L4Re::Video::Pixel_info, 1510
has_more
L4virtio::Svr::Request_processor, 1712
hdr
L4virtio::Svr::Dev_config, 1677

i
L4vcpu::Vcpu, 1606
IA32 Port I/O API, 255
l4util_in16, 256
l4util_in32, 256
l4util_in8, 257
l4util_ins16, 257
l4util_ins32, 258
l4util_ins8, 258
l4util_out16, 258
l4util_out32, 259
l4util_out8, 259
l4util_outs16, 260
l4util_outs32, 260
l4util_outs8, 260
id
L4::Kobject_typeid< T >, 1109
id_received
L4::Ipc::Gen_fpage< T >, 959
idle_time
L4::Scheduler, 1144
In_area
L4Re::Rm::F, 1385
indirect_bfm_t
L4virtio::Virtqueue::Desc::Flags, 1748
Info
L4::Kip::Mem_desc, 1087
info
L4::Icu, 929
L4::Scheduler, 1145
L4Re::Dataspace, 1306
L4Re::Video::Goos, 1501
L4Re::Video::View, 1514
Info_acpi_rsdp
L4::Kip::Mem_desc, 1086
Info_sub_type
L4::Kip::Mem_desc, 1086
init
L4Re::Util::Event_t< PAYLOAD >, 1425
L4virtio::Svr::Driver_mem_list_t< DATA >, 1699
init_infos
L4Re::Util::Video::Goos_svr, 1450
init_mem_info
L4virtio::Svr::Device_t< DATA >, 1691
init_poll
L4Re::Util::Event_t< PAYLOAD >, 1426
init_queue
L4virtio::Driver::Virtqueue, 1648
Initial Environment, 284
l4re_env, 285
l4re_env_get_cap, 286
l4re_env_get_cap_e, 287
l4re_env_get_cap_l, 288
l4re_kip, 288
initial_caps
L4Re::Env, 1326
initialize_rings
L4virtio::Driver::Virtqueue, 1649
insert
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, 672
cxx::Avl_tree< Node, Get_key, Compare >, 681
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 738
cxx::H_list< T, POLICY >, 775
insert_after
cxx::H_list< T, POLICY >, 776
insert_before
cxx::H_list< T, POLICY >, 777
Integer Types, 289
interface
L4::Meta, 1115
Interface for asynchronous ISR handlers with a given IRQ capability., 291
l4irq_request_cap, 291
Interface for asynchronous ISR handlers., 292
l4irq_release, 293
l4irq_request, 293
Interface using direct functionality., 294, 296
l4irq_attach, 297
l4irq_attach_cap, 294
l4irq_attach_cap_ft, 295
l4irq_attach_ft, 297
l4irq_attach_thread, 298
l4irq_attach_thread_cap, 295

l4irq_attach_thread_cap_ft, 296
l4irq_attach_thread_ft, 298
l4irq_detach, 299
l4irq_unmask, 299
l4irq_unmask_and_wait_any, 300
l4irq_wait, 300
l4irq_wait_any, 300
Internal constants, 301
Internal functions, 303
Internal Helpers, 301
internal_loop
L4::Server< LOOP_HOOKS >, 1154
Interrupt controller, 303
l4_icu_bind, 306
l4_icu_bind_u, 307
L4_ICU_FLAG_MSI, 306
L4_icu_flags, 305
l4_icu_info, 308
l4_icu_info_t, 305
l4_icu_info_u, 308
l4_icu_mask, 309
l4_icu_mask_u, 310
l4_icu_msi_info, 311
l4_icu_msi_info_u, 312
l4_icu_set_mode, 312
l4_icu_set_mode_u, 313
l4_icu_unbind, 314
l4_icu_unbind_u, 315
l4_icu_unmask, 316
l4_icu_unmask_u, 316
Invalid
L4::Cap_base, 876
L4virtio::Ptr< T >, 1652
Invalid_capability
L4::Invalid_capability, 936
Invalid_type
L4virtio::Ptr< T >, 1652
IO interface, 261
L4IO_DEVICE_ANY, 263
L4IO_DEVICE_INVALID, 263
L4IO_DEVICE_OTHER, 263
L4IO_DEVICE_PCI, 263
l4io_device_types_t, 262
L4IO_DEVICE_USB, 263
l4io_has_resource, 264
l4io_iomem_flags_t, 263
l4io_lookup_device, 264
l4io_lookup_resource, 264
L4IO_MEM_CACHED, 263
L4IO_MEM_EAGER_MAP, 263
L4IO_MEM_NONCACHED, 263
L4IO_MEM_USE_MTRR, 263
L4IO_MEM_USE_RESERVED_AREA, 263
l4io_release_iomem, 265
l4io_release_ioport, 265
l4io_request_iomem, 266
l4io_request_iomem_region, 266
l4io_request_ioport, 267
l4io_request_resource_iomem, 268
L4IO_RESOURCE_ANY, 263
L4IO_RESOURCE_INVALID, 263
L4IO_RESOURCE_IRQ, 263
L4IO_RESOURCE_MEM, 263
L4IO_RESOURCE_PORT, 263
l4io_resource_t, 262
l4io_resource_types_t, 263
io.h
l4io_get_root_device, 1855
l4io_iterate_devices, 1855
l4io_request_all_ioports, 1856
l4io_request_icu, 1856
io_page_fault
L4::Io_pager, 939
ioctl
L4Re::Vfs::Special_file, 1490
Iostream
L4::Ipc::Iostream, 965
IPC-Gate API, 268
l4_ipc_gate_get_infos, 269
l4_rcv_ep_bind_thread, 270
ipc.h
l4_ipc_to_errno, 2203
ipc_client
L4_RPC_DEF, 2141
ipc_iface
L4_INLINE_RPC, 2144
L4_INLINE_RPC_NF, 2145
L4_INLINE_RPC_NF_OP, 2145
L4_INLINE_RPC_OP, 2146
L4_RPC, 2146
L4_RPC_NF, 2147
L4_RPC_NF_OP, 2147
L4_RPC_OP, 2148
ipc_stream
operator<<, 1900–1902
operator>>, 1903–1907
irq
L4Re::Util::Event_t< PAYLOAD >, 1427
IRQ handling library, 271
irq.h
l4util_irq_acknowledge, 1934, 1936
irq_disable_save
L4vcpu::Vcpu, 1606
irq_enable
L4vbus::Pci_dev, 1579
L4vbus::Pci_host_bridge, 1585
L4vcpu::Vcpu, 1607
irq_restore
L4vcpu::Vcpu, 1608
IRQs, 271
l4_irq_detach, 274
l4_irq_detach_u, 275
L4_IRQ_F_BOTH, 274
L4_IRQ_F_BOTH_EDGE, 274
L4_IRQ_F_CLEAR_WAKEUP, 274
L4_IRQ_F_EDGE, 274

L4_IRQ_F_LEVEL, 274
L4_IRQ_F_LEVEL_HIGH, 274
L4_IRQ_F_LEVEL_LOW, 274
L4_IRQ_F_MASK, 274
L4_IRQ_F_NEG, 274
L4_IRQ_F_NEG_EDGE, 274
L4_IRQ_F_NONE, 274
L4_IRQ_F_POS, 274
L4_IRQ_F_POS_EDGE, 274
L4_IRQ_F_SET_WAKEUP, 274
L4_irq_mode, 273
l4_irq_mux_chain, 275
l4_irq_mux_chain_u, 276
l4_irq_receive, 277
l4_irq_receive_u, 278
l4_irq_trigger, 279
l4_irq_trigger_u, 280
l4_irq_unmask, 281
l4_irq_unmask_u, 282
l4_irq_wait, 283
l4_irq_wait_u, 284
is_compatible
L4vbus::Device, 1555
is_compound
L4::Ipc::Gen_fpage< T >, 959
is_irq_entry
L4vcpu::Vcpu, 1608
is_nil
L4::Ipc::Varg, 1017
is_of
L4::Ipc::Varg, 1017
is_of_int
L4::Ipc::Varg, 1018
is_online
L4::Scheduler, 1146
is_page_fault_entry
L4vcpu::Vcpu, 1609
is_static
L4Re::Util::Dataspace_svr, 1411
is_valid
L4::Cap_base, 879
L4virtio::Ptr< T >, 1653
is_virtual
L4::Kip::Mem_desc, 1092
is_writable
L4virtio::Svr::Driver_mem_region_t< DATA >, 1708
Istream
L4::Ipc::Istream, 973
Item_bytes
L4::Ipc::Msg, 629
Item_words
L4::Ipc::Msg, 629
iter
cxx::Bits::Basic_list< POLICY >, 748
cxx::H_list< T, POLICY >, 777
Iterator
cxx::Avl_tree< Node, Get_key, Compare >, 681
Kept_ds
L4Re::Rm, 1375
Kernel Debugger, 317
l4_debugger_global_id, 318
l4_debugger_kobj_to_id, 318
l4_debugger_set_object_name, 319
Kernel Interface Page, 319
l4_kernel_info_version_offset, 321
l4_kip_clock, 322
l4_kip_clock_lw, 323
l4_kip_clock_ns, 324
L4_KIP_OFFS_READ_NS, 321
L4_KIP_OFFS_READ_US, 321
l4_kip_version, 324
l4_kip_version_string, 325
Kernel Interface Page API, 326
l4util_kip_for_each_feature, 326
l4util_kip_kernel_abi_version, 327
l4util_kip_kernel_has_feature, 327
l4util_kip_kernel_is_ux, 327
Kernel Objects, 328
Kernel-provided semaphore, 329
l4_semaphore_down, 330
l4_semaphore_up, 330
kobj_to_id
L4::Debugger, 890
kobject_typeid
L4 kernel object type information, 361
Kumem allocator utility, 331
Kumem utiltities, 332
kumem_alloc, 332
kumem_alloc
Kumem utiltities, 332
kumem_alloc.h
l4re_util_kumem_alloc, 1980

L
cxx::Bits::Direction, 767
L4, 610
cap_cast, 614
cap_dynamic_cast, 615
cap_reinterpret_cast, 616, 617
PROTO_ANY, 614
PROTO_EMPTY, 614
round_order, 617
throw_ipc_exception, 618, 619
trunc_order, 619
L4 IPC Opcodes, 333
L4_ICU_OP_BIND, 334
L4_ICU_OP_INFO, 334
L4_ICU_OP_MASK, 334
L4_ICU_OP_MSI_INFO, 334
L4_ICU_OP_SET_MODE, 334
L4_ICU_OP_UNBIND, 334
L4_ICU_OP_UNMASK, 334
L4_icu_opcode, 333
L4_IPC_GATE_BIND_OP, 335
L4_IPC_GATE_GET_INFO_OP, 335
L4_ipc_gate_ops, 334

L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP, 335
L4_PLATFORM_CTL_CPU_DISABLE_OP, 335
L4_PLATFORM_CTL_CPU_ENABLE_OP, 335
L4_platform_ctl_ops, 335
L4_PLATFORM_CTL_SYS_SHUTDOWN_OP, 335
L4_PLATFORM_CTL_SYS_SUSPEND_OP, 335
L4_TASK_ADD_KU_MEM_OP, 335
L4_TASK_CAP_INFO_OP, 335
L4_TASK_LDT_SET_X86_OP, 335
L4_TASK_MAP_OP, 335
L4_TASK_MAP_VGICC_ARM_OP, 335
L4_task_ops, 335
L4_TASK_UNMAP_OP, 335
L4_THREAD_AMD64_GET_SEGMENT_INFO_OP, 336
L4_THREAD_AMD64_SET_SEGMENT_BASE_OP, 336
L4_THREAD_ARM_TPIDRURO_OP, 336
L4_THREAD_CONTROL_OP, 336
L4_THREAD_EX_REGS_OP, 336
L4_THREAD_MODIFY_SENDER_OP, 336
L4_THREAD_OPCODE_MASK, 336
L4_thread_ops, 335
L4_THREAD_REGISTER_DELETE_IRQ_OP, 336
L4_THREAD_STATS_OP, 336
L4_THREAD_SWITCH_OP, 336
L4_THREAD_VCPU_CONTROL_OP, 336
L4_THREAD_VCPU_RESUME_OP, 336
L4_THREAD_X86_GDT_OP, 336
L4_VCON_GET_ATTR_OP, 336
L4_vcon_ops, 336
L4_VCON_READ_OP, 336
L4_VCON_SET_ATTR_OP, 336
L4_VCON_WRITE_OP, 336
L4 kernel object type information, 360
kobject_typeid, 361
L4 Vbus functions, 348
l4vbus_assign_dma_domain, 350
L4vbus_dma_domain_assign_flags, 350
L4VBUS_DMAD_BIND, 350
L4VBUS_DMAD_KERNEL_DMA_SPACE, 350
L4VBUS_DMAD_L4RE_DMA_SPACE, 350
L4VBUS_DMAD_UNBIND, 350
l4vbus_get_device, 351
l4vbus_get_device_by_hid, 352
l4vbus_get_hid, 353
l4vbus_get_next_device, 353
l4vbus_get_resource, 355
l4vbus_is_compatible, 356
l4vbus_release_ioport, 356
l4vbus_release_resource, 357
l4vbus_request_ioport, 358
l4vbus_request_resource, 358
l4vbus_vicu_get_cap, 359
L4 VIRTIO Block Device, 337
L4virtio_block_operations, 337
L4VIRTIO_BLOCK_S_IOERR, 338
L4VIRTIO_BLOCK_S_OK, 338
L4VIRTIO_BLOCK_S_UNSUPP, 338
L4virtio_block_status, 338
L4VIRTIO_BLOCK_T_DISCARD, 338
L4VIRTIO_BLOCK_T_FLUSH, 338
L4VIRTIO_BLOCK_T_GET_ID, 338
L4VIRTIO_BLOCK_T_IN, 338
L4VIRTIO_BLOCK_T_OUT, 338
L4VIRTIO_BLOCK_T_WRITE_ZEROES, 338
L4 VIRTIO Input Device, 338
L4 VIRTIO Interface, 339
L4 VIRTIO Transport Layer, 340
L4_virtio_cmd, 342
L4_virtio_irq_status, 342
L4_virtio_opcodes, 343
L4VIRTIO_CMD_CFG_QUEUE, 342
L4VIRTIO_CMD_MASK, 342
L4VIRTIO_CMD_NONE, 342
L4VIRTIO_CMD_SET_STATUS, 342
l4virtio_config_queue, 345
l4virtio_config_queue_t, 342
l4virtio_config_queues, 345
l4virtio_device_config, 346
l4virtio_device_config_ds, 346
L4virtio_device_ids, 343
l4virtio_device_notification_irq, 346
L4virtio_device_status, 344
L4virtio_feature_bits, 344
L4VIRTIO_FEATURE_CMD_CONFIG, 344
L4VIRTIO_FEATURE_VERSION_1, 344
L4VIRTIO_ID_9P, 343
L4VIRTIO_ID_BALLOON, 343
L4VIRTIO_ID_BLOCK, 343
L4VIRTIO_ID_CAIF, 344
L4VIRTIO_ID_CONSOLE, 343
L4VIRTIO_ID_CRYPTO, 344
L4VIRTIO_ID_GPU, 344
L4VIRTIO_ID_INPUT, 344
L4VIRTIO_ID_NET, 343
L4VIRTIO_ID_RNG, 343
L4VIRTIO_ID_RPMSG, 343
L4VIRTIO_ID_RPROC_SERIAL, 344
L4VIRTIO_ID_SCSI, 343
L4VIRTIO_ID_SOCK, 344
L4VIRTIO_ID_VSOCK, 344
L4VIRTIO_IRQ_STATUS_CONFIG, 343
L4VIRTIO_IRQ_STATUS_VRING, 343
L4VIRTIO_OP_CONFIG_QUEUE, 343
L4VIRTIO_OP_REGISTER_DS, 343
L4VIRTIO_OP_REGISTER_IFACE, 343
L4VIRTIO_OP_SET_STATUS, 343
l4virtio_register_ds, 347
l4virtio_register_iface, 347
l4virtio_set_status, 348
L4VIRTIO_STATUS_ACKNOWLEDGE, 344
L4VIRTIO_STATUS_DEVICE_NEEDS_RESET, 344

L4VIRTIO_STATUS_DRIVER, 344
L4VIRTIO_STATUS_DRIVER_OK, 344
L4VIRTIO_STATUS_FAILED, 344
L4VIRTIO_STATUS_FEATURES_OK, 344
l4/cxx/alloc.h, 1857, 1858
l4/cxx/atomic.h, 2319, 2320
l4/cxx/avl_map, 1858, 1860
l4/cxx/avl_set, 1861, 1863
l4/cxx/avl_tree, 1866, 1868
l4/cxx/basic_ostream, 1872, 1873
l4/cxx/basic_vector.h, 1876, 1877
l4/cxx/bits/bst.h, 1877, 1879
l4/cxx/bits/bst_base.h, 1881, 1883
l4/cxx/bits/bst_iter.h, 1884, 1886
l4/cxx/bits/smart_ptr_list.h, 1887, 1889
l4/cxx/exceptions, 1890, 1892
l4/cxx/iostream, 1894, 1896
l4/cxx/ipc_helper, 1896, 1897
l4/cxx/ipc_server, 2153, 2155
l4/cxx/ipc_stream, 1898, 1908
l4/cxx/l4iostream, 1916, 1917
l4/cxx/l4types.h, 1917, 1918
l4/cxx/main_thread, 1919, 1920
l4/cxx/pair, 1920, 1921
l4/cxx/ref_ptr_list, 1922, 1923
l4/cxx/std_exc_io, 1923, 1924
l4/cxx/string.h, 1925, 1927
l4/cxx/thread, 2272, 2274
l4/cxx/unique_ptr_list, 1927, 1929
l4/irq/irq.h, 1929, 1930
l4/libedid/edid.h, 1942, 1943
l4/libgfxbitmap/bitmap.h, 1943, 1949
l4/libgfxbitmap/font.h, 1949, 1954
l4/libgfxbitmap/support, 1955, 1957
l4/re/c/dataspace.h, 1957, 1958
l4/re/c/debug.h, 1959, 1960
l4/re/c/dma_space.h, 1961, 1963
l4/re/c/event.h, 1964, 1965
l4/re/c/log.h, 1968, 1969
l4/re/c/mem_alloc.h, 1969, 1971
l4/re/c/namespace.h, 1971, 1973
l4/re/c/rm.h, 1973, 1975
l4/re/c/util/cap_alloc.h, 1977, 1978
l4/re/c/util/kumem_alloc.h, 1979, 1980
l4/re/c/util/video/goos_fb.h, 1981, 1982
l4/re/c/video/colors.h, 1982, 1984
l4/re/c/video/goos.h, 1985, 1987
l4/re/c/video/view.h, 1988, 1990
l4/re/cap_alloc, 1991, 1992
l4/re/consts, 2137, 2139
l4/re/consts.h, 2135, 2137
l4/re/dataspace, 1997, 1999
l4/re/dataspace-sys.h, 2000, 2001
l4/re/debug, 2001, 2003
l4/re/dma_space, 2003, 2005
l4/re/elf_aux.h, 2006, 2007
l4/re/env, 2008, 2009
l4/re/env.h, 2010, 2012
l4/re/error_helper, 2014, 2016
l4/re/event.h, 1966, 1967
l4/re/impl/dataspace_impl.h, 2019, 2020
l4/re/impl/mem_alloc_impl.h, 2022, 2023
l4/re/impl/namespace_impl.h, 2023, 2024
l4/re/impl/rm_impl.h, 2026, 2027
l4/re/l4aux.h, 2028, 2029
l4/re/log, 2030, 2031
l4/re/log-sys.h, 2031, 2032
l4/re/mem_alloc, 2032, 2033
l4/re/mem_alloc-sys.h, 2034, 2035
l4/re/mmio_space, 2035, 2037
l4/re/namespace, 2037, 2039
l4/re/namespace-sys.h, 2040, 2041
l4/re/parent, 2041, 2043
l4/re/parent-sys.h, 2043, 2044
l4/re/protocols.h, 2044, 2045
l4/re/random, 2046, 2047
l4/re/rm, 2047, 2049
l4/re/rm-sys.h, 2053
l4/re/shared_cap, 2054, 2055
l4/re/unique_cap, 2059, 2060
l4/re/util/bitmap_cap_alloc, 2063, 2065
l4/re/util/cap, 2066, 2067
l4/re/util/cap_alloc, 1994, 1996
l4/re/util/cap_alloc_impl.h, 2067, 2069
l4/re/util/counting_cap_alloc, 2069, 2071
l4/re/util/event, 2017, 2018
l4/re/util/item_alloc, 2073, 2074
l4/re/util/kumem_alloc, 2075, 2076
l4/re/util/region_mapping, 2076, 2078
l4/re/util/shared_cap, 2056, 2058
l4/re/util/unique_cap, 2061, 2063
l4/re/video/goos-sys.h, 2083
l4/shmc/shmc.h, 2084, 2086
l4/sigma0/sigma0.h, 2088, 2091
l4/sys/__kernel_object_impl.h, 2092, 2093
l4/sys/__ktrace-impl.h, 2093, 2094
l4/sys/__typeinfo.h, 2095, 2098
l4/sys/__vm-arm.h, 2108, 2109
l4/sys/arm_smccc, 2110, 2111
l4/sys/arm_smccc.h, 2111, 2113
l4/sys/assert.h, 2308, 2310
l4/sys/cache.h, 2114, 2116
l4/sys/capability, 2122, 2124
l4/sys/compiler.h, 2125, 2127
l4/sys/consts.h, 2129, 2131
l4/sys/cxx/ipc_client, 2139, 2141
l4/sys/cxx/ipc_iface, 2142, 2148
l4/sys/cxx/ipc_types, 2156, 2158
l4/sys/cxx/smart_capability_1x, 2164, 2165
l4/sys/cxx/types, 2168, 2169
l4/sys/debugger, 2172, 2173
l4/sys/debugger.h, 2174, 2178
l4/sys/err.h, 2181, 2182
l4/sys/exception, 2182, 2184
l4/sys/factory, 2184, 2186
l4/sys/factory.h, 2188, 2189

l4/sys/icu, 2193, 2194
l4/sys/icu.h, 2195, 2198
l4/sys/ipc.h, 2201, 2203
l4/sys/ipc_gate, 2209, 2211
l4/sys/ipc_gate.h, 2211, 2214
l4/sys/irq, 2215, 2216
l4/sys/irq.h, 1938, 1940
l4/sys/kernel_object.h, 2218, 2219
l4/sys/kip, 2220, 2222
l4/sys/kip.h, 2350, 2351
l4/sys/ktrace.h, 2223, 2225
l4/sys/l4int.h, 2225, 2226
l4/sys/memdesc.h, 2229, 2231
l4/sys/meta, 2233, 2235
l4/sys/pager, 2235, 2237
l4/sys/platform_control, 2237, 2239
l4/sys/platform_control.h, 2239, 2241
l4/sys/rcv_endpoint, 2243, 2245
l4/sys/rcv_endpoint.h, 2245, 2247
l4/sys/scheduler, 2248, 2249
l4/sys/scheduler.h, 2249, 2252
l4/sys/semaphore, 2254, 2256
l4/sys/semaphore.h, 2256, 2258
l4/sys/smart_capability, 2259, 2260
l4/sys/task, 2262, 2264
l4/sys/task.h, 2265, 2266
l4/sys/thread, 2269, 2271
l4/sys/thread.h, 2373, 2375
l4/sys/typeinfo_svr, 2274, 2276
l4/sys/types.h, 2277, 2279
l4/sys/utcb.h, 2282, 2284
l4/sys/vcon, 2294, 2296
l4/sys/vcon.h, 2296, 2299
l4/sys/vhw.h, 2302, 2304
l4/sys/vm, 2305, 2306
l4/util/assert.h, 2306, 2307
l4/util/atomic.h, 2311, 2314
l4/util/backtrace.h, 2320, 2322
l4/util/base64.h, 2322, 2324
l4/util/bitops.h, 2324, 2326
l4/util/elf.h, 2329, 2335
l4/util/getopt.h, 2345, 2346
l4/util/keymap.h, 2347, 2348
l4/util/kip.h, 2348, 2349
l4/util/kprintf.h, 2353
l4/util/l4_macros.h, 1847, 1848
l4/util/list_alloc.h, 2354, 2356
l4/util/lock.h, 2357, 2358
l4/util/mb_info.h, 2358, 2362
l4/util/parse_cmd.h, 2366, 2367
l4/util/rand.h, 2367, 2369
l4/util/splitlog2.h, 2369, 2370
l4/util/thread.h, 2371, 2372
l4/vbus/vbus.h, 2383, 2386
l4/vbus/vbus_interfaces.h, 2387, 2389
l4/vbus/vbus_types.h, 2390, 2392
L4::Alloc_list, 858
L4::Arm_smccc, 859
call, 859
L4::Base_exception, 861
L4::Basic_registry, 863
dispatch, 864
find, 864
L4::Bounds_error, 865
L4::Cap< T >, 868
Cap, 870, 871
copy, 872
move, 872
L4::Cap_base, 873
cap, 877
Cap_base, 876
Cap_type, 875
copy, 877
fpage, 878
Invalid, 876
is_valid, 879
move, 880
No_init, 876
No_init_type, 876
snd_base, 881
validate, 882
L4::Com_error, 883
Com_error, 886
L4::Debugger, 886
get_object_name, 889
global_id, 890
kobj_to_id, 890
query_log_name, 890
query_log_typeid, 891
set_object_name, 892
switch_log, 892
L4::Element_already_exists, 893
L4::Element_not_found, 895
L4::Epiface, 897
dispatch, 898
get_buffer_demand, 899
obj_cap, 899
server_iface, 899
set_server, 900
L4::Epiface_t< Derived, IFACE, BASE, bool >, 901
dispatch, 903
L4::Epiface_t0< RPC_IFACE, BASE >, 904
obj_cap, 906
L4::Exception, 906
Rpcs, 907
L4::Exception_tracer, 907
L4::Factory, 909
create, 912, 913
create_factory, 914
create_gate, 915
create_task, 916
L4::Factory::Lstr, 917
Lstr, 918
L4::Factory::Nil, 919
L4::Factory::S, 920
operator l4_msgtag_t, 922

put, 922–924
S, 921
L4::Icu, 924
bind, 927
info, 929
mask, 929
msi_info, 930
set_mode, 931
unbind, 931
L4::Icu::Info, 932
L4::Invalid_capability, 934
cap, 936
Invalid_capability, 936
L4::Io_pager, 937
io_page_fault, 939
L4::Iommu, 939
bind, 942
unbind, 942
L4::IOModifier, 942
L4::Ipc, 620
buf_cp_in, 622
buf_cp_out, 623
buf_in, 623
make_cap, 624
make_cap_full, 624
make_cap_rw, 625
make_cap_rws, 626
msg_ptr, 627
read, 627
str_cp_in, 627
L4::Ipc::Array< ELEM_TYPE, LEN_TYPE >, 943
L4::Ipc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >, 946
L4::Ipc::Array_ref< ELEM_TYPE, LEN_TYPE >, 947
L4::Ipc::As_value< T >, 949
L4::Ipc::Buf_item, 950
L4::Ipc::Call, 951
L4::Ipc::Call_t< RIGHTS >, 952
L4::Ipc::Call_zero_send_timeout, 953
L4::Ipc::Cap< T >, 955
Cap, 956
Cap_mask, 956
from_ci, 957
Rights_mask, 956
L4::Ipc::Gen_fpage< T >, 957
cap_received, 959
id_received, 959
is_compound, 959
local_id_received, 960
L4::Ipc::In_out< T >, 960
L4::Ipc::Iostream, 961
call, 965
get, 966, 967
Iostream, 965
put, 967, 968
reply_and_wait, 968, 969
reset, 969
L4::Ipc::Istream, 970
get, 974, 975
Istream, 973
receive, 976
reset, 976
skip, 977
tag, 977
wait, 978, 979
L4::Ipc::Msg, 628
align_to, 630
Br_bytes, 629
check_size, 631, 632
Item_bytes, 629
Item_words, 629
Mr_bytes, 629
Mr_words, 629
msg_add, 632
msg_get, 633
Word_bytes, 629
L4::Ipc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >, 980
L4::Ipc::Msg::Cls_buffer, 982
L4::Ipc::Msg::Cls_data, 983
L4::Ipc::Msg::Cls_item, 984
L4::Ipc::Msg::Dir_in, 985
L4::Ipc::Msg::Dir_out, 986
L4::Ipc::Msg::Do_in_data, 987
L4::Ipc::Msg::Do_in_items, 988
L4::Ipc::Msg::Do_out_data, 989
L4::Ipc::Msg::Do_out_items, 990
L4::Ipc::Msg::Do_rcv_buffers, 991
L4::Ipc::Msg::Elem< Array< A, LEN > >, 994
L4::Ipc::Msg::Elem< Array< A, LEN > & >, 993
L4::Ipc::Msg::Elem< Array_ref< A, LEN > & >, 995
L4::Ipc::Msg::Is_valid_rpc_type< T >, 996
L4::Ipc::Msg::Svr_arg_pack< IPC_TYPE >, 998
L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >, 998
L4::Ipc::Msg_ptr< T >, 1000
Msg_ptr, 1000
L4::Ipc::Opt< T >, 1001
L4::Ipc::Ostream, 1002
put, 1005, 1006
send, 1006
tag, 1007
L4::Ipc::Out< T >, 1008
L4::Ipc::Ret_array< T >, 1009
L4::Ipc::Send_only, 1010
L4::Ipc::Small_buf, 1011
Small_buf, 1011, 1012
L4::Ipc::Snd_item, 1012
L4::Ipc::Str_cp_in< T >, 1013
Str_cp_in, 1014
L4::Ipc::Varg, 1014
data, 1016
get_value, 1016
is_nil, 1017
is_of, 1017
is_of_int, 1018

length, 1018
tag, 1019
type, 1019
value, 1019
L4::Ipc::Varg_list< MAX >, 1020
L4::Ipc::Varg_list_ref, 1022
Varg_list_ref, 1023
L4::Ipc::Varg_list_ref::Iterator, 1024
L4::Ipc_gate, 1025
get_infos, 1027
L4::Ipc_svr, 633
L4::Ipc_svr::Br_manager_no_buffers, 1028
alloc_buffer_demand, 1031
L4::Ipc_svr::Compound_reply, 1032
L4::Ipc_svr::Default_loop_hooks, 1033
L4::Ipc_svr::Default_setup_wait, 1035
L4::Ipc_svr::Default_timeout, 1036
L4::Ipc_svr::Direct_dispatch< R >, 1038
L4::Ipc_svr::Direct_dispatch< R * >, 1040
L4::Ipc_svr::Exc_dispatch< R, Exc >, 1041
L4::Ipc_svr::Ignore_errors, 1042
L4::Ipc_svr::Server_iface, 1044
add_timeout, 1047
alloc_buffer_demand, 1047
get_rcv_cap, 1048
rcv_cap, 1048, 1049
realloc_rcv_cap, 1050
remove_timeout, 1051
L4::Ipc_svr::Timeout, 1051
expired, 1053
timeout, 1053
L4::Ipc_svr::Timeout_queue, 1054
add, 1055
handle_expired_timeouts, 1056
next_timeout, 1057
remove, 1058
timeout_expired, 1059
L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >, 1060
add_timeout, 1063
remove_timeout, 1064
L4::Irq, 1065
detach, 1068
receive, 1069
unmask, 1070
wait, 1070
L4::Irq_eoi, 1071
unmask, 1073
L4::Irq_handler_object, 1074
L4::Irq_mux, 1076
chain, 1079
L4::Irqep_t< Derived, BASE, bool >, 1080
dispatch, 1083
obj_cap, 1084
L4::Kip::Mem_desc, 1084
all, 1087, 1088
Arch, 1087
Bootloader, 1087
Conventional, 1087
count, 1089
Dedicated, 1087
end, 1090
first, 1091
Info, 1087
Info_acpi_rsdp, 1086
Info_sub_type, 1086
is_virtual, 1092
Mem_desc, 1087
Mem_type, 1086
Reserved, 1087
set, 1092
Shared, 1087
size, 1093
start, 1093
sub_type, 1093
type, 1094
Undefined, 1087
L4::Kobject, 1094
cap, 1095
dec_refcnt, 1096
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, 1097
__Iface, 1099
__Iface_list, 1099
__check_protocols__, 1100
c, 1100
Class, 1099
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, 1101
__Iface, 1103
__Iface_list, 1103
__check_protocols__, 1104
c, 1104
Class, 1103
L4::Kobject_demand< T >, 1105
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >, 1105
L4::Kobject_typeid< T >, 1107
Demand, 1108
demand, 1108
id, 1109
proto_dispatch, 1109
L4::Kobject_typeid< void >, 1110
L4::Kobject_x< Derived, ARGS >, 1111
L4::Meta, 1112
interface, 1115
num_interfaces, 1115
supports, 1115
L4::Out_of_memory, 1116
L4::Pager, 1119
page_fault, 1122
L4::Platform_control, 1123
cpu_allow_shutdown, 1125
cpu_disable, 1125
cpu_enable, 1125
system_shutdown, 1126

system_suspend, 1126
L4::Poll_timeout_kipclock, 1127
Poll_timeout_kipclock, 1127
set, 1128
test, 1129
timed_out, 1130
L4::Proto_t< P >, 1130
L4::Rcv_endpoint, 1131
bind_thread, 1134
L4::Registry_iface, 1135
register_irq_obj, 1136
register_obj, 1137, 1138
unregister_obj, 1138
L4::Runtime_error, 1139
err_no, 1141
extra_str, 1141
Runtime_error, 1141
L4::Scheduler, 1142
idle_time, 1144
info, 1145
is_online, 1146
run_thread, 1146
L4::Semaphore, 1147
down, 1150
up, 1151
L4::Server< LOOP_HOOKS >, 1151
internal_loop, 1154
loop, 1155
Server, 1154
L4::Server_object, 1156
dispatch, 1159
L4::Server_object_t< IFACE, BASE >, 1161
dispatch_meta_request, 1163
get_buffer_demand, 1163
proto_dispatch, 1164
L4::Server_object_x< Derived, IFACE, BASE >, 1165
L4::Smart_cap< T, SMART >, 1168
Smart_cap, 1171
L4::String, 1171
L4::Task, 1172
add_ku_mem, 1175
cap_equal, 1176
cap_valid, 1176
delete_obj, 1177
map, 1177
release_cap, 1178
unmap, 1178
unmap_batch, 1179
L4::Thread, 1180
control, 1184
ex_regs, 1184, 1185
modify_senders, 1186
register_del_irq, 1187
stats_time, 1188
switch_to, 1188
vcpu_control, 1189
vcpu_control_ext, 1190
vcpu_resume_commit, 1190
vcpu_resume_start, 1191
L4::Thread::Attr, 1191
alien, 1193
Attr, 1192
bind, 1193
exc_handler, 1194
pager, 1195
ux_host_syscall, 1195
L4::Thread::Modify_senders, 1196
add, 1196
L4::Triggerable, 1198
trigger, 1200
L4::Type_info, 1201
L4::Type_info::Demand, 1202
Demand, 1203
no_demand, 1204
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, 1205
Caps, 1207
Flags, 1207
Mem, 1207
Ports, 1207
L4::Type_info::Demand_union_t< D1, D2 >, 1207
L4::Typeid, 634
L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y >, 1211
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >, 1212
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y >, 1213
L4::Typeid::Detail::_Rpcs< OPCODE, O, X >, 1210
L4::Typeid::Detail::Rpcs_end, 1214
L4::Typeid::P_dispatch< LIST >, 1215
L4::Typeid::Raw_ipc< CLASS >, 1216
L4::Typeid::Rpc_nocode< OPERATION >, 1216
L4::Typeid::Rpcs< RPCS >, 1219
L4::Typeid::Rpcs_code< OPCODE_TYPE >, 1221
L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS >, 1222
L4::Typeid::Rpcs_sys< ARG >, 1224
L4::Types, 635
L4::Types::Bool< V >, 1226
L4::Types::False, 1228
L4::Types::Flags< BITS_ENUM, UNDERLYING >, 1229
clear, 1233
Flags, 1232
from_raw, 1233
None, 1232
None_type, 1232
L4::Types::Flags_ops_t< DT >, 1234
L4::Types::Flags_t< DT, T >, 1236
L4::Types::Int_for_size< SIZE, bool >, 1238
L4::Types::Int_for_type< T >, 1239
L4::Types::Same< A, B >, 1240
L4::Types::True, 1242
L4::Unknown_error, 1244
L4::Vcon, 1247

get_attr, 1249
read, 1250
read_with_flags, 1251
send, 1252
set_attr, 1252
write, 1253
L4::Vm, 1254
l4_addr_consts_t
Memory related, 408
l4_align_stack_for_direct_fncall
Basic Macros, 135
L4_ALWAYS_INLINE
Basic Macros, 134
L4_AMD64_SEGMENT_FS
segment.h, 1810
L4_AMD64_SEGMENT_GS
segment.h, 1810
l4_arm_smccc_call
arm_smccc.h, 2112
l4_assert
assert.h, 2310
L4_BASE_ARM_SMCCC_CAP
Capabilities, 154
L4_BASE_CAPS_LAST
Capabilities, 154
L4_BASE_DEBUGGER_CAP
Capabilities, 154
L4_BASE_FACTORY_CAP
Capabilities, 154
L4_BASE_ICU_CAP
Capabilities, 154
L4_BASE_IOMMU_CAP
Capabilities, 154
L4_BASE_LOG_CAP
Capabilities, 154
L4_BASE_PAGER_CAP
Capabilities, 154
L4_BASE_SCHEDULER_CAP
Capabilities, 154
L4_BASE_TASK_CAP
Capabilities, 154
L4_BASE_THREAD_CAP
Capabilities, 154
L4_BDR_IO_SHIFT
Buffer Registers (BRs), 144
L4_BDR_MEM_SHIFT
Buffer Registers (BRs), 144
L4_BDR_OBJ_SHIFT
Buffer Registers (BRs), 144
l4_buf_regs_t, 1257
l4_buffer_desc_consts_t
Buffer Registers (BRs), 144
l4_busy_wait_ns
Timestamp Counter, 534
l4_busy_wait_us
Timestamp Counter, 535
l4_bytes_to_mwords
Memory related, 408
l4_cache_clean_data
Cache Consistency, 150
l4_cache_coherent
Cache Consistency, 150
l4_cache_dma_coherent
Cache Consistency, 151
l4_cache_flush_data
Cache Consistency, 151
l4_cache_inv_data
Cache Consistency, 152
l4_calibrate_tsc
Timestamp Counter, 535
l4_cap_consts_t
Capabilities, 153
L4_CAP_FPAGE_D
Flex pages, 241
L4_CAP_FPAGE_R
Flex pages, 241
L4_cap_fpage_rights
Flex pages, 241
L4_CAP_FPAGE_RO
Flex pages, 241
L4_CAP_FPAGE_RS
Flex pages, 241
L4_CAP_FPAGE_RSD
Flex pages, 242
L4_CAP_FPAGE_RW
Flex pages, 241
L4_CAP_FPAGE_RWD
Flex pages, 242
L4_CAP_FPAGE_RWS
Flex pages, 242
L4_CAP_FPAGE_RWSD
Flex pages, 242
L4_CAP_FPAGE_S
Flex pages, 241
L4_CAP_FPAGE_W
Flex pages, 241
L4_CAP_MASK
Capabilities, 154
L4_CAP_SHIFT
Capabilities, 154
L4_CAP_SIZE
Capabilities, 154
l4_capability_equal
Capabilities, 155
l4_capability_next
types.h, 2279
l4_debugger_get_object_name
debugger.h, 2176
l4_debugger_global_id
Kernel Debugger, 318
l4_debugger_kobj_to_id
Kernel Debugger, 318
l4_debugger_query_log_name
debugger.h, 2176
l4_debugger_query_log_typeid
debugger.h, 2177

l4_debugger_set_object_name
Kernel Debugger, 319
l4_debugger_switch_log
debugger.h, 2177
l4_default_caps_t
Capabilities, 154
L4_DISABLE_COPY
capability, 2123
L4_E2BIG
Error codes, 217
L4_EACCESS
Error codes, 217
L4_EADDRNOTAVAIL
Error codes, 217
L4_EAGAIN
Error codes, 217
L4_EBADPROTO
Error codes, 217
L4_EBUSY
Error codes, 217
L4_EEXIST
Error codes, 217
L4_EFAULT
Error codes, 217
L4_EINVAL
Error codes, 217
L4_EIO
Error codes, 217
L4_EIPC_HI
Error codes, 217
L4_EIPC_LO
Error codes, 217
L4_EMSGMISSARG
Error codes, 217
L4_EMSGTOOLONG
Error codes, 217
L4_EMSGTOOSHORT
Error codes, 217
L4_ENAMETOOLONG
Error codes, 217
L4_ENODEV
Error codes, 217
L4_ENOENT
Error codes, 217
L4_ENOMEM
Error codes, 217
L4_ENOREPLY
Error codes, 217
L4_ENOSPC
Error codes, 217
L4_ENOSYS
Error codes, 217
L4_ENXIO
Error codes, 217
L4_EOK
Error codes, 217
L4_EPERM
Error codes, 217
L4_ERANGE
Error codes, 217
L4_ERRNOMAX
Error codes, 217
l4_error
Error Handling, 212
l4_error_code_t
Error codes, 216
l4_exc_regs_t, 1258
flags, 1260
ss, 1260
l4_factory_create
Factory, 227
l4_factory_create_factory
Factory, 227
l4_factory_create_gate
Factory, 228
l4_factory_create_irq
Factory, 228
l4_factory_create_task
Factory, 229
l4_factory_create_thread
Factory, 230
l4_factory_create_vm
Factory, 230
L4_FP_ALL_SPACES
Task, 496
L4_FP_DELETE_OBJ
Task, 496
L4_FP_OTHER_SPACES
Task, 496
l4_fpage
Flex pages, 244
L4_FPAGE_ADDR_BITS
Flex pages, 243
L4_FPAGE_ADDR_SHIFT
Flex pages, 243
l4_fpage_all
Flex pages, 245
L4_FPAGE_BUFFERABLE
Flex pages, 242
L4_FPAGE_C_IPCGATE_SVR
Flex pages, 244
L4_FPAGE_C_NO_REF_CNT
Flex pages, 244
L4_FPAGE_C_OBJ_RIGHT1
Flex pages, 244
L4_FPAGE_C_OBJ_RIGHT2
Flex pages, 244
L4_FPAGE_C_OBJ_RIGHT3
Flex pages, 244
L4_FPAGE_C_OBJ_RIGHTS
Flex pages, 244
L4_FPAGE_C_REF_CNT
Flex pages, 244
L4_FPAGE_CACHE_OPT
Flex pages, 242
l4_fpage_cacheability_opt_t

Flex pages, 242
L4_FPAGE_CACHEABLE
Flex pages, 242
l4_fpage_consts
Flex pages, 242
l4_fpage_contains
Flex pages, 245
L4_fpage_control
Flex pages, 243
L4_FPAGE_CONTROL_MASK
Flex pages, 243
L4_FPAGE_CONTROL_OFFSET_SHIFT
Flex pages, 243
l4_fpage_invalid
Flex pages, 246
L4_FPAGE_IO
Flex pages, 244
l4_fpage_ioport
Flex pages, 246
l4_fpage_max_order
Flex pages, 247
l4_fpage_memaddr
Flex pages, 248
L4_FPAGE_MEMORY
Flex pages, 244
L4_FPAGE_OBJ
Flex pages, 244
l4_fpage_obj
Flex pages, 248
l4_fpage_page
Flex pages, 249
L4_fpage_rights
Flex pages, 243
l4_fpage_rights
Flex pages, 249
L4_FPAGE_RIGHTS_BITS
Flex pages, 243
L4_FPAGE_RIGHTS_MASK
Flex pages, 243
L4_FPAGE_RIGHTS_SHIFT
Flex pages, 243
L4_FPAGE_RO
Flex pages, 243
L4_FPAGE_RW
Flex pages, 243
L4_FPAGE_RWX
Flex pages, 243
L4_FPAGE_RX
Flex pages, 243
l4_fpage_set_rights
Flex pages, 250
l4_fpage_size
Flex pages, 251
L4_FPAGE_SIZE_BITS
Flex pages, 243
L4_FPAGE_SIZE_SHIFT
Flex pages, 243
L4_FPAGE_SPECIAL
Flex pages, 244
l4_fpage_t, 1261
L4_fpage_type
Flex pages, 244
l4_fpage_type
Flex pages, 251
L4_FPAGE_TYPE_BITS
Flex pages, 243
L4_FPAGE_TYPE_SHIFT
Flex pages, 243
L4_FPAGE_UNCACHEABLE
Flex pages, 242
L4_FPAGE_W
Flex pages, 243
L4_FPAGE_X
Flex pages, 243
l4_get_hz
Timestamp Counter, 536
L4_HIDDEN
Basic Macros, 134
l4_icu_bind
Interrupt controller, 306
l4_icu_bind_u
Interrupt controller, 307
L4_ICU_FLAG_MSI
Interrupt controller, 306
L4_icu_flags
Interrupt controller, 305
l4_icu_info
Interrupt controller, 308
l4_icu_info_t, 1261
features, 1263
Interrupt controller, 305
l4_icu_info_u
Interrupt controller, 308
l4_icu_mask
Interrupt controller, 309
l4_icu_mask_u
Interrupt controller, 310
l4_icu_msi_info
Interrupt controller, 311
l4_icu_msi_info_t, 1263
l4_icu_msi_info_u
Interrupt controller, 312
L4_ICU_OP_BIND
L4 IPC Opcodes, 334
L4_ICU_OP_INFO
L4 IPC Opcodes, 334
L4_ICU_OP_MASK
L4 IPC Opcodes, 334
L4_ICU_OP_MSI_INFO
L4 IPC Opcodes, 334
L4_ICU_OP_SET_MODE
L4 IPC Opcodes, 334
L4_ICU_OP_UNBIND
L4 IPC Opcodes, 334
L4_ICU_OP_UNMASK
L4 IPC Opcodes, 334

L4_icu_opcode
L4 IPC Opcodes, 333
l4_icu_set_mode
Interrupt controller, 312
l4_icu_set_mode_u
Interrupt controller, 313
l4_icu_unbind
Interrupt controller, 314
l4_icu_unbind_u
Interrupt controller, 315
l4_icu_unmask
Interrupt controller, 316
l4_icu_unmask_u
Interrupt controller, 316
L4_INLINE_RPC
ipc_iface, 2144
L4_INLINE_RPC_NF
ipc_iface, 2145
L4_INLINE_RPC_NF_OP
ipc_iface, 2145
L4_INLINE_RPC_OP
ipc_iface, 2146
L4_INVALID_ADDR
Memory related, 408
L4_INVALID_CAP
Capabilities, 154
l4_iofpage
Flex pages, 252
L4_IOPORT_MAX
Flex pages, 240
l4_ipc
Object Invocation, 435
l4_ipc_call
Object Invocation, 436
L4_IPC_ENOT_EXISTENT
Error Handling, 211
l4_ipc_error
Error Handling, 213
l4_ipc_error_code
Error Handling, 214
L4_IPC_ERROR_MASK
Error Handling, 211
L4_IPC_GATE_BIND_OP
L4 IPC Opcodes, 335
L4_IPC_GATE_GET_INFO_OP
L4 IPC Opcodes, 335
l4_ipc_gate_get_infos
IPC-Gate API, 269
L4_ipc_gate_ops
L4 IPC Opcodes, 334
l4_ipc_is_rcv_error
Error Handling, 215
l4_ipc_is_snd_error
Error Handling, 215
L4_IPC_REABORTED
Error Handling, 212
L4_IPC_RECANCELED
Error Handling, 212
l4_ipc_receive
Object Invocation, 437
L4_IPC_REMAPFAILED
Error Handling, 212
L4_IPC_REMSGCUT
Error Handling, 212
l4_ipc_reply_and_wait
Object Invocation, 438
L4_IPC_RERCVPFTO
Error Handling, 212
L4_IPC_RESNDPFTO
Error Handling, 212
L4_IPC_RETIMEOUT
Error Handling, 211
L4_IPC_SEABORTED
Error Handling, 212
L4_IPC_SECANCELED
Error Handling, 212
L4_IPC_SEMAPFAILED
Error Handling, 212
L4_IPC_SEMSGCUT
Error Handling, 212
l4_ipc_send
Object Invocation, 440
l4_ipc_send_and_wait
Object Invocation, 441
L4_IPC_SERCVPFTO
Error Handling, 212
L4_IPC_SESNDPFTO
Error Handling, 212
L4_IPC_SETIMEOUT
Error Handling, 212
l4_ipc_sleep
Object Invocation, 442
L4_IPC_SND_ERR_MASK
Error Handling, 211
l4_ipc_tcr_error_t
Error Handling, 211
l4_ipc_timeout
Timeouts, 527
L4_IPC_TIMEOUT_0
Timeouts, 527
l4_ipc_to_errno
ipc.h, 2203
l4_ipc_wait
Object Invocation, 443
l4_irq_detach
IRQs, 274
l4_irq_detach_u
IRQs, 275
L4_IRQ_F_BOTH
IRQs, 274
L4_IRQ_F_BOTH_EDGE
IRQs, 274
L4_IRQ_F_CLEAR_WAKEUP
IRQs, 274
L4_IRQ_F_EDGE
IRQs, 274

L4_IRQ_F_LEVEL
IRQs, 274
L4_IRQ_F_LEVEL_HIGH
IRQs, 274
L4_IRQ_F_LEVEL_LOW
IRQs, 274
L4_IRQ_F_MASK
IRQs, 274
L4_IRQ_F_NEG
IRQs, 274
L4_IRQ_F_NEG_EDGE
IRQs, 274
L4_IRQ_F_NONE
IRQs, 274
L4_IRQ_F_POS
IRQs, 274
L4_IRQ_F_POS_EDGE
IRQs, 274
L4_IRQ_F_SET_WAKEUP
IRQs, 274
L4_irq_mode
IRQs, 273
l4_irq_mux_chain
IRQs, 275
l4_irq_mux_chain_u
IRQs, 276
l4_irq_receive
IRQs, 277
l4_irq_receive_u
IRQs, 278
l4_irq_trigger
IRQs, 279
l4_irq_trigger_u
IRQs, 280
l4_irq_unmask
IRQs, 281
l4_irq_unmask_u
IRQs, 282
l4_irq_wait
IRQs, 283
l4_irq_wait_u
IRQs, 284
l4_is_fpage_writable
Flex pages, 252
l4_is_invalid_cap
Capabilities, 155
l4_is_valid_cap
Capabilities, 156
L4_ITEM_CONT
Message Items, 413
L4_ITEM_MAP
Message Items, 413
l4_kernel_info_get_mem_desc_end
Memory descriptors (C version), 400
l4_kernel_info_get_mem_desc_is_virtual
Memory descriptors (C version), 400
l4_kernel_info_get_mem_desc_start
Memory descriptors (C version), 401
l4_kernel_info_get_mem_desc_subtype
Memory descriptors (C version), 401
l4_kernel_info_get_mem_desc_type
Memory descriptors (C version), 401
l4_kernel_info_get_num_mem_descs
Memory descriptors (C version), 402
l4_kernel_info_mem_desc_t, 1264
Memory descriptors (C version), 399
l4_kernel_info_set_mem_desc
Memory descriptors (C version), 402
l4_kernel_info_t, 1265
l4_kernel_info_version_offset
Kernel Interface Page, 321
l4_kip_clock
Kernel Interface Page, 322
l4_kip_clock_lw
Kernel Interface Page, 323
l4_kip_clock_ns
Kernel Interface Page, 324
L4_KIP_OFFS_READ_NS
Kernel Interface Page, 321
L4_KIP_OFFS_READ_US
Kernel Interface Page, 321
l4_kip_version
Kernel Interface Page, 324
l4_kip_version_string
Kernel Interface Page, 325
L4_LOG2_PAGESIZE
Memory related, 407
L4_LOG2_SUPERPAGESIZE
Memory related, 407
l4_map_control
Message Items, 413
L4_MAP_ITEM_GRANT
Message Items, 413
L4_MAP_ITEM_MAP
Message Items, 413
l4_map_obj_control
Message Items, 414
l4_mem_info_acpi_rsdp
Memory descriptors (C version), 400
l4_mem_info_sub_type_t
Memory descriptors (C version), 399
L4_mem_op_widths
Memory operations., 403
l4_mem_read
Memory operations., 404
l4_mem_type_archspecific
Memory descriptors (C version), 400
l4_mem_type_bootloader
Memory descriptors (C version), 400
l4_mem_type_conventional
Memory descriptors (C version), 400
l4_mem_type_dedicated
Memory descriptors (C version), 400
l4_mem_type_info
Memory descriptors (C version), 400
l4_mem_type_reserved

Memory descriptors (C version), 400
l4_mem_type_shared
Memory descriptors (C version), 400
l4_mem_type_t
Memory descriptors (C version), 400
l4_mem_type_undefined
Memory descriptors (C version), 400
L4_MEM_WIDTH_1BYTE
Memory operations., 404
L4_MEM_WIDTH_2BYTE
Memory operations., 404
L4_MEM_WIDTH_4BYTE
Memory operations., 404
l4_mem_write
Memory operations., 404
l4_msg_item_consts_t
Message Items, 412
l4_msg_regs_t, 1267
l4_msgtag
Message Tag, 419
L4_MSGTAG_ERROR
Message Tag, 417
L4_MSGTAG_FLAGS
Message Tag, 417
l4_msgtag_flags
Message Tag, 417, 420
l4_msgtag_has_error
Message Tag, 420
l4_msgtag_is_exception
Message Tag, 421
l4_msgtag_is_io_page_fault
Message Tag, 422
l4_msgtag_is_page_fault
Message Tag, 422
l4_msgtag_is_preemption
Message Tag, 423
l4_msgtag_is_sigma0
Message Tag, 424
l4_msgtag_is_sys_exception
Message Tag, 424
l4_msgtag_items
Message Tag, 425
l4_msgtag_label
Message Tag, 426
l4_msgtag_protocol
Message Tag, 418
L4_MSGTAG_SCHEDULE
Message Tag, 417
l4_msgtag_t, 1268
flags, 1269
Message Tag, 417
L4_MSGTAG_TRANSFER_FPU
Message Tag, 417
l4_msgtag_words
Message Tag, 427
L4_NOTHROW
Basic Macros, 134
l4_ns_to_tsc
Timestamp Counter, 536
l4_obj_fpage
Flex pages, 253
L4_obj_fpage_ctl
Flex pages, 244
L4_PAGEMASK
Memory related, 407
l4_platform_ctl_cpu_allow_shutdown
Platform Control C API, 446
L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP
L4 IPC Opcodes, 335
l4_platform_ctl_cpu_disable
Platform Control C API, 447
L4_PLATFORM_CTL_CPU_DISABLE_OP
L4 IPC Opcodes, 335
l4_platform_ctl_cpu_enable
Platform Control C API, 447
L4_PLATFORM_CTL_CPU_ENABLE_OP
L4 IPC Opcodes, 335
L4_platform_ctl_ops
L4 IPC Opcodes, 335
L4_platform_ctl_proto
Message Tag, 418
L4_PLATFORM_CTL_SYS_SHUTDOWN_OP
L4 IPC Opcodes, 335
L4_PLATFORM_CTL_SYS_SUSPEND_OP
L4 IPC Opcodes, 335
l4_platform_ctl_system_shutdown
Platform Control C API, 448
l4_platform_ctl_system_suspend
Platform Control C API, 448
L4_PROTO_ALLOW_SYSCALL
Message Tag, 418
L4_PROTO_DEBUGGER
Message Tag, 418
L4_PROTO_DMA_SPACE
Message Tag, 418
L4_PROTO_EXCEPTION
Message Tag, 418
L4_PROTO_FACTORY
Message Tag, 418
L4_PROTO_IO_PAGE_FAULT
Message Tag, 418
L4_PROTO_IOMMU
Message Tag, 418
L4_PROTO_IRQ
Message Tag, 418
L4_PROTO_IRQ_MUX
Message Tag, 418
L4_PROTO_IRQ_SENDER
Message Tag, 418
L4_PROTO_KOBJECT
Message Tag, 418
L4_PROTO_LOG
Message Tag, 418
L4_PROTO_META
Message Tag, 418
L4_PROTO_NONE

Message Tag, 418
L4_PROTO_PAGE_FAULT
Message Tag, 418
L4_PROTO_PF_EXCEPTION
Message Tag, 418
L4_PROTO_PLATFORM_CTL
Message Tag, 419
L4_PROTO_PREEMPTION
Message Tag, 418
L4_PROTO_SCHEDULER
Message Tag, 418
L4_PROTO_SEMAPHORE
Message Tag, 418
L4_PROTO_SIGMA0
Message Tag, 418
L4_PROTO_SMCCC
Message Tag, 418
L4_PROTO_SYS_EXCEPTION
Message Tag, 418
L4_PROTO_TASK
Message Tag, 418
L4_PROTO_THREAD
Message Tag, 418
L4_PROTO_VM
Message Tag, 418
L4_RCV_EP_BIND_OP
rcv_endpoint.h, 2247
l4_rcv_ep_bind_thread
IPC-Gate API, 270
L4_rcv_ep_ops
rcv_endpoint.h, 2247
L4_RCV_ITEM_LOCAL_ID
Message Items, 413
L4_RCV_ITEM_SINGLE_CAP
Message Items, 413
l4_rcv_timeout
Timeouts, 528
l4_rdpmc
Timestamp Counter, 537
l4_rdpmc_32
Timestamp Counter, 537
l4_rdtsc
Timestamp Counter, 538
l4_rdtsc_32
Timestamp Counter, 538
l4_round_page
Memory related, 408
l4_round_size
Memory related, 409
L4_RPC
ipc_iface, 2146
L4_RPC_DEF
ipc_client, 2141
L4_RPC_NF
ipc_iface, 2147
L4_RPC_NF_OP
ipc_iface, 2147
L4_RPC_OP
ipc_iface, 2148
l4_sched_cpu_set
Scheduler, 471
l4_sched_cpu_set_t, 1270
gran_offset, 1272
granularity, 1270
offset, 1271
set, 1271
l4_sched_param_t, 1273
L4_SCHEDULER_CLASS_FIXED_PRIO
Scheduler, 470
L4_SCHEDULER_CLASS_WFQ
Scheduler, 470
L4_scheduler_classes
Scheduler, 470
l4_scheduler_idle_time
Scheduler, 471
L4_SCHEDULER_IDLE_TIME_OP
Scheduler, 471
l4_scheduler_info
Scheduler, 472
L4_SCHEDULER_INFO_OP
Scheduler, 471
l4_scheduler_is_online
Scheduler, 473
L4_scheduler_ops
Scheduler, 470
l4_scheduler_run_thread
Scheduler, 473
L4_SCHEDULER_RUN_THREAD_OP
Scheduler, 471
l4_semaphore_down
Kernel-provided semaphore, 330
l4_semaphore_up
Kernel-provided semaphore, 330
l4_sleep
util.h, 1801, 1804
Utility Functions, 542
l4_snd_fpage_t, 1274
l4_snd_timeout
Timeouts, 528
l4_sndfpage_add
Object Invocation, 444
L4_SUPERPAGEMASK
Memory related, 407
L4_SUPERPAGESIZE
Memory related, 407
L4_sys_segment
segment.h, 1810
l4_syscall_flags_t
Object Invocation, 434
L4_SYSF_CALL
Object Invocation, 434
L4_SYSF_NONE
Object Invocation, 434
L4_SYSF_OPEN_WAIT
Object Invocation, 434
L4_SYSF_RECV

Object Invocation, 434
L4_SYSF_REPLY
Object Invocation, 434
L4_SYSF_REPLY_AND_WAIT
Object Invocation, 434
L4_SYSF_SEND
Object Invocation, 434
L4_SYSF_SEND_AND_WAIT
Object Invocation, 434
L4_SYSF_WAIT
Object Invocation, 434
l4_task_add_ku_mem
Task, 496
L4_TASK_ADD_KU_MEM_OP
L4 IPC Opcodes, 335
l4_task_cap_equal
Task, 496
L4_TASK_CAP_INFO_OP
L4 IPC Opcodes, 335
l4_task_cap_valid
Task, 497
l4_task_delete_obj
Task, 497
L4_TASK_LDT_SET_X86_OP
L4 IPC Opcodes, 335
L4_task_ldt_x86_consts
segment.h, 1810, 1817
L4_TASK_LDT_X86_ENTRY_SIZE
segment.h, 1811, 1817
L4_TASK_LDT_X86_MAX_ENTRIES
segment.h, 1811, 1817
l4_task_map
Task, 498
L4_TASK_MAP_OP
L4 IPC Opcodes, 335
L4_TASK_MAP_VGICC_ARM_OP
L4 IPC Opcodes, 335
L4_task_ops
L4 IPC Opcodes, 335
l4_task_release_cap
Task, 499
l4_task_unmap
Task, 499
l4_task_unmap_batch
Task, 500
L4_TASK_UNMAP_OP
L4 IPC Opcodes, 335
l4_task_vgicc_map
Task, 501
L4_THREAD_AMD64_GET_SEGMENT_INFO_OP
L4 IPC Opcodes, 336
L4_THREAD_AMD64_SET_SEGMENT_BASE_OP
L4 IPC Opcodes, 336
l4_thread_arm_set_tpidruro
Thread, 505
L4_THREAD_ARM_TPIDRURO_OP
L4 IPC Opcodes, 336
L4_THREAD_CONTROL_ALIEN
Thread, 504
l4_thread_control_alien
Thread control, 521
l4_thread_control_bind
Thread control, 522
L4_THREAD_CONTROL_BIND_TASK
Thread, 504
l4_thread_control_commit
Thread control, 523
l4_thread_control_exc_handler
Thread control, 523
L4_thread_control_flags
Thread, 504
L4_THREAD_CONTROL_MR_IDX_BIND_TASK
Thread, 505
L4_THREAD_CONTROL_MR_IDX_BIND_UTCB
Thread, 505
L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER
Thread, 505
L4_THREAD_CONTROL_MR_IDX_FLAG_VALS
Thread, 505
L4_THREAD_CONTROL_MR_IDX_FLAGS
Thread, 505
L4_THREAD_CONTROL_MR_IDX_PAGER
Thread, 505
L4_thread_control_mr_indices
Thread, 505
L4_THREAD_CONTROL_OP
L4 IPC Opcodes, 336
l4_thread_control_pager
Thread control, 524
L4_THREAD_CONTROL_SET_EXC_HANDLER
Thread, 504
L4_THREAD_CONTROL_SET_PAGER
Thread, 504
l4_thread_control_start
Thread control, 524
l4_thread_control_ux_host_syscall
Thread control, 525
L4_THREAD_CONTROL_UX_NATIVE
Thread, 504
l4_thread_ex_regs
Thread, 507
L4_THREAD_EX_REGS_CANCEL
Thread, 505
L4_thread_ex_regs_flags
Thread, 505
L4_THREAD_EX_REGS_OP
L4 IPC Opcodes, 336
l4_thread_ex_regs_ret
Thread, 508
l4_thread_ex_regs_ret_u
Thread, 509
L4_THREAD_EX_REGS_TRIGGER_EXCEPTION
Thread, 505
l4_thread_ex_regs_u
Thread, 510
l4_thread_modify_sender_add

Thread, 511
l4_thread_modify_sender_commit
Thread, 512
L4_THREAD_MODIFY_SENDER_OP
L4 IPC Opcodes, 336
l4_thread_modify_sender_start
Thread, 512
L4_THREAD_OPCODE_MASK
L4 IPC Opcodes, 336
L4_thread_ops
L4 IPC Opcodes, 335
l4_thread_register_del_irq
Thread, 512
L4_THREAD_REGISTER_DELETE_IRQ_OP
L4 IPC Opcodes, 336
l4_thread_regs_t, 1275
L4_THREAD_STATS_OP
L4 IPC Opcodes, 336
l4_thread_stats_time
Thread, 513
l4_thread_switch
Thread, 514
L4_THREAD_SWITCH_OP
L4 IPC Opcodes, 336
l4_thread_vcpu_control
Thread, 514
l4_thread_vcpu_control_ext
Thread, 515
l4_thread_vcpu_control_ext_u
Thread, 516
L4_THREAD_VCPU_CONTROL_OP
L4 IPC Opcodes, 336
l4_thread_vcpu_control_u
Thread, 517
l4_thread_vcpu_resume_commit
Thread, 518
L4_THREAD_VCPU_RESUME_OP
L4 IPC Opcodes, 336
l4_thread_vcpu_resume_start
Thread, 519
L4_THREAD_X86_GDT_OP
L4 IPC Opcodes, 336
l4_thread_yield
Thread, 519
l4_timeout
Timeouts, 528
l4_timeout_abs
Timeouts, 529
l4_timeout_get
Timeouts, 530
l4_timeout_is_absolute
Timeouts, 531
l4_timeout_rel
Timeouts, 531
l4_timeout_rel_get
Timeouts, 532
l4_timeout_s, 1276
Timeouts, 527
l4_timeout_t, 1277
Timeouts, 527
l4_touch_ro
Utility Functions, 544
l4_touch_rw
Utility Functions, 544
l4_trunc_page
Memory related, 410
l4_trunc_size
Memory related, 411
l4_tsc_init
Timestamp Counter, 539
l4_tsc_to_ns
Timestamp Counter, 539
l4_tsc_to_s_and_ns
Timestamp Counter, 540
l4_tsc_to_us
Timestamp Counter, 540
L4_TYPE_VHW_FRAMEBUFFER
Fiasco-UX Virtual devices, 237
L4_TYPE_VHW_INPUT
Fiasco-UX Virtual devices, 237
L4_TYPE_VHW_NET
Fiasco-UX Virtual devices, 237
L4_TYPE_VHW_NONE
Fiasco-UX Virtual devices, 237
L4_TYPES_FLAGS_OPS_DEF
types, 2169
l4_unmap_flags_t
Task, 495
l4_usleep
Utility Functions, 545
l4_utcb_br
Virtual Registers (UTCBs), 590
L4_UTCB_BUF_REGS_OFFSET
x86 Virtual Registers (UTCB), 605
L4_utcb_consts_x86
x86 Virtual Registers (UTCB), 604
l4_utcb_exc
Exception registers, 222
l4_utcb_exc_is_ex_regs_exception
Exception registers, 222
l4_utcb_exc_is_pf
Exception registers, 223
l4_utcb_exc_pc
Exception registers, 223
l4_utcb_exc_pc_set
Exception registers, 224
L4_UTCB_EXCEPTION_REGS_SIZE
x86 Virtual Registers (UTCB), 605
L4_UTCB_GENERIC_BUFFERS_SIZE
x86 Virtual Registers (UTCB), 605
L4_UTCB_GENERIC_DATA_SIZE
x86 Virtual Registers (UTCB), 605
L4_UTCB_INHERIT_FPU
x86 Virtual Registers (UTCB), 605
l4_utcb_mr
Virtual Registers (UTCBs), 590

l4_utcb_mr64_idx
Timeouts, 532
L4_UTCB_MSG_REGS_OFFSET
x86 Virtual Registers (UTCB), 605
L4_UTCB_OFFSET
x86 Virtual Registers (UTCB), 605
l4_utcb_t
Virtual Registers (UTCBs), 590
l4_utcb_tcr
Virtual Registers (UTCBs), 591
L4_UTCB_THREAD_REGS_OFFSET
x86 Virtual Registers (UTCB), 605
l4_vcon_attr_t, 1278
set_raw, 1279
L4_VCON_ECHO
Virtual Console, 576
l4_vcon_get_attr
Virtual Console, 577
L4_VCON_GET_ATTR_OP
L4 IPC Opcodes, 336
l4_vcon_get_attr_u
Virtual Console, 578
L4_vcon_i_flags
Virtual Console, 576
L4_VCON_ICANON
Virtual Console, 576
L4_VCON_ICRNL
Virtual Console, 576
L4_VCON_IGNCR
Virtual Console, 576
L4_VCON_INLCR
Virtual Console, 576
L4_vcon_l_flags
Virtual Console, 576
L4_vcon_o_flags
Virtual Console, 576
L4_VCON_OCRNL
Virtual Console, 576
L4_VCON_ONLCR
Virtual Console, 576
L4_VCON_ONLRET
Virtual Console, 576
L4_vcon_ops
L4 IPC Opcodes, 336
l4_vcon_read
Virtual Console, 579
L4_vcon_read_flags
vcon.h, 2299
L4_VCON_READ_OP
L4 IPC Opcodes, 336
L4_VCON_READ_SIZE
Virtual Console, 577
L4_VCON_READ_SIZE_MASK
vcon.h, 2299
L4_VCON_READ_STAT_BREAK
vcon.h, 2299
L4_VCON_READ_STAT_DONE
vcon.h, 2299
l4_vcon_read_u
Virtual Console, 579
l4_vcon_read_with_flags
Virtual Console, 580
l4_vcon_send
Virtual Console, 581
l4_vcon_send_u
Virtual Console, 582
l4_vcon_set_attr
Virtual Console, 583
L4_VCON_SET_ATTR_OP
L4 IPC Opcodes, 336
l4_vcon_set_attr_raw
Virtual Console, 584
l4_vcon_set_attr_u
Virtual Console, 584
L4_vcon_size_consts
Virtual Console, 577
l4_vcon_write
Virtual Console, 585
L4_VCON_WRITE_OP
L4 IPC Opcodes, 336
L4_VCON_WRITE_SIZE
Virtual Console, 577
l4_vcon_write_u
Virtual Console, 586
L4_VCPU_F_EXCEPTIONS
vCPU API, 595
L4_VCPU_F_FPU_ENABLED
vCPU API, 595
L4_VCPU_F_IRQ
vCPU API, 595
L4_VCPU_F_PAGE_FAULTS
vCPU API, 595
L4_VCPU_F_USER_MODE
vCPU API, 595
l4_vcpu_ipc_regs_t, 1280
L4_VCPU_OFFSET_EXT_INFOS
vCPU API, 596
L4_VCPU_OFFSET_EXT_STATE
vCPU API, 596
l4_vcpu_regs_t, 1281
ax, 1282
bp, 1283
bx, 1283
cx, 1283
di, 1283
dx, 1284
si, 1284
L4_VCPU_SF_IRQ_PENDING
vCPU API, 596
L4_vcpu_state_flags
vCPU API, 594
L4_vcpu_state_offset
vCPU API, 595
l4_vcpu_state_t, 1285
L4_vcpu_sticky_flags
vCPU API, 596

l4_vhw_descriptor, 1287
l4_vhw_entry, 1289
l4_vhw_entry_type
Fiasco-UX Virtual devices, 237
L4_virtio_cmd
L4 VIRTIO Transport Layer, 342
L4_virtio_irq_status
L4 VIRTIO Transport Layer, 342
L4_virtio_opcodes
L4 VIRTIO Transport Layer, 343
l4_vm_svm_vmcb_control_area, 1290
l4_vm_svm_vmcb_state_save_area, 1291
l4_vm_svm_vmcb_state_save_area_seg, 1292
l4_vm_svm_vmcb_t, 1293
l4_vm_tz_state, 1294
L4_VM_VMX_BASIC_REG
VM API for VMX, 552
L4_vm_vmx_caps_regs
VM API for VMX, 551
l4_vm_vmx_clear
VM API for VMX, 552
L4_VM_VMX_CR0_FIXED0_REG
VM API for VMX, 552
L4_VM_VMX_CR0_FIXED1_REG
VM API for VMX, 552
L4_VM_VMX_CR4_FIXED0_REG
VM API for VMX, 552
L4_VM_VMX_CR4_FIXED1_REG
VM API for VMX, 552
L4_vm_vmx_dfl1_regs
VM API for VMX, 552
L4_VM_VMX_ENTRY_CTLS_DFL1_REG
VM API for VMX, 552
L4_VM_VMX_EPT_VPID_CAP_REG
VM API for VMX, 552
L4_VM_VMX_EXIT_CTLS_DFL1_REG
VM API for VMX, 552
l4_vm_vmx_field_len
VM API for VMX, 553
l4_vm_vmx_field_order
VM API for VMX, 554
l4_vm_vmx_get_caps
VM API for VMX, 554
l4_vm_vmx_get_caps_default1
VM API for VMX, 555
l4_vm_vmx_get_cr2_index
VM API for VMX, 555
L4_VM_VMX_MISC_REG
VM API for VMX, 552
L4_VM_VMX_NUM_CAPS_REGS
VM API for VMX, 552
L4_VM_VMX_NUM_DFL1_REGS
VM API for VMX, 552
L4_VM_VMX_PINBASED_CTLS_DFL1_REG
VM API for VMX, 552
L4_VM_VMX_PROCBASED_CTLS2_REG
VM API for VMX, 552
L4_VM_VMX_PROCBASED_CTLS_DFL1_REG
VM API for VMX, 552
l4_vm_vmx_ptr_load
VM API for VMX, 556
l4_vm_vmx_read
VM API for VMX, 556
l4_vm_vmx_read_16
VM API for VMX, 557
l4_vm_vmx_read_32
VM API for VMX, 558
l4_vm_vmx_read_64
VM API for VMX, 558
l4_vm_vmx_read_nat
VM API for VMX, 559
L4_VM_VMX_TRUE_ENTRY_CTLS_REG
VM API for VMX, 552
L4_VM_VMX_TRUE_EXIT_CTLS_REG
VM API for VMX, 552
L4_VM_VMX_TRUE_PINBASED_CTLS_REG
VM API for VMX, 552
L4_VM_VMX_TRUE_PROCBASED_CTLS_REG
VM API for VMX, 552
L4_VM_VMX_VMCS_CR2
VM API for VMX, 551
L4_VM_VMX_VMCS_ENUM_REG
VM API for VMX, 552
L4_VM_VMX_VMCS_MSR_CSTAR
VM API for VMX, 551
L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE
VM API for VMX, 551
L4_VM_VMX_VMCS_MSR_LSTAR
VM API for VMX, 551
L4_VM_VMX_VMCS_MSR_STAR
VM API for VMX, 551
L4_VM_VMX_VMCS_MSR_SYSCALL_MASK
VM API for VMX, 551
L4_VM_VMX_VMCS_MSR_TSC_AUX
VM API for VMX, 551
L4_VM_VMX_VMCS_XCR0
VM API for VMX, 551
l4_vm_vmx_write
VM API for VMX, 560
l4_vm_vmx_write_16
VM API for VMX, 561
l4_vm_vmx_write_32
VM API for VMX, 562
l4_vm_vmx_write_64
VM API for VMX, 562
l4_vm_vmx_write_nat
VM API for VMX, 563
L4_WHOLE_ADDRESS_SPACE
Flex pages, 240
L4_WHOLE_IOADDRESS_SPACE
Flex pages, 240
L4IO_DEVICE_ANY
IO interface, 263
L4IO_DEVICE_INVALID
IO interface, 263
L4IO_DEVICE_OTHER

IO interface, 263
L4IO_DEVICE_PCI
IO interface, 263
l4io_device_types_t
IO interface, 262
L4IO_DEVICE_USB
IO interface, 263
l4io_get_root_device
io.h, 1855
l4io_has_resource
IO interface, 264
l4io_iomem_flags_t
IO interface, 263
l4io_iterate_devices
io.h, 1855
l4io_lookup_device
IO interface, 264
l4io_lookup_resource
IO interface, 264
L4IO_MEM_CACHED
IO interface, 263
L4IO_MEM_EAGER_MAP
IO interface, 263
L4IO_MEM_NONCACHED
IO interface, 263
L4IO_MEM_USE_MTRR
IO interface, 263
L4IO_MEM_USE_RESERVED_AREA
IO interface, 263
l4io_release_iomem
IO interface, 265
l4io_release_ioport
IO interface, 265
l4io_request_all_ioports
io.h, 1856
l4io_request_icu
io.h, 1856
l4io_request_iomem
IO interface, 266
l4io_request_iomem_region
IO interface, 266
l4io_request_ioport
IO interface, 267
l4io_request_resource_iomem
IO interface, 268
L4IO_RESOURCE_ANY
IO interface, 263
L4IO_RESOURCE_INVALID
IO interface, 263
L4IO_RESOURCE_IRQ
IO interface, 263
L4IO_RESOURCE_MEM
IO interface, 263
L4IO_RESOURCE_PORT
IO interface, 263
l4io_resource_t
IO interface, 262
l4io_resource_types_t
IO interface, 263
l4irq_attach
Interface using direct functionality., 297
l4irq_attach_cap
Interface using direct functionality., 294
l4irq_attach_cap_ft
Interface using direct functionality., 295
l4irq_attach_ft
Interface using direct functionality., 297
l4irq_attach_thread
Interface using direct functionality., 298
l4irq_attach_thread_cap
Interface using direct functionality., 295
l4irq_attach_thread_cap_ft
Interface using direct functionality., 296
l4irq_attach_thread_ft
Interface using direct functionality., 298
l4irq_detach
Interface using direct functionality., 299
l4irq_release
Interface for asynchronous ISR handlers., 293
l4irq_request
Interface for asynchronous ISR handlers., 293
l4irq_request_cap
Interface for asynchronous ISR handlers with a given IRQ capability., 291
l4irq_unmask
Interface using direct functionality., 299
l4irq_unmask_and_wait_any
Interface using direct functionality., 300
l4irq_wait
Interface using direct functionality., 300
l4irq_wait_any
Interface using direct functionality., 300
l4la_alloc
list_alloc.h, 2355
l4la_avail
list_alloc.h, 2355
l4la_dump
list_alloc.h, 2355
l4la_free
list_alloc.h, 2355
l4la_init
list_alloc.h, 2356
L4Re, 636
chkcap, 641
chkipc, 642
chksys, 643, 645, 646
make_shared_cap, 647
make_shared_del_cap, 648
make_unique_cap, 649
make_unique_del_cap, 649
Shared_cap, 638
shared_cap, 638
Shared_del_cap, 639
shared_del_cap, 639
throw_error, 650
Unique_cap, 640

unique_cap, 640
Unique_del_cap, 640
unique_del_cap, 641
L4Re C Interface, 361
l4re_inhibitor_acquire, 363
L4Re C++ Interface, 364
L4Re Capability API, 366
cap_alloc, 368
make_ref_cap, 367
make_ref_del_cap, 367
L4Re ELF Auxiliary Information, 369
L4RE_ELF_AUX_ELEM, 370
L4RE_ELF_AUX_ELEM_T, 370
L4RE_ELF_AUX_T_KIP_ADDR, 371
L4RE_ELF_AUX_T_NONE, 371
L4RE_ELF_AUX_T_STACK_ADDR, 371
L4RE_ELF_AUX_T_STACK_SIZE, 371
L4RE_ELF_AUX_T_VMA, 371
L4Re Protocol identifiers, 371
L4Re Util C Interface, 372
L4Re Util C++ Interface, 372
L4Re::Cap_alloc, 1295
alloc, 1296
free, 1297
get_cap_alloc, 1298
L4Re::Console, 1299
L4Re::Dataspace, 1301
allocate, 1303
clear, 1304
copy_in, 1304
flags, 1306
info, 1306
map, 1307
map_region, 1308
size, 1309
L4Re::Dataspace::F, 1309
Bufferable, 1310
Cacheable, 1310
Caching_mask, 1310
Caching_shift, 1310
Flags, 1310
Normal, 1310
R, 1310
Rights_mask, 1310
Ro, 1310
RW, 1310
Uncacheable, 1310
W, 1310
L4Re::Dataspace::Stats, 1311
L4Re::Debug_obj, 1311
debug, 1314
L4Re::Dma_space, 1314
associate, 1317
Attribute, 1316
Attributes, 1315
Bidirectional, 1316
Coherent, 1317
Direction, 1316
disassociate, 1317
From_device, 1316
map, 1318
No_sync, 1316
None, 1316
Phys_space, 1317
Space_attrib, 1317
To_device, 1316
unmap, 1318
L4Re::Env, 1319
env, 1322
factory, 1322, 1323
first_free_cap, 1323
first_free_utcb, 1324
get, 1324
get_cap, 1325
initial_caps, 1326
log, 1327
main_thread, 1327, 1328
mem_alloc, 1328
parent, 1329
rm, 1329, 1330
scheduler, 1330
task, 1331
utcb_area, 1331
L4Re::Event, 1332
get_buffer, 1335
L4Re::Event_buffer_t< PAYLOAD >, 1335
Event_buffer_t, 1337
next, 1337
put, 1337
L4Re::Event_buffer_t< PAYLOAD >::Event, 1338
L4Re::Inhibitor, 1339
acquire, 1342
Name_max, 1342
next_lock_info, 1343
release, 1344
L4Re::Log, 1344
print, 1347
printn, 1347
L4Re::Mem_alloc, 1347
alloc, 1350
Continuous, 1350
Mem_alloc_flags, 1350
Pinned, 1350
Super_pages, 1350
L4Re::Mmio_space, 1352
Access_width, 1354
mmio_read, 1354
mmio_write, 1355
Wd_16bit, 1354
Wd_32bit, 1354
Wd_64bit, 1354
Wd_8bit, 1354
L4Re::Namespace, 1356
Link, 1359
Overwrite, 1359
Partly_resolved, 1358

query, 1359, 1360
Query_result_flags, 1358
Register_flags, 1358
register_obj, 1361
Ro, 1359
Rs, 1359
Rw, 1359
Rws, 1359
Strong, 1359
Trusted, 1359
unlink, 1361
L4Re::Ned::Cmd_control, 1362
execute, 1363
L4Re::Parent, 1364
signal, 1367
L4Re::Random, 1367
get_random, 1370
L4Re::Rm, 1371
Area, 1374
attach, 1375, 1376
Caching_shift, 1375
detach, 1377–1379
Detach_again, 1375
Detach_exact, 1375
Detach_flags, 1374
Detach_keep, 1375
Detach_overlap, 1375
Detach_result, 1375
Detached_ds, 1375
find, 1379
free_area, 1380
get_areas, 1381
get_regions, 1381
Kept_ds, 1375
Region, 1374
Region_flag_shifts, 1375
reserve_area, 1382, 1383
Split_ds, 1375
L4Re::Rm::F, 1384
Attach_flags, 1384
Attach_mask, 1385
Cache_buffered, 1385
Cache_normal, 1385
Cache_uncached, 1385
Caching_mask, 1385
Detach_free, 1385
Ds_map_mask, 1385
Eager_map, 1385
In_area, 1385
Pager, 1385
R, 1385
Region_flags, 1385
Region_flags_mask, 1385
Reserved, 1385
Rights_mask, 1385
RW, 1385
RWX, 1385
RX, 1385
Search_addr, 1385
W, 1385
X, 1385
L4Re::Rm::Range, 1385
L4Re::Smart_cap_auto< Unmap_flags >, 1386
L4Re::Smart_count_cap< Unmap_flags >, 1387
L4Re::Util, 651
make_shared_cap, 660
make_shared_del_cap, 660
make_unique_cap, 661
make_unique_del_cap, 661
Shared_cap, 653
shared_cap, 655
Shared_del_cap, 656
shared_del_cap, 656
Unique_cap, 657
unique_cap, 658
Unique_del_cap, 658
unique_del_cap, 659
L4Re::Util::Br_manager, 1388
alloc_buffer_demand, 1391
get_rcv_cap, 1392
realloc_rcv_cap, 1392
set_rcv_cap_flags, 1393
L4Re::Util::Br_manager_hooks, 1394
L4Re::Util::Br_manager_timeout_hooks, 1395
L4Re::Util::Cap_alloc_base, 1398
L4Re::Util::Counter< COUNTER >, 1399
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, 1400
alloc, 1402, 1404
Counting_cap_alloc, 1402
free, 1404
release, 1405
setup, 1407
take, 1407
L4Re::Util::Dataspace_svr, 1408
allocate, 1409
clear, 1410
copy, 1410
is_static, 1411
map, 1411
map_hook, 1412
page_shift, 1412
release, 1413
take, 1413
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, 1414
foreach_available_event, 1416
process, 1416
L4Re::Util::Event_buffer_t< PAYLOAD >, 1417
attach, 1419
buf, 1420
detach, 1420
L4Re::Util::Event_svr< SVR >, 1421
L4Re::Util::Event_t< PAYLOAD >, 1423
buffer, 1424
init, 1425

init_poll, 1426
irq, 1427
Mode, 1424
Mode_irq, 1424
Mode_polling, 1424
L4Re::Util::Item_alloc_base, 1427
L4Re::Util::Names::Name, 1428
L4Re::Util::Names::Name_space, 1429
alloc_dynamic_entry, 1431
copy_receive_cap, 1431
free_capability, 1431
free_dynamic_entry, 1432
free_epiface, 1432
get_epiface, 1432
L4Re::Util::Object_registry, 1433
Object_registry, 1435
register_irq_obj, 1436
register_obj, 1436, 1437
unregister_obj, 1438
L4Re::Util::Ref_cap< T >, 1439
L4Re::Util::Ref_del_cap< T >, 1440
L4Re::Util::Registry_server< LOOP_HOOKS >, 1441
loop, 1445
Registry_server, 1444, 1445
L4Re::Util::Smart_cap_auto< Unmap_flags >, 1446
L4Re::Util::Smart_count_cap< Unmap_flags >, 1447
L4Re::Util::Vcon_svr< SVR >, 1448
L4Re::Util::Video::Goos_svr, 1449
get_fb, 1450
init_infos, 1450
refresh, 1451
screen_info, 1451
view_info, 1452
L4Re::Vfs, 662
L4Re::Vfs::Be_file, 1452
data_space, 1456
fstat64, 1456
unlock_all_locks, 1456
L4Re::Vfs::Be_file_system, 1457
~Be_file_system, 1459
Be_file_system, 1458
type, 1459
L4Re::Vfs::Directory, 1460
faccessat, 1462
link, 1462
mkdir, 1462
rename, 1463
rmdir, 1463
symlink, 1464
unlink, 1464
L4Re::Vfs::File, 1465
L4Re::Vfs::File_system, 1467
mount, 1468
type, 1469
L4Re::Vfs::Fs, 1469
alloc_fd, 1472
free_fd, 1472
get_file, 1472
mount, 1473
set_fd, 1473
L4Re::Vfs::Generic_file, 1474
fchmod, 1476
fstat64, 1477
get_status_flags, 1477
set_status_flags, 1477
unlock_all_locks, 1478
L4Re::Vfs::Mman, 1479
L4Re::Vfs::Ops, 1480
L4Re::Vfs::Regular_file, 1483
data_space, 1485
fdatasync, 1485
fsync, 1485
ftruncate64, 1485
get_lock, 1486
lseek64, 1486
readv, 1486
set_lock, 1487
writev, 1487
L4Re::Vfs::Special_file, 1488
ioctl, 1490
L4Re::Video::Color_component, 1491
Color_component, 1492
dump, 1492
get, 1492
operator==, 1493
set, 1493
shift, 1493
size, 1494
L4Re::Video::Goos, 1495
create_buffer, 1499
create_view, 1499
delete_buffer, 1500
delete_view, 1500
F_auto_refresh, 1499
F_dynamic_buffers, 1499
F_dynamic_views, 1499
F_pointer, 1499
Flags, 1498
get_static_buffer, 1500
info, 1501
view, 1501
L4Re::Video::Goos::Info, 1502
L4Re::Video::Pixel_info, 1503
a, 1506
b, 1507
bits_per_pixel, 1507
bytes_per_pixel, 1508
dump, 1508
g, 1509
has_alpha, 1510
operator==, 1510
padding, 1511
Pixel_info, 1505, 1506
r, 1511
L4Re::Video::View, 1512
F_above, 1514

F_dyn_allocated, 1514
F_flags_mask, 1514
F_fully_dynamic, 1514
F_none, 1513
F_set_background, 1514
F_set_buffer, 1513
F_set_buffer_offset, 1513
F_set_bytes_per_line, 1513
F_set_flags, 1514
F_set_pixel, 1514
F_set_position, 1514
Flags, 1513
info, 1514
refresh, 1515
set_info, 1515
set_viewport, 1516
stack, 1516
V_flags, 1514
L4Re::Video::View::Info, 1517
l4re_aux_t, 1520
l4re_debug_obj_debug
Debug interface, 180
l4re_dma_space_associate
DMA Space Interface, 173
L4RE_DMA_SPACE_BIDIRECTIONAL
dma_space.h, 1962
L4RE_DMA_SPACE_COHERENT
dma_space.h, 1963
l4re_dma_space_direction
dma_space.h, 1962
l4re_dma_space_disassociate
DMA Space Interface, 174
L4RE_DMA_SPACE_FROM_DEVICE
dma_space.h, 1962
l4re_dma_space_map
DMA Space Interface, 174
L4RE_DMA_SPACE_NONE
dma_space.h, 1962
L4RE_DMA_SPACE_PHYS_SPACE
dma_space.h, 1963
l4re_dma_space_space_attribs
dma_space.h, 1962
l4re_dma_space_t
DMA Space Interface, 173
L4RE_DMA_SPACE_TO_DEVICE
dma_space.h, 1962
l4re_dma_space_unmap
DMA Space Interface, 175
l4re_ds_allocate
Dataspace interface, 178
l4re_ds_clear
Dataspace interface, 178
l4re_ds_copy_in
Dataspace interface, 178
L4RE_DS_F_BUFFERABLE
Dataspace interface, 177
L4RE_DS_F_CACHEABLE
Dataspace interface, 177
L4RE_DS_F_CACHING_MASK
Dataspace interface, 177
L4RE_DS_F_CACHING_SHIFT
Dataspace interface, 177
L4RE_DS_F_NORMAL
Dataspace interface, 177
L4RE_DS_F_UNCACHEABLE
Dataspace interface, 177
l4re_ds_flags
Dataspace interface, 178
l4re_ds_info
Dataspace interface, 179
l4re_ds_map_flags
Dataspace interface, 177
l4re_ds_size
Dataspace interface, 179
l4re_ds_stats_t, 1521
L4RE_ELF_AUX_ELEM
L4Re ELF Auxiliary Information, 370
L4RE_ELF_AUX_ELEM_T
L4Re ELF Auxiliary Information, 370
l4re_elf_aux_mword_t, 1521
l4re_elf_aux_t, 1522
L4RE_ELF_AUX_T_KIP_ADDR
L4Re ELF Auxiliary Information, 371
L4RE_ELF_AUX_T_NONE
L4Re ELF Auxiliary Information, 371
L4RE_ELF_AUX_T_STACK_ADDR
L4Re ELF Auxiliary Information, 371
L4RE_ELF_AUX_T_STACK_SIZE
L4Re ELF Auxiliary Information, 371
L4RE_ELF_AUX_T_VMA
L4Re ELF Auxiliary Information, 371
l4re_elf_aux_vma_t, 1523
l4re_env
Initial Environment, 285
l4re_env_cap_entry_t, 1524
flags, 1525
l4re_env_cap_entry_t, 1524
l4re_env_get_cap
Initial Environment, 286
l4re_env_get_cap_e
Initial Environment, 287
l4re_env_get_cap_l
Initial Environment, 288
l4re_env_t, 1526
env.h, 2012
l4re_event_get_axis_info
Event interface, 219
l4re_event_get_buffer
Event interface, 219
l4re_event_get_num_streams
Event interface, 220
l4re_event_get_stream_info
Event interface, 220
l4re_event_get_stream_info_for_id
Event interface, 221
l4re_event_t, 1527

l4re_inhibitor_acquire
L4Re C Interface, 363
l4re_kip
Initial Environment, 288
l4re_log_print
Log interface, 390
l4re_log_print_srv
Log interface, 390
l4re_log_printn
Log interface, 391
l4re_log_printn_srv
Log interface, 392
l4re_ma_alloc
Memory allocator, 395
l4re_ma_alloc_align
Memory allocator, 396
l4re_ma_alloc_align_srv
Memory allocator, 397
l4re_ma_flags
Memory allocator, 394
l4re_ns_query_srv
Namespace interface, 429
l4re_ns_query_to_srv
Namespace interface, 430
l4re_ns_register_flags
Namespace interface, 429
l4re_ns_register_obj_srv
Namespace interface, 431
L4RE_PROTO_DATASPACE
protocols.h, 2045
L4RE_PROTO_DEBUG
protocols.h, 2045
L4RE_PROTO_DMA_SPACE
protocols.h, 2045
L4RE_PROTO_EVENT
protocols.h, 2045
L4RE_PROTO_GOOS
protocols.h, 2045
L4RE_PROTO_INHIBITOR
protocols.h, 2045
L4RE_PROTO_MMIO_SPACE
protocols.h, 2045
L4RE_PROTO_NAMESPACE
protocols.h, 2045
L4RE_PROTO_PARENT
protocols.h, 2045
L4RE_PROTO_RM
protocols.h, 2045
L4RE_PROTO_RSVD_1
protocols.h, 2045
L4re_protocols
protocols.h, 2045
l4re_rm_attach
Region map interface, 456
l4re_rm_attach_srv
Region map interface, 458
L4RE_RM_CACHING_SHIFT
Region map interface, 456
l4re_rm_detach
Region map interface, 458
l4re_rm_detach_ds
Region map interface, 459
l4re_rm_detach_ds_unmap
Region map interface, 460
l4re_rm_detach_srv
Region map interface, 461
l4re_rm_detach_unmap
Region map interface, 462
L4RE_RM_F_ATTACH_FLAGS
Region map interface, 456
L4RE_RM_F_CACHE_BUFFERED
Region map interface, 456
L4RE_RM_F_CACHE_NORMAL
Region map interface, 456
L4RE_RM_F_CACHE_UNCACHED
Region map interface, 456
L4RE_RM_F_CACHING
Region map interface, 456
L4RE_RM_F_EAGER_MAP
Region map interface, 456
L4RE_RM_F_IN_AREA
Region map interface, 456
L4RE_RM_F_NO_ALIAS
Region map interface, 456
L4RE_RM_F_PAGER
Region map interface, 456
L4RE_RM_F_R
Region map interface, 456
L4RE_RM_F_RESERVED
Region map interface, 456
L4RE_RM_F_SEARCH_ADDR
Region map interface, 456
l4re_rm_find
Region map interface, 463
l4re_rm_find_srv
Region map interface, 464
l4re_rm_flags_values
Region map interface, 456
l4re_rm_free_area
Region map interface, 465
l4re_rm_free_area_srv
Region map interface, 466
L4RE_RM_REGION_FLAGS
Region map interface, 456
l4re_rm_reserve_area
Region map interface, 466
l4re_rm_reserve_area_srv
Region map interface, 467
l4re_rm_show_lists
Region map interface, 468
l4re_util_cap_last
Capability allocator, 157
l4re_util_kumem_alloc
kumem_alloc.h, 1980
l4re_video_color_component_t, 1528
l4re_video_goos_create_buffer

Video API, 567
l4re_video_goos_create_view
Video API, 568
l4re_video_goos_delete_buffer
Video API, 568
l4re_video_goos_delete_view
Video API, 570
l4re_video_goos_get_static_buffer
Video API, 570
l4re_video_goos_get_view
Video API, 570
l4re_video_goos_info
Video API, 571
l4re_video_goos_info_flags_t
Video API, 567
l4re_video_goos_info_t, 1529
l4re_video_goos_refresh
Video API, 571
l4re_video_pixel_info_t, 1530
l4re_video_view_get_info
Video API, 572
l4re_video_view_info_flags_t
Video API, 567
l4re_video_view_info_t, 1531
l4re_video_view_refresh
Video API, 572
l4re_video_view_set_info
Video API, 572
l4re_video_view_set_viewport
Video API, 573
l4re_video_view_stack
Video API, 573
l4re_video_view_t, 1533
Video API, 566
l4shmc_add_chunk
Chunks, 158
l4shmc_add_signal
Signals, 488
l4shmc_area_overhead
Shared Memory Library, 477
l4shmc_area_size
Shared Memory Library, 477
l4shmc_area_size_free
Shared Memory Library, 478
l4shmc_attach
Shared Memory Library, 478
l4shmc_attach_signal
Signals, 488
l4shmc_attach_signal_to
Signals, 489
l4shmc_attach_to
Shared Memory Library, 479
l4shmc_check_magic
Signals, 489
l4shmc_chunk_capacity
Chunks, 159
l4shmc_chunk_consumed
Consumer, 165
l4shmc_chunk_overhead
Shared Memory Library, 479
l4shmc_chunk_ptr
Chunks, 159
l4shmc_chunk_ready
Producer, 449
l4shmc_chunk_ready_sig
Producer, 450
l4shmc_chunk_signal
Chunks, 160
l4shmc_chunk_size
Consumer, 166
l4shmc_chunk_try_to_take
Producer, 450
l4shmc_connect_chunk_signal
Shared Memory Library, 479
l4shmc_create
Shared Memory Library, 481
l4shmc_enable_chunk
Consumer, 166
l4shmc_enable_signal
Consumer, 169
l4shmc_get_chunk
Chunks, 160
l4shmc_get_chunk_to
Chunks, 160
l4shmc_get_signal_to
Signals, 490
l4shmc_is_chunk_clear
Producer, 451
l4shmc_is_chunk_ready
Consumer, 167
l4shmc_iterate_chunk
Chunks, 161
l4shmc_signal_cap
Signals, 490
l4shmc_trigger
Producer, 451
l4shmc_wait_any
Consumer, 169
l4shmc_wait_any_to
Consumer, 170
l4shmc_wait_any_try
Consumer, 170
l4shmc_wait_chunk
Consumer, 167
l4shmc_wait_chunk_to
Consumer, 167
l4shmc_wait_chunk_try
Consumer, 168
l4shmc_wait_signal
Consumer, 171
l4shmc_wait_signal_to
Consumer, 171
l4shmc_wait_signal_try
Consumer, 171
l4sigma0_debug_dump
Sigma0 API, 483

L4SIGMA0_IPCERROR
Sigma0 API, 483
l4sigma0_map_anypage
Sigma0 API, 483
l4sigma0_map_errstr
Sigma0 API, 484
l4sigma0_map_iomem
Sigma0 API, 484
l4sigma0_map_kip
Sigma0 API, 485
l4sigma0_map_mem
Sigma0 API, 485
l4sigma0_new_client
Sigma0 API, 486
L4SIGMA0_NOFPAGE
Sigma0 API, 483
L4SIGMA0_NOTALIGNED
Sigma0 API, 483
L4SIGMA0_OK
Sigma0 API, 483
l4sigma0_return_flags_t
Sigma0 API, 483
L4SIGMA0_SMALLERFPAGE
Sigma0 API, 483
l4util_add16
Atomic Instructions, 112
l4util_add16_res
Atomic Instructions, 112
l4util_add32
Atomic Instructions, 112
l4util_add32_res
Atomic Instructions, 113
l4util_add8
Atomic Instructions, 113
l4util_add8_res
Atomic Instructions, 113
l4util_and16
Atomic Instructions, 114
l4util_and16_res
Atomic Instructions, 114
l4util_and32
Atomic Instructions, 114
l4util_and32_res
Atomic Instructions, 115
l4util_and8
Atomic Instructions, 115
l4util_and8_res
Atomic Instructions, 115
l4util_atomic_add
Atomic Instructions, 116
l4util_atomic_inc
Atomic Instructions, 116
l4util_backtrace
backtrace.h, 2321
l4util_bsf
Bit Manipulation, 136
l4util_bsr
Bit Manipulation, 137
l4util_btc
Bit Manipulation, 137
l4util_btr
Bit Manipulation, 138
l4util_bts
Bit Manipulation, 139
l4util_clear_bit
Bit Manipulation, 139
l4util_cmpxchg
Atomic Instructions, 116
l4util_cmpxchg16
Atomic Instructions, 117
l4util_cmpxchg32
Atomic Instructions, 118
l4util_cmpxchg64
Atomic Instructions, 118
l4util_cmpxchg8
Atomic Instructions, 119
l4util_complement_bit
Bit Manipulation, 140
l4util_cpu_capabilities
CPU related functions, 147
l4util_cpu_capabilities_nocheck
CPU related functions, 147
l4util_cpu_has_cpuid
CPU related functions, 148
l4util_dec16
Atomic Instructions, 119
l4util_dec16_res
Atomic Instructions, 120
l4util_dec32
Atomic Instructions, 120
l4util_dec32_res
Atomic Instructions, 120
l4util_dec8
Atomic Instructions, 121
l4util_dec8_res
Atomic Instructions, 121
l4util_find_first_set_bit
Bit Manipulation, 140
l4util_find_first_zero_bit
Bit Manipulation, 141
l4util_idt_desc_t, 1534
l4util_idt_header_t, 1535
l4util_in16
IA32 Port I/O API, 256
l4util_in32
IA32 Port I/O API, 256
l4util_in8
IA32 Port I/O API, 257
l4util_inc16
Atomic Instructions, 121
l4util_inc16_res
Atomic Instructions, 121
l4util_inc32
Atomic Instructions, 122
l4util_inc32_res
Atomic Instructions, 122

l4util_inc8
Atomic Instructions, 122
l4util_inc8_res
Atomic Instructions, 123
l4util_ins16
IA32 Port I/O API, 257
l4util_ins32
IA32 Port I/O API, 258
l4util_ins8
IA32 Port I/O API, 258
l4util_ioport_map
port_io.h, 1821
l4util_irq_acknowledge
irq.h, 1934, 1936
l4util_kip_for_each_feature
Kernel Interface Page API, 326
l4util_kip_kernel_abi_version
Kernel Interface Page API, 327
l4util_kip_kernel_has_feature
Kernel Interface Page API, 327
l4util_kip_kernel_is_ux
Kernel Interface Page API, 327
l4util_l4mod_info, 1536
vbe_ctrl_info, 1536
l4util_l4mod_mod, 1537
l4util_mb_addr_range_t, 1538
l4util_mb_apm_t, 1539
l4util_mb_drive_t, 1540
drive_cylinders, 1541
drive_mode, 1541
drive_number, 1541
l4util_mb_for_each_mmap_entry
mb_info.h, 2361
l4util_mb_info_t, 1542
L4UTIL_MB_MEMORY
mb_info.h, 2361
l4util_mb_mod_t, 1543
l4util_mb_vbe_ctrl_t, 1544
l4util_mb_vbe_mode_t, 1545
l4util_micros2l4to
util.h, 1802, 1805
Utility Functions, 546
l4util_next_power2
Bit Manipulation, 141
l4util_or16
Atomic Instructions, 123
l4util_or16_res
Atomic Instructions, 123
l4util_or32
Atomic Instructions, 124
l4util_or32_res
Atomic Instructions, 124
l4util_or8
Atomic Instructions, 124
l4util_or8_res
Atomic Instructions, 125
l4util_out16
IA32 Port I/O API, 258
l4util_out32
IA32 Port I/O API, 259
l4util_out8
IA32 Port I/O API, 259
l4util_outs16
IA32 Port I/O API, 260
l4util_outs32
IA32 Port I/O API, 260
l4util_outs8
IA32 Port I/O API, 260
l4util_rand
Random number support, 452
l4util_set_bit
Bit Manipulation, 142
l4util_splitlog2_hdl
Utility Functions, 546
l4util_splitlog2_size
Utility Functions, 547
l4util_srand
Random number support, 453
l4util_sub16
Atomic Instructions, 125
l4util_sub16_res
Atomic Instructions, 125
l4util_sub32
Atomic Instructions, 126
l4util_sub32_res
Atomic Instructions, 126
l4util_sub8
Atomic Instructions, 127
l4util_sub8_res
Atomic Instructions, 127
l4util_test_bit
Bit Manipulation, 142
l4util_xchg
Atomic Instructions, 127
l4util_xchg16
Atomic Instructions, 128
l4util_xchg32
Atomic Instructions, 128
l4util_xchg8
Atomic Instructions, 128
L4vbus, 663
L4vbus GPIO functions, 373
l4vbus_gpio_config_get, 375
l4vbus_gpio_config_pad, 376
l4vbus_gpio_config_pull, 376
L4vbus_gpio_generic_func, 374
l4vbus_gpio_get, 377
l4vbus_gpio_multi_config_pad, 378
l4vbus_gpio_multi_get, 379
l4vbus_gpio_multi_set, 379
l4vbus_gpio_multi_setup, 380
L4VBUS_GPIO_PIN_PULL_DOWN, 375
L4VBUS_GPIO_PIN_PULL_NONE, 375
L4VBUS_GPIO_PIN_PULL_UP, 375
L4vbus_gpio_pull_modes, 375
l4vbus_gpio_set, 381

l4vbus_gpio_setup, 382
L4VBUS_GPIO_SETUP_INPUT, 374
L4VBUS_GPIO_SETUP_IRQ, 374
L4VBUS_GPIO_SETUP_OUTPUT, 374
l4vbus_gpio_to_irq, 383
L4vbus PCI functions, 384
l4vbus_pci_cfg_read, 384
l4vbus_pci_cfg_write, 385
l4vbus_pci_irq_enable, 386
l4vbus_pcidev_cfg_read, 387
l4vbus_pcidev_cfg_write, 387
l4vbus_pcidev_irq_enable, 388
L4vbus::Device, 1549
_bus, 1557
bus_cap, 1551
dev_handle, 1552
device, 1552
device_by_hid, 1553
get_resource, 1554
is_compatible, 1555
next_device, 1555
operator!=, 1556
operator==, 1556
L4vbus::Gpio_module, 1557
config_pad, 1560
get, 1561
pin, 1561
set, 1562
setup, 1562
L4vbus::Gpio_module::Pin_slice, 1563
L4vbus::Gpio_pin, 1564
config_get, 1567
config_pad, 1568
config_pull, 1568
get, 1569
pin, 1569
set, 1570
setup, 1570
to_irq, 1571
L4vbus::Icu, 1572
Src_dev_handle, 1574
Src_types, 1574
vicu, 1574
L4vbus::Pci_dev, 1575
cfg_read, 1578
cfg_write, 1578
irq_enable, 1579
L4vbus::Pci_host_bridge, 1580
cfg_read, 1583
cfg_write, 1584
irq_enable, 1585
L4vbus::Pm< DEC >, 1586
L4vbus::Vbus, 1587
assign_dma_domain, 1590, 1591
release_ioport, 1592
release_resource, 1593
request_ioport, 1593
request_resource, 1595
root, 1596
l4vbus_assign_dma_domain
L4 Vbus functions, 350
L4VBUS_DEVICE_F_CHILDREN
vbus_types.h, 2392
l4vbus_device_flags_t
vbus_types.h, 2391
l4vbus_device_t, 1596
L4vbus_dma_domain_assign_flags
L4 Vbus functions, 350
L4VBUS_DMAD_BIND
L4 Vbus functions, 350
L4VBUS_DMAD_KERNEL_DMA_SPACE
L4 Vbus functions, 350
L4VBUS_DMAD_L4RE_DMA_SPACE
L4 Vbus functions, 350
L4VBUS_DMAD_UNBIND
L4 Vbus functions, 350
l4vbus_get_device
L4 Vbus functions, 351
l4vbus_get_device_by_hid
L4 Vbus functions, 352
l4vbus_get_hid
L4 Vbus functions, 353
l4vbus_get_next_device
L4 Vbus functions, 353
l4vbus_get_resource
L4 Vbus functions, 355
l4vbus_gpio_config_get
L4vbus GPIO functions, 375
l4vbus_gpio_config_pad
L4vbus GPIO functions, 376
l4vbus_gpio_config_pull
L4vbus GPIO functions, 376
L4vbus_gpio_generic_func
L4vbus GPIO functions, 374
l4vbus_gpio_get
L4vbus GPIO functions, 377
l4vbus_gpio_multi_config_pad
L4vbus GPIO functions, 378
l4vbus_gpio_multi_get
L4vbus GPIO functions, 379
l4vbus_gpio_multi_set
L4vbus GPIO functions, 379
l4vbus_gpio_multi_setup
L4vbus GPIO functions, 380
L4VBUS_GPIO_PIN_PULL_DOWN
L4vbus GPIO functions, 375
L4VBUS_GPIO_PIN_PULL_NONE
L4vbus GPIO functions, 375
L4VBUS_GPIO_PIN_PULL_UP
L4vbus GPIO functions, 375
L4vbus_gpio_pull_modes
L4vbus GPIO functions, 375
l4vbus_gpio_set
L4vbus GPIO functions, 381
l4vbus_gpio_setup
L4vbus GPIO functions, 382

L4VBUS_GPIO_SETUP_INPUT
L4vbus GPIO functions, 374
L4VBUS_GPIO_SETUP_IRQ
L4vbus GPIO functions, 374
L4VBUS_GPIO_SETUP_OUTPUT
L4vbus GPIO functions, 374
l4vbus_gpio_to_irq
L4vbus GPIO functions, 383
L4VBUS_ICU_SRC_DEV_HANDLE
vbus.h, 2385
l4vbus_icu_src_types
vbus.h, 2385
L4VBUS_IFACE_SHIFT
vbus_interfaces.h, 2388
l4vbus_iface_type_t
vbus_interfaces.h, 2388, 2389
l4vbus_is_compatible
L4 Vbus functions, 356
L4VBUS_NULL
vbus.h, 2385
l4vbus_pci_cfg_read
L4vbus PCI functions, 384
l4vbus_pci_cfg_write
L4vbus PCI functions, 385
l4vbus_pci_irq_enable
L4vbus PCI functions, 386
l4vbus_pcidev_cfg_read
L4vbus PCI functions, 387
l4vbus_pcidev_cfg_write
L4vbus PCI functions, 387
l4vbus_pcidev_irq_enable
L4vbus PCI functions, 388
l4vbus_release_ioport
L4 Vbus functions, 356
l4vbus_release_resource
L4 Vbus functions, 357
l4vbus_request_ioport
L4 Vbus functions, 358
l4vbus_request_resource
L4 Vbus functions, 358
L4VBUS_RESOURCE_BUS
vbus_types.h, 2392
L4VBUS_RESOURCE_DMA_DOMAIN
vbus_types.h, 2392
L4VBUS_RESOURCE_F_MEM_MMIO_READ
vbus_types.h, 2392
L4VBUS_RESOURCE_F_MEM_MMIO_WRITE
vbus_types.h, 2392
L4VBUS_RESOURCE_F_MEM_READ_ONLY
vbus_types.h, 2392
l4vbus_resource_flags_t
vbus_types.h, 2392
L4VBUS_RESOURCE_GPIO
vbus_types.h, 2392
L4VBUS_RESOURCE_INVALID
vbus_types.h, 2392
L4VBUS_RESOURCE_IRQ
vbus_types.h, 2392
L4VBUS_RESOURCE_MAX
vbus_types.h, 2392
L4VBUS_RESOURCE_MEM
vbus_types.h, 2392
L4VBUS_RESOURCE_PORT
vbus_types.h, 2392
l4vbus_resource_t, 1597
l4vbus_resource_type_t
vbus_types.h, 2392
L4VBUS_ROOT_BUS
vbus.h, 2385
l4vbus_subinterface_supported
vbus_interfaces.h, 2389
l4vbus_vicu_get_cap
L4 Vbus functions, 359
L4vcpu::State, 1598
add, 1599
clear, 1600
set, 1600
State, 1599
L4vcpu::Vcpu, 1600
cast, 1604
entry_ip, 1604
entry_sp, 1605
ext_alloc, 1605
i, 1606
irq_disable_save, 1606
irq_enable, 1607
irq_restore, 1608
is_irq_entry, 1608
is_page_fault_entry, 1609
r, 1609
saved_state, 1610
state, 1610, 1611
task, 1611
wait_for_event, 1611
l4vcpu_ext_alloc
Extended vCPU support, 225
l4vcpu_irq_disable
vCPU Support Library, 597
l4vcpu_irq_disable_save
vCPU Support Library, 598
l4vcpu_irq_enable
vCPU Support Library, 599
l4vcpu_irq_restore
vCPU Support Library, 600
l4vcpu_is_irq_entry
vCPU Support Library, 601
l4vcpu_is_page_fault_entry
vCPU Support Library, 602
l4vcpu_print_state
vCPU Support Library, 602
l4vcpu_wait_for_event
vCPU Support Library, 603
L4virtio, 664
L4virtio::Device, 1612
config_queue, 1615
device_config, 1616

device_notification_irq, 1616
register_ds, 1617
register_iface, 1618
set_status, 1618
L4virtio::Driver::Block_device, 1619
add_block, 1622
process_request, 1623
process_used_queue, 1624
send_request, 1624
setup_device, 1625
start_request, 1626
L4virtio::Driver::Block_device::Handle, 1627
L4virtio::Driver::Device, 1628
bind_notification_irq, 1631
config_queue, 1632
driver_acknowledge, 1633
driver_connect, 1633
max_queue_size, 1635
register_ds, 1636
send, 1637
send_and_wait, 1638
wait, 1639
wait_for_next_used, 1640
L4virtio::Driver::Virtqueue, 1641
alloc_descriptor, 1644
desc, 1645
enqueue_descriptor, 1646
find_next_used, 1646
free_descriptor, 1647
init_queue, 1648
initialize_rings, 1649
L4virtio::Ptr< T >, 1650
get, 1652
Invalid, 1652
Invalid_type, 1652
is_valid, 1653
L4virtio::Svr::Bad_descriptor, 1654
Bad_address, 1655
Bad_descriptor, 1655
Bad_flags, 1655
Bad_next, 1655
Bad_rights, 1655
Bad_size, 1655
Error, 1655
message, 1656
L4virtio::Svr::Block_dev_base< Ds_data >, 1656
Block_dev_base, 1660
finalize_request, 1660
get_writeback, 1661
process_request, 1661
set_blk_size, 1662
set_config_wce, 1662
set_discard, 1662
set_size_max, 1664
set_topology, 1664
set_write_zeroes, 1664
L4virtio::Svr::Block_request< Ds_data >, 1665
data_size, 1666
next_block, 1667
L4virtio::Svr::Data_buffer, 1668
copy_to, 1669
Data_buffer, 1669
done, 1670
set, 1670
skip, 1671
L4virtio::Svr::Dev_config, 1672
change_queue_config, 1674
Dev_config, 1673, 1674
ds, 1675
get_cmd, 1675
guest_features, 1676
hdr, 1677
negotiated_features, 1677
qconfig, 1678
reset_cmd, 1678
reset_queue, 1679
set_device_needs_reset, 1680
set_status, 1681
status, 1682
L4virtio::Svr::Dev_features, 1683
L4virtio::Svr::Dev_status, 1684
running, 1686
L4virtio::Svr::Device_t< DATA >, 1687
device_error, 1689
device_notify_irq, 1690
handle_mem_cmd_write, 1690
init_mem_info, 1691
register_driver_irq, 1692
reset_queue_config, 1692
setup_queue, 1693
L4virtio::Svr::Driver_mem_list_t< DATA >, 1694
add, 1697
find, 1698
full, 1698
init, 1699
load_desc, 1700–1702
remove, 1703
L4virtio::Svr::Driver_mem_region_t< DATA >, 1704
contains, 1706
Driver_mem_region_t, 1706
drv_base, 1707
ds, 1707
ds_offset, 1708
empty, 1708
flags, 1708
is_writable, 1708
local, 1709
local_base, 1710
size, 1710
L4virtio::Svr::Request_processor, 1711
current_flags, 1712
has_more, 1712
next, 1713
start, 1715, 1716
L4virtio::Svr::Virtqueue, 1718
consumed, 1720

desc, 1721
desc_avail, 1721
disable_notify, 1722
enable_notify, 1722
next_avail, 1723
L4virtio::Svr::Virtqueue::Head_desc, 1724
desc, 1724
operator Null_ptr_check const ∗, 1725
valid, 1725
L4virtio::Virtqueue, 1726
avail_align, 1729
avail_size, 1729
desc_align, 1730
desc_size, 1730
disable, 1731
dump, 1732
get_avail_idx, 1732
get_tail_avail_idx, 1733
no_notify_guest, 1733
no_notify_host, 1734, 1735
num, 1735
ready, 1736
setup, 1737
setup_simple, 1738
total_size, 1739, 1740
used_align, 1741
used_size, 1741
L4virtio::Virtqueue::Avail, 1742
L4virtio::Virtqueue::Avail::Flags, 1744
no_irq_bfm_t, 1744
L4virtio::Virtqueue::Desc, 1745
L4virtio::Virtqueue::Desc::Flags, 1747
indirect_bfm_t, 1748
next_bfm_t, 1748
write_bfm_t, 1749
L4virtio::Virtqueue::Used, 1749
L4virtio::Virtqueue::Used::Flags, 1750
no_notify_bfm_t, 1751
L4virtio::Virtqueue::Used_elem, 1752
Used_elem, 1752
l4virtio_block_config_t, 1753
l4virtio_block_discard_t, 1754
l4virtio_block_header_t, 1755
L4virtio_block_operations
L4 VIRTIO Block Device, 337
L4VIRTIO_BLOCK_S_IOERR
L4 VIRTIO Block Device, 338
L4VIRTIO_BLOCK_S_OK
L4 VIRTIO Block Device, 338
L4VIRTIO_BLOCK_S_UNSUPP
L4 VIRTIO Block Device, 338
L4virtio_block_status
L4 VIRTIO Block Device, 338
L4VIRTIO_BLOCK_T_DISCARD
L4 VIRTIO Block Device, 338
L4VIRTIO_BLOCK_T_FLUSH
L4 VIRTIO Block Device, 338
L4VIRTIO_BLOCK_T_GET_ID
L4 VIRTIO Block Device, 338
L4VIRTIO_BLOCK_T_IN
L4 VIRTIO Block Device, 338
L4VIRTIO_BLOCK_T_OUT
L4 VIRTIO Block Device, 338
L4VIRTIO_BLOCK_T_WRITE_ZEROES
L4 VIRTIO Block Device, 338
L4VIRTIO_CMD_CFG_QUEUE
L4 VIRTIO Transport Layer, 342
L4VIRTIO_CMD_MASK
L4 VIRTIO Transport Layer, 342
L4VIRTIO_CMD_NONE
L4 VIRTIO Transport Layer, 342
L4VIRTIO_CMD_SET_STATUS
L4 VIRTIO Transport Layer, 342
l4virtio_config_hdr_t, 1756
status, 1757
l4virtio_config_queue
L4 VIRTIO Transport Layer, 345
l4virtio_config_queue_t, 1757
L4 VIRTIO Transport Layer, 342
l4virtio_config_queues
L4 VIRTIO Transport Layer, 345
l4virtio_device_config
L4 VIRTIO Transport Layer, 346
l4virtio_device_config_ds
L4 VIRTIO Transport Layer, 346
L4virtio_device_ids
L4 VIRTIO Transport Layer, 343
l4virtio_device_notification_irq
L4 VIRTIO Transport Layer, 346
L4virtio_device_status
L4 VIRTIO Transport Layer, 344
L4virtio_feature_bits
L4 VIRTIO Transport Layer, 344
L4VIRTIO_FEATURE_CMD_CONFIG
L4 VIRTIO Transport Layer, 344
L4VIRTIO_FEATURE_VERSION_1
L4 VIRTIO Transport Layer, 344
L4VIRTIO_ID_9P
L4 VIRTIO Transport Layer, 343
L4VIRTIO_ID_BALLOON
L4 VIRTIO Transport Layer, 343
L4VIRTIO_ID_BLOCK
L4 VIRTIO Transport Layer, 343
L4VIRTIO_ID_CAIF
L4 VIRTIO Transport Layer, 344
L4VIRTIO_ID_CONSOLE
L4 VIRTIO Transport Layer, 343
L4VIRTIO_ID_CRYPTO
L4 VIRTIO Transport Layer, 344
L4VIRTIO_ID_GPU
L4 VIRTIO Transport Layer, 344
L4VIRTIO_ID_INPUT
L4 VIRTIO Transport Layer, 344
L4VIRTIO_ID_NET
L4 VIRTIO Transport Layer, 343
L4VIRTIO_ID_RNG

L4 VIRTIO Transport Layer, 343
L4VIRTIO_ID_RPMSG
L4 VIRTIO Transport Layer, 343
L4VIRTIO_ID_RPROC_SERIAL
L4 VIRTIO Transport Layer, 344
L4VIRTIO_ID_SCSI
L4 VIRTIO Transport Layer, 343
L4VIRTIO_ID_SOCK
L4 VIRTIO Transport Layer, 344
L4VIRTIO_ID_VSOCK
L4 VIRTIO Transport Layer, 344
l4virtio_input_absinfo_t, 1759
l4virtio_input_config_t, 1759
l4virtio_input_devids_t, 1760
l4virtio_input_event_t, 1761
L4VIRTIO_IRQ_STATUS_CONFIG
L4 VIRTIO Transport Layer, 343
L4VIRTIO_IRQ_STATUS_VRING
L4 VIRTIO Transport Layer, 343
L4VIRTIO_OP_CONFIG_QUEUE
L4 VIRTIO Transport Layer, 343
L4VIRTIO_OP_REGISTER_DS
L4 VIRTIO Transport Layer, 343
L4VIRTIO_OP_REGISTER_IFACE
L4 VIRTIO Transport Layer, 343
L4VIRTIO_OP_SET_STATUS
L4 VIRTIO Transport Layer, 343
l4virtio_register_ds
L4 VIRTIO Transport Layer, 347
l4virtio_register_iface
L4 VIRTIO Transport Layer, 347
l4virtio_set_status
L4 VIRTIO Transport Layer, 348
L4VIRTIO_STATUS_ACKNOWLEDGE
L4 VIRTIO Transport Layer, 344
L4VIRTIO_STATUS_DEVICE_NEEDS_RESET
L4 VIRTIO Transport Layer, 344
L4VIRTIO_STATUS_DRIVER
L4 VIRTIO Transport Layer, 344
L4VIRTIO_STATUS_DRIVER_OK
L4 VIRTIO Transport Layer, 344
L4VIRTIO_STATUS_FAILED
L4 VIRTIO Transport Layer, 344
L4VIRTIO_STATUS_FEATURES_OK
L4 VIRTIO Transport Layer, 344
length
L4::Ipc::Varg, 1018
Libedid_block_size
EDID parsing functionality, 182
libedid_check_header
EDID parsing functionality, 182
libedid_checksum
EDID parsing functionality, 182
Libedid_consts
EDID parsing functionality, 181
libedid_dump
EDID parsing functionality, 182
libedid_dump_standard_timings
EDID parsing functionality, 183
libedid_num_ext_blocks
EDID parsing functionality, 183
libedid_pnp_id
EDID parsing functionality, 183
libedid_prefered_resolution
EDID parsing functionality, 184
libedid_revision
EDID parsing functionality, 184
libedid_version
EDID parsing functionality, 184
Link
L4Re::Namespace, 1359
link
L4Re::Vfs::Directory, 1462
List_alloc
cxx::List_alloc, 789
list_alloc.h
l4la_alloc, 2355
l4la_avail, 2355
l4la_dump, 2355
l4la_free, 2355
l4la_init, 2356
load_desc
L4virtio::Svr::Driver_mem_list_t< DATA >, 1700–1702
local
L4virtio::Svr::Driver_mem_region_t< DATA >, 1709
local_base
L4virtio::Svr::Driver_mem_region_t< DATA >, 1710
local_id_received
L4::Ipc::Gen_fpage< T >, 960
log
L4Re::Env, 1327
Log interface, 389
l4re_log_print, 390
l4re_log_print_srv, 390
l4re_log_printn, 391
l4re_log_printn_srv, 392
Logging interface, 393
loop
L4::Server< LOOP_HOOKS >, 1155
L4Re::Util::Registry_server< LOOP_HOOKS >, 1445
Low-Level Thread Functions, 393
Low_mask
cxx::Bitfield< T, LSB, MSB >, 700
lower_bound_node
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 739
cxx::Bits::Bst< Node, Get_key, Compare >, 758
Lsb
cxx::Bitfield< T, LSB, MSB >, 700
lseek64
L4Re::Vfs::Regular_file, 1486
Lstr

L4::Factory::Lstr, 918

main_thread
L4Re::Env, 1327, 1328
make_cap
L4::Ipc, 624
make_cap_full
L4::Ipc, 624
make_cap_rw
L4::Ipc, 625
make_cap_rws
L4::Ipc, 626
make_ref_cap
L4Re Capability API, 367
make_ref_del_cap
L4Re Capability API, 367
make_shared_cap
L4Re, 647
L4Re::Util, 660
make_shared_del_cap
L4Re, 648
L4Re::Util, 660
make_unique_cap
L4Re, 649
L4Re::Util, 661
make_unique_del_cap
L4Re, 649
L4Re::Util, 661
map
L4::Task, 1177
L4Re::Dataspace, 1307
L4Re::Dma_space, 1318
L4Re::Util::Dataspace_svr, 1411
map_hook
L4Re::Util::Dataspace_svr, 1412
map_region
L4Re::Dataspace, 1308
Mask
cxx::Bitfield< T, LSB, MSB >, 700
mask
L4::Icu, 929
Masks
cxx::Bitfield< T, LSB, MSB >, 700
max
Small C++ Template Library, 492
max_free_slabs
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 687
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 695
max_queue_size
L4virtio::Driver::Device, 1635
MB_ARD_MEMORY
mb_info.h, 2361
MB_ART_MEMORY
mb_info.h, 2361
mb_info.h
l4util_mb_for_each_mmap_entry, 2361
L4UTIL_MB_MEMORY, 2361
MB_ARD_MEMORY, 2361
MB_ART_MEMORY, 2361
Mem
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, 1207
mem_alloc
L4Re::Env, 1328
Mem_alloc_flags
L4Re::Mem_alloc, 1350
Mem_desc
L4::Kip::Mem_desc, 1087
Mem_type
L4::Kip::Mem_desc, 1086
Memory allocator, 394
l4re_ma_alloc, 395
l4re_ma_alloc_align, 396
l4re_ma_alloc_align_srv, 397
l4re_ma_flags, 394
Memory descriptors (C version), 398
l4_kernel_info_get_mem_desc_end, 400
l4_kernel_info_get_mem_desc_is_virtual, 400
l4_kernel_info_get_mem_desc_start, 401
l4_kernel_info_get_mem_desc_subtype, 401
l4_kernel_info_get_mem_desc_type, 401
l4_kernel_info_get_num_mem_descs, 402
l4_kernel_info_mem_desc_t, 399
l4_kernel_info_set_mem_desc, 402
l4_mem_info_acpi_rsdp, 400
l4_mem_info_sub_type_t, 399
l4_mem_type_archspecific, 400
l4_mem_type_bootloader, 400
l4_mem_type_conventional, 400
l4_mem_type_dedicated, 400
l4_mem_type_info, 400
l4_mem_type_reserved, 400
l4_mem_type_shared, 400
l4_mem_type_t, 400
l4_mem_type_undefined, 400
Memory operations., 403
L4_mem_op_widths, 403
l4_mem_read, 404
L4_MEM_WIDTH_1BYTE, 404
L4_MEM_WIDTH_2BYTE, 404
L4_MEM_WIDTH_4BYTE, 404
l4_mem_write, 404
Memory related, 405
l4_addr_consts_t, 408
l4_bytes_to_mwords, 408
L4_INVALID_ADDR, 408
L4_LOG2_PAGESIZE, 407
L4_LOG2_SUPERPAGESIZE, 407
L4_PAGEMASK, 407
l4_round_page, 408
l4_round_size, 409
L4_SUPERPAGEMASK, 407
L4_SUPERPAGESIZE, 407
l4_trunc_page, 410
l4_trunc_size, 411

message
L4virtio::Svr::Bad_descriptor, 1656
Message Items, 411
L4_ITEM_CONT, 413
L4_ITEM_MAP, 413
l4_map_control, 413
L4_MAP_ITEM_GRANT, 413
L4_MAP_ITEM_MAP, 413
l4_map_obj_control, 414
l4_msg_item_consts_t, 412
L4_RCV_ITEM_LOCAL_ID, 413
L4_RCV_ITEM_SINGLE_CAP, 413
Message Registers (MRs), 415
Message Tag, 415
l4_msgtag, 419
L4_MSGTAG_ERROR, 417
L4_MSGTAG_FLAGS, 417
l4_msgtag_flags, 417, 420
l4_msgtag_has_error, 420
l4_msgtag_is_exception, 421
l4_msgtag_is_io_page_fault, 422
l4_msgtag_is_page_fault, 422
l4_msgtag_is_preemption, 423
l4_msgtag_is_sigma0, 424
l4_msgtag_is_sys_exception, 424
l4_msgtag_items, 425
l4_msgtag_label, 426
l4_msgtag_protocol, 418
L4_MSGTAG_SCHEDULE, 417
l4_msgtag_t, 417
L4_MSGTAG_TRANSFER_FPU, 417
l4_msgtag_words, 427
L4_platform_ctl_proto, 418
L4_PROTO_ALLOW_SYSCALL, 418
L4_PROTO_DEBUGGER, 418
L4_PROTO_DMA_SPACE, 418
L4_PROTO_EXCEPTION, 418
L4_PROTO_FACTORY, 418
L4_PROTO_IO_PAGE_FAULT, 418
L4_PROTO_IOMMU, 418
L4_PROTO_IRQ, 418
L4_PROTO_IRQ_MUX, 418
L4_PROTO_IRQ_SENDER, 418
L4_PROTO_KOBJECT, 418
L4_PROTO_LOG, 418
L4_PROTO_META, 418
L4_PROTO_NONE, 418
L4_PROTO_PAGE_FAULT, 418
L4_PROTO_PF_EXCEPTION, 418
L4_PROTO_PLATFORM_CTL, 419
L4_PROTO_PREEMPTION, 418
L4_PROTO_SCHEDULER, 418
L4_PROTO_SEMAPHORE, 418
L4_PROTO_SIGMA0, 418
L4_PROTO_SMCCC, 418
L4_PROTO_SYS_EXCEPTION, 418
L4_PROTO_TASK, 418
L4_PROTO_THREAD, 418
L4_PROTO_VM, 418
min
Small C++ Template Library, 492
mkdir
L4Re::Vfs::Directory, 1462
mmio_read
L4Re::Mmio_space, 1354
mmio_write
L4Re::Mmio_space, 1355
Mode
L4Re::Util::Event_t< PAYLOAD >, 1424
Mode_irq
L4Re::Util::Event_t< PAYLOAD >, 1424
Mode_polling
L4Re::Util::Event_t< PAYLOAD >, 1424
modify_senders
L4::Thread, 1186
mount
L4Re::Vfs::File_system, 1468
L4Re::Vfs::Fs, 1473
move
L4::Cap< T >, 872
L4::Cap_base, 880
Mr_bytes
L4::Ipc::Msg, 629
Mr_words
L4::Ipc::Msg, 629
Msb
cxx::Bitfield< T, LSB, MSB >, 700
msg_add
L4::Ipc::Msg, 632
msg_get
L4::Ipc::Msg, 633
Msg_ptr
L4::Ipc::Msg_ptr< T >, 1000
msg_ptr
L4::Ipc, 627
msi_info
L4::Icu, 930

N
cxx::Bits::Direction, 767
Name-space API, 428
Name_max
L4Re::Inhibitor, 1342
Namespace interface, 428
l4re_ns_query_srv, 429
l4re_ns_query_to_srv, 430
l4re_ns_register_flags, 429
l4re_ns_register_obj_srv, 431
negotiated_features
L4virtio::Svr::Dev_config, 1677
next
L4Re::Event_buffer_t< PAYLOAD >, 1337
L4virtio::Svr::Request_processor, 1713
next_avail
L4virtio::Svr::Virtqueue, 1723
next_bfm_t
L4virtio::Virtqueue::Desc::Flags, 1748

next_block
L4virtio::Svr::Block_request< Ds_data >, 1667
next_device
L4vbus::Device, 1555
next_lock_info
L4Re::Inhibitor, 1343
next_timeout
L4::Ipc_svr::Timeout_queue, 1057
no_demand
L4::Type_info::Demand, 1204
No_init
L4::Cap_base, 876
No_init_type
L4::Cap_base, 876
no_irq_bfm_t
L4virtio::Virtqueue::Avail::Flags, 1744
no_notify_bfm_t
L4virtio::Virtqueue::Used::Flags, 1751
no_notify_guest
L4virtio::Virtqueue, 1733
no_notify_host
L4virtio::Virtqueue, 1734, 1735
No_sync
L4Re::Dma_space, 1316
None
L4::Types::Flags< BITS_ENUM, UNDERLYING >, 1232
L4Re::Dma_space, 1316
None_type
L4::Types::Flags< BITS_ENUM, UNDERLYING >, 1232
Normal
L4Re::Dataspace::F, 1310
NT_ASRS
ELF binary format, 201
NT_AUXV
ELF binary format, 201
NT_FPREGSET
ELF binary format, 201
NT_GWINDOWS
ELF binary format, 201
NT_LWPSINFO
ELF binary format, 201
NT_LWPSTATUS
ELF binary format, 201
NT_PLATFORM
ELF binary format, 201
NT_PRCRED
ELF binary format, 201
NT_PRFPXREG
ELF binary format, 201
NT_PRPSINFO
ELF binary format, 201
NT_PRSTATUS
ELF binary format, 201
NT_PRXREG
ELF binary format, 201
NT_PSINFO
ELF binary format, 201
NT_PSTATUS
ELF binary format, 201
NT_TASKSTRUCT
ELF binary format, 201
NT_UTSNAME
ELF binary format, 201
NT_VERSION
ELF binary format, 202
num
L4virtio::Virtqueue, 1735
num_interfaces
L4::Meta, 1115

obj_cap
L4::Epiface, 899
L4::Epiface_t0< RPC_IFACE, BASE >, 906
L4::Irqep_t< Derived, BASE, bool >, 1084
Object Invocation, 432
l4_ipc, 435
l4_ipc_call, 436
l4_ipc_receive, 437
l4_ipc_reply_and_wait, 438
l4_ipc_send, 440
l4_ipc_send_and_wait, 441
l4_ipc_sleep, 442
l4_ipc_wait, 443
l4_sndfpage_add, 444
l4_syscall_flags_t, 434
L4_SYSF_CALL, 434
L4_SYSF_NONE, 434
L4_SYSF_OPEN_WAIT, 434
L4_SYSF_RECV, 434
L4_SYSF_REPLY, 434
L4_SYSF_REPLY_AND_WAIT, 434
L4_SYSF_SEND, 434
L4_SYSF_SEND_AND_WAIT, 434
L4_SYSF_WAIT, 434
Object_registry
L4Re::Util::Object_registry, 1435
object_size
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 687
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 695
objects_per_slab
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 687
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 695
offset
l4_sched_cpu_set_t, 1271
operator l4_msgtag_t
L4::Factory::S, 922
operator new
Small C++ Template Library, 493
operator Null_ptr_check const *
L4virtio::Svr::Virtqueue::Head_desc, 1725
operator!

cxx::Bits::Direction, 767
operator!=
L4vbus::Device, 1556
operator<<
ipc_stream, 1900–1902
operator>>
ipc_stream, 1903–1907
operator*
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, 744
operator()
cxx::Pair_first_compare< Cmp, Typ >, 805
operator->
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, 744
operator=
cxx::Auto_ptr< T >, 667
operator==
L4Re::Video::Color_component, 1493
L4Re::Video::Pixel_info, 1510
L4vbus::Device, 1556
operator[]
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, 673
cxx::Bitmap_base, 722
cxx::List< D, Alloc >, 787
Overwrite
L4Re::Namespace, 1359

p_flags
Elf32_Phdr, 841
Elf64_Phdr, 852
p_type
Elf32_Phdr, 841
Elf64_Phdr, 852
padding
L4Re::Video::Pixel_info, 1511
page_fault
L4::Pager, 1122
page_shift
L4Re::Util::Dataspace_svr, 1412
Pager
L4Re::Rm::F, 1385
pager
L4::Thread::Attr, 1195
Pair
cxx::Pair< First, Second >, 803
Pair_first_compare
cxx::Pair_first_compare< Cmp, Typ >, 805
parent
L4Re::Env, 1329
Parent API, 445
parse_cmdline
Comfortable Command Line Parsing, 162
Partly_resolved
L4Re::Namespace, 1358
PF_ARM_SB
ELF binary format, 193
PF_MASKOS
ELF binary format, 203
PF_MASKPROC
ELF binary format, 203
PF_R
ELF binary format, 203
PF_W
ELF binary format, 203
PF_X
ELF binary format, 203
Phys_space
L4Re::Dma_space, 1317
pin
L4vbus::Gpio_module, 1561
L4vbus::Gpio_pin, 1569
Pinned
L4Re::Mem_alloc, 1350
Pixel_info
L4Re::Video::Pixel_info, 1505, 1506
Platform Control C API, 446
l4_platform_ctl_cpu_allow_shutdown, 446
l4_platform_ctl_cpu_disable, 447
l4_platform_ctl_cpu_enable, 447
l4_platform_ctl_system_shutdown, 448
l4_platform_ctl_system_suspend, 448
Poll_timeout_kipclock
L4::Poll_timeout_kipclock, 1127
pop_front
cxx::Bits::Smart_ptr_list< ITEM >, 770
cxx::H_list< T, POLICY >, 778
cxx::S_list< T, POLICY >, 814
port_io.h
l4util_ioport_map, 1821
Ports
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, 1207
print
L4Re::Log, 1347
printn
L4Re::Log, 1347
process
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, 1416
process_request
L4virtio::Driver::Block_device, 1623
L4virtio::Svr::Block_dev_base< Ds_data >, 1661
process_used_queue
L4virtio::Driver::Block_device, 1624
Producer, 449, 451
l4shmc_chunk_ready, 449
l4shmc_chunk_ready_sig, 450
l4shmc_chunk_try_to_take, 450
l4shmc_is_chunk_clear, 451
l4shmc_trigger, 451
PROTO_ANY
L4, 614
proto_dispatch
L4::Kobject_typeid< T >, 1109
L4::Server_object_t< IFACE, BASE >, 1164

PROTO_EMPTY
L4, 614
protocols.h
L4RE_PROTO_DATASPACE, 2045
L4RE_PROTO_DEBUG, 2045
L4RE_PROTO_DMA_SPACE, 2045
L4RE_PROTO_EVENT, 2045
L4RE_PROTO_GOOS, 2045
L4RE_PROTO_INHIBITOR, 2045
L4RE_PROTO_MMIO_SPACE, 2045
L4RE_PROTO_NAMESPACE, 2045
L4RE_PROTO_PARENT, 2045
L4RE_PROTO_RM, 2045
L4RE_PROTO_RSVD_1, 2045
L4re_protocols, 2045
pSLIM_BMAP_START_LSB
bitmap.h, 1945
PT_DYNAMIC
ELF binary format, 203
PT_GNU_EH_FRAME
ELF binary format, 203
PT_GNU_RELRO
ELF binary format, 203
PT_GNU_STACK
ELF binary format, 203
PT_HIOS
ELF binary format, 203
PT_HIPROC
ELF binary format, 203
PT_INTERP
ELF binary format, 203
PT_L4_AUX
ELF binary format, 203
PT_L4_KIP
ELF binary format, 203
PT_L4_STACK
ELF binary format, 203
PT_LOAD
ELF binary format, 203
PT_LOOS
ELF binary format, 203
PT_LOPROC
ELF binary format, 203
PT_NOTE
ELF binary format, 203
PT_NULL
ELF binary format, 203
PT_NUM
ELF binary format, 203
PT_PHDR
ELF binary format, 203
PT_SHLIB
ELF binary format, 203
PT_TLS
ELF binary format, 203
ptr
cxx::Ref_ptr< T, CNT >, 810
push_back
cxx::List_item, 793
push_front
cxx::List_item, 794
put
L4::Factory::S, 922–924
L4::Ipc::Iostream, 967, 968
L4::Ipc::Ostream, 1005, 1006
L4Re::Event_buffer_t< PAYLOAD >, 1337

qconfig
L4virtio::Svr::Dev_config, 1678
query
L4Re::Namespace, 1359, 1360
query_log_name
L4::Debugger, 890
query_log_typeid
L4::Debugger, 891
Query_result_flags
L4Re::Namespace, 1358

R
cxx::Bits::Direction, 767
L4Re::Dataspace::F, 1310
L4Re::Rm::F, 1385
r
L4Re::Video::Pixel_info, 1511
L4vcpu::Vcpu, 1609
R_386_32
ELF binary format, 205
R_386_COPY
ELF binary format, 205
R_386_GLOB_DAT
ELF binary format, 205
R_386_GOT32
ELF binary format, 205
R_386_GOTOFF
ELF binary format, 205
R_386_GOTPC
ELF binary format, 205
R_386_JMP_SLOT
ELF binary format, 205
R_386_NONE
ELF binary format, 205
R_386_NUM
ELF binary format, 205
R_386_PC32
ELF binary format, 205
R_386_PLT32
ELF binary format, 205
R_386_RELATIVE
ELF binary format, 205
R_386_TLS_DTPMOD32
ELF binary format, 205
R_386_TLS_DTPOFF32
ELF binary format, 205
R_386_TLS_GD
ELF binary format, 205
R_386_TLS_GD_32
ELF binary format, 205

R_386_TLS_GD_CALL
ELF binary format, 205
R_386_TLS_GD_POP
ELF binary format, 205
R_386_TLS_GD_PUSH
ELF binary format, 205
R_386_TLS_GOTIE
ELF binary format, 205
R_386_TLS_IE
ELF binary format, 205
R_386_TLS_IE_32
ELF binary format, 205
R_386_TLS_LDM
ELF binary format, 205
R_386_TLS_LDM_32
ELF binary format, 205
R_386_TLS_LDM_CALL
ELF binary format, 205
R_386_TLS_LDM_POP
ELF binary format, 205
R_386_TLS_LDM_PUSH
ELF binary format, 205
R_386_TLS_LDO_32
ELF binary format, 205
R_386_TLS_LE
ELF binary format, 205
R_386_TLS_LE_32
ELF binary format, 205
R_386_TLS_TPOFF
ELF binary format, 205
R_386_TLS_TPOFF32
ELF binary format, 205
R_AARCH64_NONE
ELF binary format, 205
R_ARM_ABS12
ELF binary format, 206
R_ARM_ABS16
ELF binary format, 206
R_ARM_ABS32
ELF binary format, 206
R_ARM_ABS8
ELF binary format, 206
R_ARM_COPY
ELF binary format, 206
R_ARM_GLOB_DAT
ELF binary format, 206
R_ARM_GOT32
ELF binary format, 206
R_ARM_GOTOFF
ELF binary format, 206
R_ARM_GOTPC
ELF binary format, 206
R_ARM_JUMP_SLOT
ELF binary format, 206
R_ARM_NONE
ELF binary format, 206
R_ARM_NUM
ELF binary format, 206
R_ARM_PC24
ELF binary format, 206
R_ARM_PLT32
ELF binary format, 206
R_ARM_REL32
ELF binary format, 206
R_ARM_RELATIVE
ELF binary format, 206
R_ARM_THM_PC11
ELF binary format, 206
R_ARM_THM_PC9
ELF binary format, 206
R_X86_64_16
ELF binary format, 207
R_X86_64_32
ELF binary format, 207
R_X86_64_32S
ELF binary format, 207
R_X86_64_64
ELF binary format, 206
R_X86_64_8
ELF binary format, 207
R_X86_64_COPY
ELF binary format, 207
R_X86_64_DTPMOD64
ELF binary format, 207
R_X86_64_DTPOFF32
ELF binary format, 207
R_X86_64_DTPOFF64
ELF binary format, 207
R_X86_64_GLOB_DAT
ELF binary format, 207
R_X86_64_GOT32
ELF binary format, 206
R_X86_64_GOTPCREL
ELF binary format, 207
R_X86_64_GOTTPOFF
ELF binary format, 207
R_X86_64_JUMP_SLOT
ELF binary format, 207
R_X86_64_NONE
ELF binary format, 206
R_X86_64_PC16
ELF binary format, 207
R_X86_64_PC32
ELF binary format, 206
R_X86_64_PC8
ELF binary format, 207
R_X86_64_PLT32
ELF binary format, 206
R_X86_64_RELATIVE
ELF binary format, 207
R_X86_64_TLSGD
ELF binary format, 207
R_X86_64_TLSLD
ELF binary format, 207
R_X86_64_TPOFF32
ELF binary format, 207

R_X86_64_TPOFF64
ELF binary format, 207
Random number support, 452
l4util_rand, 452
l4util_srand, 453
rbegin
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 740
cxx::Bits::Bst< Node, Get_key, Compare >, 759
rcv_cap
L4::Ipc_svr::Server_iface, 1048, 1049
rcv_endpoint.h
L4_RCV_EP_BIND_OP, 2247
L4_rcv_ep_ops, 2247
read
L4::Ipc, 627
L4::Vcon, 1250
read_with_flags
L4::Vcon, 1251
readv
L4Re::Vfs::Regular_file, 1486
ready
L4virtio::Virtqueue, 1736
realloc_rcv_cap
L4::Ipc_svr::Server_iface, 1050
L4Re::Util::Br_manager, 1392
Realtime API, 453
receive
L4::Ipc::Istream, 976
L4::Irq, 1069
Ref_ptr
cxx::Ref_ptr< T, CNT >, 809, 810
refresh
L4Re::Util::Video::Goos_svr, 1451
L4Re::Video::View, 1515
Region
L4Re::Rm, 1374
Region map API, 453
Region map interface, 454
l4re_rm_attach, 456
l4re_rm_attach_srv, 458
L4RE_RM_CACHING_SHIFT, 456
l4re_rm_detach, 458
l4re_rm_detach_ds, 459
l4re_rm_detach_ds_unmap, 460
l4re_rm_detach_srv, 461
l4re_rm_detach_unmap, 462
L4RE_RM_F_ATTACH_FLAGS, 456
L4RE_RM_F_CACHE_BUFFERED, 456
L4RE_RM_F_CACHE_NORMAL, 456
L4RE_RM_F_CACHE_UNCACHED, 456
L4RE_RM_F_CACHING, 456
L4RE_RM_F_EAGER_MAP, 456
L4RE_RM_F_IN_AREA, 456
L4RE_RM_F_NO_ALIAS, 456
L4RE_RM_F_PAGER, 456
L4RE_RM_F_R, 456
L4RE_RM_F_RESERVED, 456
L4RE_RM_F_SEARCH_ADDR, 456
l4re_rm_find, 463
l4re_rm_find_srv, 464
l4re_rm_flags_values, 456
l4re_rm_free_area, 465
l4re_rm_free_area_srv, 466
L4RE_RM_REGION_FLAGS, 456
l4re_rm_reserve_area, 466
l4re_rm_reserve_area_srv, 467
l4re_rm_show_lists, 468
Region_flag_shifts
L4Re::Rm, 1375
Region_flags
L4Re::Rm::F, 1385
Region_flags_mask
L4Re::Rm::F, 1385
register_del_irq
L4::Thread, 1187
register_driver_irq
L4virtio::Svr::Device_t< DATA >, 1692
register_ds
L4virtio::Device, 1617
L4virtio::Driver::Device, 1636
Register_flags
L4Re::Namespace, 1358
register_iface
L4virtio::Device, 1618
register_irq_obj
L4::Registry_iface, 1136
L4Re::Util::Object_registry, 1436
register_obj
L4::Registry_iface, 1137, 1138
L4Re::Namespace, 1361
L4Re::Util::Object_registry, 1436, 1437
Registry_server
L4Re::Util::Registry_server< LOOP_HOOKS >, 1444, 1445
release
cxx::Auto_ptr< T >, 668
cxx::Ref_ptr< T, CNT >, 811
L4Re::Inhibitor, 1344
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, 1405
L4Re::Util::Dataspace_svr, 1413
release_cap
L4::Task, 1178
release_ioport
L4vbus::Vbus, 1592
release_resource
L4vbus::Vbus, 1593
remove
cxx::Avl_tree< Node, Get_key, Compare >, 681
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 741
cxx::H_list< T, POLICY >, 779
cxx::List_item, 794
L4::Ipc_svr::Timeout_queue, 1058
L4virtio::Svr::Driver_mem_list_t< DATA >, 1703

remove_all
cxx::Bits::Bst< Node, Get_key, Compare >, 760
remove_timeout
L4::Ipc_svr::Server_iface, 1051
L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >, 1064
remove_tree
cxx::Bits::Bst< Node, Get_key, Compare >, 761
rename
L4Re::Vfs::Directory, 1463
rend
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 742
cxx::Bits::Bst< Node, Get_key, Compare >, 762
replace
cxx::H_list< T, POLICY >, 780
reply_and_wait
L4::Ipc::Iostream, 968, 969
Reply_compound
Server-Side IPC framework, 475
Reply_mode
Server-Side IPC framework, 475
Reply_separate
Server-Side IPC framework, 475
request_ioport
L4vbus::Vbus, 1593
request_resource
L4vbus::Vbus, 1595
reserve_area
L4Re::Rm, 1382, 1383
Reserved
L4::Kip::Mem_desc, 1087
L4Re::Rm::F, 1385
reset
L4::Ipc::Iostream, 969
L4::Ipc::Istream, 976
reset_cmd
L4virtio::Svr::Dev_config, 1678
reset_queue
L4virtio::Svr::Dev_config, 1679
reset_queue_config
L4virtio::Svr::Device_t< DATA >, 1692
Rights_mask
L4::Ipc::Cap< T >, 956
L4Re::Dataspace::F, 1310
L4Re::Rm::F, 1385
rm
L4Re::Env, 1329, 1330
rmdir
L4Re::Vfs::Directory, 1463
Ro
L4Re::Dataspace::F, 1310
L4Re::Namespace, 1359
root
L4vbus::Vbus, 1596
round_order
L4, 617
Rpcs
L4::Exception, 907
Rs
L4Re::Namespace, 1359
run_thread
L4::Scheduler, 1146
running
L4virtio::Svr::Dev_status, 1686
Runtime_error
L4::Runtime_error, 1141
RW
L4Re::Dataspace::F, 1310
L4Re::Rm::F, 1385
Rw
L4Re::Namespace, 1359
Rws
L4Re::Namespace, 1359
RWX
L4Re::Rm::F, 1385
RX
L4Re::Rm::F, 1385

S
L4::Factory::S, 921
saved_state
L4vcpu::Vcpu, 1610
scan_zero
cxx::Bitmap< BITS >, 716
cxx::Bitmap_base, 723
Scheduler, 469
l4_sched_cpu_set, 471
L4_SCHEDULER_CLASS_FIXED_PRIO, 470
L4_SCHEDULER_CLASS_WFQ, 470
L4_scheduler_classes, 470
l4_scheduler_idle_time, 471
L4_SCHEDULER_IDLE_TIME_OP, 471
l4_scheduler_info, 472
L4_SCHEDULER_INFO_OP, 471
l4_scheduler_is_online, 473
L4_scheduler_ops, 470
l4_scheduler_run_thread, 473
L4_SCHEDULER_RUN_THREAD_OP, 471
scheduler
L4Re::Env, 1330
screen_info
L4Re::Util::Video::Goos_svr, 1451
Search_addr
L4Re::Rm::F, 1385
segment.h
fiasco_amd64_segment_info, 1811
fiasco_amd64_set_fs, 1807, 1811
fiasco_amd64_set_segment_base, 1807, 1812
L4_AMD64_SEGMENT_FS, 1810
L4_AMD64_SEGMENT_GS, 1810
L4_sys_segment, 1810
L4_task_ldt_x86_consts, 1810, 1817
L4_TASK_LDT_X86_ENTRY_SIZE, 1811, 1817
L4_TASK_LDT_X86_MAX_ENTRIES, 1811, 1817
send
L4::Ipc::Ostream, 1006

L4::Vcon, 1252
L4virtio::Driver::Device, 1637
send_and_wait
L4virtio::Driver::Device, 1638
send_request
L4virtio::Driver::Block_device, 1624
Server
L4::Server< LOOP_HOOKS >, 1154
Server-Side IPC framework, 474
Reply_compound, 475
Reply_mode, 475
Reply_separate, 475
server_iface
L4::Epiface, 899
set
cxx::Bitfield< T, LSB, MSB >, 702
L4::Kip::Mem_desc, 1092
L4::Poll_timeout_kipclock, 1128
l4_sched_cpu_set_t, 1271
L4Re::Video::Color_component, 1493
L4vbus::Gpio_module, 1562
L4vbus::Gpio_pin, 1570
L4vcpu::State, 1600
L4virtio::Svr::Data_buffer, 1670
set_attr
L4::Vcon, 1252
set_bit
cxx::Bitmap_base, 724
set_blk_size
L4virtio::Svr::Block_dev_base< Ds_data >, 1662
set_config_wce
L4virtio::Svr::Block_dev_base< Ds_data >, 1662
set_device_needs_reset
L4virtio::Svr::Dev_config, 1680
set_dirty
cxx::Bitfield< T, LSB, MSB >, 702
set_discard
L4virtio::Svr::Block_dev_base< Ds_data >, 1662
set_fd
L4Re::Vfs::Fs, 1473
set_info
L4Re::Video::View, 1515
set_lock
L4Re::Vfs::Regular_file, 1487
set_mode
L4::Icu, 931
set_object_name
L4::Debugger, 892
set_raw
l4_vcon_attr_t, 1279
set_rcv_cap_flags
L4Re::Util::Br_manager, 1393
set_server
L4::Epiface, 900
set_size_max
L4virtio::Svr::Block_dev_base< Ds_data >, 1664
set_status
L4virtio::Device, 1618
L4virtio::Svr::Dev_config, 1681
set_status_flags
L4Re::Vfs::Generic_file, 1477
set_topology
L4virtio::Svr::Block_dev_base< Ds_data >, 1664
set_unshifted
cxx::Bitfield< T, LSB, MSB >, 703
set_unshifted_dirty
cxx::Bitfield< T, LSB, MSB >, 704
set_viewport
L4Re::Video::View, 1516
set_write_zeroes
L4virtio::Svr::Block_dev_base< Ds_data >, 1664
setup
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, 1407
L4vbus::Gpio_module, 1562
L4vbus::Gpio_pin, 1570
L4virtio::Virtqueue, 1737
setup_device
L4virtio::Driver::Block_device, 1625
setup_queue
L4virtio::Svr::Device_t< DATA >, 1693
setup_simple
L4virtio::Virtqueue, 1738
sh_flags
Elf32_Shdr, 845
Elf64_Shdr, 856
sh_type
Elf32_Shdr, 845
Elf64_Shdr, 856
Shared
L4::Kip::Mem_desc, 1087
Shared Memory Library, 476
l4shmc_area_overhead, 477
l4shmc_area_size, 477
l4shmc_area_size_free, 478
l4shmc_attach, 478
l4shmc_attach_to, 479
l4shmc_chunk_overhead, 479
l4shmc_connect_chunk_signal, 479
l4shmc_create, 481
Shared_cap
L4Re, 638
L4Re::Util, 653
shared_cap
L4Re, 638
L4Re::Util, 655
Shared_del_cap
L4Re, 639
L4Re::Util, 656
shared_del_cap
L4Re, 639
L4Re::Util, 656
SHF_ALLOC
ELF binary format, 208
SHF_ARM_COMDEF
ELF binary format, 207

SHF_ARM_ENTRYSECT
ELF binary format, 207
SHF_EXECINSTR
ELF binary format, 208
SHF_GROUP
ELF binary format, 208
SHF_INFO_LINK
ELF binary format, 208
SHF_LINK_ORDER
ELF binary format, 208
SHF_MASKOS
ELF binary format, 208
SHF_MASKPROC
ELF binary format, 208
SHF_MERGE
ELF binary format, 208
SHF_OS_NONCONFORMING
ELF binary format, 208
SHF_STRINGS
ELF binary format, 208
SHF_TLS
ELF binary format, 208
SHF_WRITE
ELF binary format, 208
shift
L4Re::Video::Color_component, 1493
Shift_type
cxx::Bitfield< T, LSB, MSB >, 699
SHN_ABS
ELF binary format, 208
SHN_COMMON
ELF binary format, 208
SHN_HIPROC
ELF binary format, 208
SHN_HIRESERVE
ELF binary format, 208
SHN_LOPROC
ELF binary format, 208
SHN_LORESERVE
ELF binary format, 208
SHN_UNDEF
ELF binary format, 208
SHT_DYNAMIC
ELF binary format, 209
SHT_DYNSYM
ELF binary format, 209
SHT_FINI_ARRAY
ELF binary format, 209
SHT_GROUP
ELF binary format, 209
SHT_HASH
ELF binary format, 209
SHT_HIOS
ELF binary format, 209
SHT_HIPROC
ELF binary format, 209
SHT_HIUSER
ELF binary format, 209
SHT_INIT_ARRAY
ELF binary format, 209
SHT_LOOS
ELF binary format, 209
SHT_LOPROC
ELF binary format, 209
SHT_LOUSER
ELF binary format, 209
SHT_NOBITS
ELF binary format, 209
SHT_NOTE
ELF binary format, 209
SHT_NULL
ELF binary format, 209
SHT_NUM
ELF binary format, 209
SHT_PREINIT_ARRAY
ELF binary format, 209
SHT_PROGBITS
ELF binary format, 209
SHT_REL
ELF binary format, 209
SHT_RELA
ELF binary format, 209
SHT_SHLIB
ELF binary format, 209
SHT_STRTAB
ELF binary format, 209
SHT_SYMTAB
ELF binary format, 209
SHT_SYMTAB_SHNDX
ELF binary format, 209
si
l4_vcpu_regs_t, 1284
Sigma0 API, 481
l4sigma0_debug_dump, 483
L4SIGMA0_IPCERROR, 483
l4sigma0_map_anypage, 483
l4sigma0_map_errstr, 484
l4sigma0_map_iomem, 484
l4sigma0_map_kip, 485
l4sigma0_map_mem, 485
l4sigma0_new_client, 486
L4SIGMA0_NOFPAGE, 483
L4SIGMA0_NOTALIGNED, 483
L4SIGMA0_OK, 483
l4sigma0_return_flags_t, 483
L4SIGMA0_SMALLERFPAGE, 483
signal
L4Re::Parent, 1367
Signals, 487
l4shmc_add_signal, 488
l4shmc_attach_signal, 488
l4shmc_attach_signal_to, 489
l4shmc_check_magic, 489
l4shmc_get_signal_to, 490
l4shmc_signal_cap, 490
size

L4::Kip::Mem_desc, 1093
L4Re::Dataspace, 1309
L4Re::Video::Color_component, 1494
L4virtio::Svr::Driver_mem_region_t< DATA >, 1710
skip
L4::Ipc::Istream, 977
L4virtio::Svr::Data_buffer, 1671
slab_size
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 687
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 695
Small C++ Template Library, 491
max, 492
min, 492
operator new, 493
Small_buf
L4::Ipc::Small_buf, 1011, 1012
Smart_cap
L4::Smart_cap< T, SMART >, 1171
snd_base
L4::Cap_base, 881
Space_attrib
L4Re::Dma_space, 1317
Split_ds
L4Re::Rm, 1375
Src_dev_handle
L4vbus::Icu, 1574
Src_types
L4vbus::Icu, 1574
ss
l4_exc_regs_t, 1260
stack
L4Re::Video::View, 1516
start
L4::Kip::Mem_desc, 1093
L4virtio::Svr::Request_processor, 1715, 1716
start_request
L4virtio::Driver::Block_device, 1626
starts_with
cxx::String, 830
State
L4vcpu::State, 1599
state
L4vcpu::Vcpu, 1610, 1611
stats_time
L4::Thread, 1188
status
L4virtio::Svr::Dev_config, 1682
l4virtio_config_hdr_t, 1757
STB_GLOBAL
ELF binary format, 209
STB_HIOS
ELF binary format, 209
STB_HIPROC
ELF binary format, 209
STB_LOCAL
ELF binary format, 209
STB_LOOS
ELF binary format, 209
STB_LOPROC
ELF binary format, 209
STB_WEAK
ELF binary format, 209
Str_cp_in
L4::Ipc::Str_cp_in< T >, 1014
str_cp_in
L4::Ipc, 627
String
cxx::String, 827
Strong
L4Re::Namespace, 1359
STT_FILE
ELF binary format, 210
STT_FUNC
ELF binary format, 210
STT_HIOS
ELF binary format, 210
STT_HIPROC
ELF binary format, 210
STT_LOOS
ELF binary format, 210
STT_LOPROC
ELF binary format, 210
STT_NOTYPE
ELF binary format, 210
STT_OBJECT
ELF binary format, 210
STT_SECTION
ELF binary format, 210
sub_type
L4::Kip::Mem_desc, 1093
Super_pages
L4Re::Mem_alloc, 1350
supports
L4::Meta, 1115
switch_log
L4::Debugger, 892
switch_to
L4::Thread, 1188
symlink
L4Re::Vfs::Directory, 1464
system_shutdown
L4::Platform_control, 1126
system_suspend
L4::Platform_control, 1126

tag
L4::Ipc::Istream, 977
L4::Ipc::Ostream, 1007
L4::Ipc::Varg, 1019
take
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, 1407
L4Re::Util::Dataspace_svr, 1413
Task, 494

L4_FP_ALL_SPACES, 496
L4_FP_DELETE_OBJ, 496
L4_FP_OTHER_SPACES, 496
l4_task_add_ku_mem, 496
l4_task_cap_equal, 496
l4_task_cap_valid, 497
l4_task_delete_obj, 497
l4_task_map, 498
l4_task_release_cap, 499
l4_task_unmap, 499
l4_task_unmap_batch, 500
l4_task_vgicc_map, 501
l4_unmap_flags_t, 495
task
L4Re::Env, 1331
L4vcpu::Vcpu, 1611
test
L4::Poll_timeout_kipclock, 1129
Thread, 502
l4_thread_arm_set_tpidruro, 505
L4_THREAD_CONTROL_ALIEN, 504
L4_THREAD_CONTROL_BIND_TASK, 504
L4_thread_control_flags, 504
L4_THREAD_CONTROL_MR_IDX_BIND_TASK, 505
L4_THREAD_CONTROL_MR_IDX_BIND_UTCB, 505
L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER, 505
L4_THREAD_CONTROL_MR_IDX_FLAG_VALS, 505
L4_THREAD_CONTROL_MR_IDX_FLAGS, 505
L4_THREAD_CONTROL_MR_IDX_PAGER, 505
L4_thread_control_mr_indices, 505
L4_THREAD_CONTROL_SET_EXC_HANDLER, 504
L4_THREAD_CONTROL_SET_PAGER, 504
L4_THREAD_CONTROL_UX_NATIVE, 504
l4_thread_ex_regs, 507
L4_THREAD_EX_REGS_CANCEL, 505
L4_thread_ex_regs_flags, 505
l4_thread_ex_regs_ret, 508
l4_thread_ex_regs_ret_u, 509
L4_THREAD_EX_REGS_TRIGGER_EXCEPTION, 505
l4_thread_ex_regs_u, 510
l4_thread_modify_sender_add, 511
l4_thread_modify_sender_commit, 512
l4_thread_modify_sender_start, 512
l4_thread_register_del_irq, 512
l4_thread_stats_time, 513
l4_thread_switch, 514
l4_thread_vcpu_control, 514
l4_thread_vcpu_control_ext, 515
l4_thread_vcpu_control_ext_u, 516
l4_thread_vcpu_control_u, 517
l4_thread_vcpu_resume_commit, 518
l4_thread_vcpu_resume_start, 519
l4_thread_yield, 519
Thread control, 520
l4_thread_control_alien, 521
l4_thread_control_bind, 522
l4_thread_control_commit, 523
l4_thread_control_exc_handler, 523
l4_thread_control_pager, 524
l4_thread_control_start, 524
l4_thread_control_ux_host_syscall, 525
Thread Control Registers (TCRs), 520
thread.h
__L4UTIL_THREAD_FUNC, 2372
throw_error
L4Re, 650
throw_ipc_exception
L4, 618, 619
timed_out
L4::Poll_timeout_kipclock, 1130
timeout
L4::Ipc_svr::Timeout, 1053
timeout_expired
L4::Ipc_svr::Timeout_queue, 1059
Timeouts, 525
l4_ipc_timeout, 527
L4_IPC_TIMEOUT_0, 527
l4_rcv_timeout, 528
l4_snd_timeout, 528
l4_timeout, 528
l4_timeout_abs, 529
l4_timeout_get, 530
l4_timeout_is_absolute, 531
l4_timeout_rel, 531
l4_timeout_rel_get, 532
l4_timeout_s, 527
l4_timeout_t, 527
l4_utcb_mr64_idx, 532
Timestamp Counter, 533
l4_busy_wait_ns, 534
l4_busy_wait_us, 535
l4_calibrate_tsc, 535
l4_get_hz, 536
l4_ns_to_tsc, 536
l4_rdpmc, 537
l4_rdpmc_32, 537
l4_rdtsc, 538
l4_rdtsc_32, 538
l4_tsc_init, 539
l4_tsc_to_ns, 539
l4_tsc_to_s_and_ns, 540
l4_tsc_to_us, 540
To_device
L4Re::Dma_space, 1316
to_irq
L4vbus::Gpio_pin, 1571
total_objects
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 690

cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 697
total_size
L4virtio::Virtqueue, 1739, 1740
trigger
L4::Triggerable, 1200
trunc_order
L4, 619
Trusted
L4Re::Namespace, 1359
type
L4::Ipc::Varg, 1019
L4::Kip::Mem_desc, 1094
L4Re::Vfs::Be_file_system, 1459
L4Re::Vfs::File_system, 1469
types
L4_TYPES_FLAGS_OPS_DEF, 2169
types.h
l4_capability_next, 2279

unbind
L4::Icu, 931
L4::Iommu, 942
Uncacheable
L4Re::Dataspace::F, 1310
Undefined
L4::Kip::Mem_desc, 1087
Unique_cap
L4Re, 640
L4Re::Util, 657
unique_cap
L4Re, 640
L4Re::Util, 658
Unique_del_cap
L4Re, 640
L4Re::Util, 658
unique_del_cap
L4Re, 641
L4Re::Util, 659
unlink
L4Re::Namespace, 1361
L4Re::Vfs::Directory, 1464
unlock_all_locks
L4Re::Vfs::Be_file, 1456
L4Re::Vfs::Generic_file, 1478
unmap
L4::Task, 1178
L4Re::Dma_space, 1318
unmap_batch
L4::Task, 1179
unmask
L4::Irq, 1070
L4::Irq_eoi, 1073
unregister_obj
L4::Registry_iface, 1138
L4Re::Util::Object_registry, 1438
up
L4::Semaphore, 1151
used_align
L4virtio::Virtqueue, 1741
Used_elem
L4virtio::Virtqueue::Used_elem, 1752
used_size
L4virtio::Virtqueue, 1741
utcb_area
L4Re::Env, 1331
util.h
l4_sleep, 1801, 1804
l4util_micros2l4to, 1802, 1805
Utility Functions, 541
l4_sleep, 542
l4_touch_ro, 544
l4_touch_rw, 544
l4_usleep, 545
l4util_micros2l4to, 546
l4util_splitlog2_hdl, 546
l4util_splitlog2_size, 547
ux_host_syscall
L4::Thread::Attr, 1195

V_flags
L4Re::Video::View, 1514
val
cxx::Bitfield< T, LSB, MSB >, 705
val_dirty
cxx::Bitfield< T, LSB, MSB >, 707
val_unshifted
cxx::Bitfield< T, LSB, MSB >, 708
valid
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, 745
L4virtio::Svr::Virtqueue::Head_desc, 1725
validate
L4::Cap_base, 882
value
L4::Ipc::Varg, 1019
Varg_list_ref
L4::Ipc::Varg_list_ref, 1023
vbe_ctrl_info
l4util_l4mod_info, 1536
Vbus API, 563
vbus.h
L4VBUS_ICU_SRC_DEV_HANDLE, 2385
l4vbus_icu_src_types, 2385
L4VBUS_NULL, 2385
L4VBUS_ROOT_BUS, 2385
vbus_interfaces.h
L4VBUS_IFACE_SHIFT, 2388
l4vbus_iface_type_t, 2388, 2389
l4vbus_subinterface_supported, 2389
vbus_types.h
L4VBUS_DEVICE_F_CHILDREN, 2392
l4vbus_device_flags_t, 2391
L4VBUS_RESOURCE_BUS, 2392
L4VBUS_RESOURCE_DMA_DOMAIN, 2392
L4VBUS_RESOURCE_F_MEM_MMIO_READ, 2392

L4VBUS_RESOURCE_F_MEM_MMIO_WRITE, 2392
L4VBUS_RESOURCE_F_MEM_READ_ONLY, 2392
l4vbus_resource_flags_t, 2392
L4VBUS_RESOURCE_GPIO, 2392
L4VBUS_RESOURCE_INVALID, 2392
L4VBUS_RESOURCE_IRQ, 2392
L4VBUS_RESOURCE_MAX, 2392
L4VBUS_RESOURCE_MEM, 2392
L4VBUS_RESOURCE_PORT, 2392
l4vbus_resource_type_t, 2392
vcon.h
L4_vcon_read_flags, 2299
L4_VCON_READ_SIZE_MASK, 2299
L4_VCON_READ_STAT_BREAK, 2299
L4_VCON_READ_STAT_DONE, 2299
vCPU API, 592
L4_VCPU_F_EXCEPTIONS, 595
L4_VCPU_F_FPU_ENABLED, 595
L4_VCPU_F_IRQ, 595
L4_VCPU_F_PAGE_FAULTS, 595
L4_VCPU_F_USER_MODE, 595
L4_VCPU_OFFSET_EXT_INFOS, 596
L4_VCPU_OFFSET_EXT_STATE, 596
L4_VCPU_SF_IRQ_PENDING, 596
L4_vcpu_state_flags, 594
L4_vcpu_state_offset, 595
L4_vcpu_sticky_flags, 596
vCPU Support Library, 596
l4vcpu_irq_disable, 597
l4vcpu_irq_disable_save, 598
l4vcpu_irq_enable, 599
l4vcpu_irq_restore, 600
l4vcpu_is_irq_entry, 601
l4vcpu_is_page_fault_entry, 602
l4vcpu_print_state, 602
l4vcpu_wait_for_event, 603
vcpu_control
L4::Thread, 1189
vcpu_control_ext
L4::Thread, 1190
vcpu_resume_commit
L4::Thread, 1190
vcpu_resume_start
L4::Thread, 1191
vicu
L4vbus::Icu, 1574
Video API, 565
F_l4re_video_goos_auto_refresh, 567
F_l4re_video_goos_dynamic_buffers, 567
F_l4re_video_goos_dynamic_views, 567
F_l4re_video_goos_pointer, 567
F_l4re_video_view_above, 567
F_l4re_video_view_dyn_allocated, 567
F_l4re_video_view_flags_mask, 567
F_l4re_video_view_none, 567
F_l4re_video_view_set_background, 567
F_l4re_video_view_set_buffer, 567
F_l4re_video_view_set_buffer_offset, 567
F_l4re_video_view_set_bytes_per_line, 567
F_l4re_video_view_set_flags, 567
F_l4re_video_view_set_pixel, 567
F_l4re_video_view_set_position, 567
l4re_video_goos_create_buffer, 567
l4re_video_goos_create_view, 568
l4re_video_goos_delete_buffer, 568
l4re_video_goos_delete_view, 570
l4re_video_goos_get_static_buffer, 570
l4re_video_goos_get_view, 570
l4re_video_goos_info, 571
l4re_video_goos_info_flags_t, 567
l4re_video_goos_refresh, 571
l4re_video_view_get_info, 572
l4re_video_view_info_flags_t, 567
l4re_video_view_refresh, 572
l4re_video_view_set_info, 572
l4re_video_view_set_viewport, 573
l4re_video_view_stack, 573
l4re_video_view_t, 566
view
L4Re::Video::Goos, 1501
view_info
L4Re::Util::Video::Goos_svr, 1452
Virtual Console, 574
L4_VCON_ECHO, 576
l4_vcon_get_attr, 577
l4_vcon_get_attr_u, 578
L4_vcon_i_flags, 576
L4_VCON_ICANON, 576
L4_VCON_ICRNL, 576
L4_VCON_IGNCR, 576
L4_VCON_INLCR, 576
L4_vcon_l_flags, 576
L4_vcon_o_flags, 576
L4_VCON_OCRNL, 576
L4_VCON_ONLCR, 576
L4_VCON_ONLRET, 576
l4_vcon_read, 579
L4_VCON_READ_SIZE, 577
l4_vcon_read_u, 579
l4_vcon_read_with_flags, 580
l4_vcon_send, 581
l4_vcon_send_u, 582
l4_vcon_set_attr, 583
l4_vcon_set_attr_raw, 584
l4_vcon_set_attr_u, 584
L4_vcon_size_consts, 577
l4_vcon_write, 585
L4_VCON_WRITE_SIZE, 577
l4_vcon_write_u, 586
Virtual Machines, 587
Virtual Registers (UTCBs), 588
l4_utcb_br, 590
l4_utcb_mr, 590
l4_utcb_t, 590

l4_utcb_tcr, 591
VM API for SVM, 548
VM API for TZ, 549
VM API for VMX, 549
L4_VM_VMX_BASIC_REG, 552
L4_vm_vmx_caps_regs, 551
l4_vm_vmx_clear, 552
L4_VM_VMX_CR0_FIXED0_REG, 552
L4_VM_VMX_CR0_FIXED1_REG, 552
L4_VM_VMX_CR4_FIXED0_REG, 552
L4_VM_VMX_CR4_FIXED1_REG, 552
L4_vm_vmx_dfl1_regs, 552
L4_VM_VMX_ENTRY_CTLS_DFL1_REG, 552
L4_VM_VMX_EPT_VPID_CAP_REG, 552
L4_VM_VMX_EXIT_CTLS_DFL1_REG, 552
l4_vm_vmx_field_len, 553
l4_vm_vmx_field_order, 554
l4_vm_vmx_get_caps, 554
l4_vm_vmx_get_caps_default1, 555
l4_vm_vmx_get_cr2_index, 555
L4_VM_VMX_MISC_REG, 552
L4_VM_VMX_NUM_CAPS_REGS, 552
L4_VM_VMX_NUM_DFL1_REGS, 552
L4_VM_VMX_PINBASED_CTLS_DFL1_REG, 552
L4_VM_VMX_PROCBASED_CTLS2_REG, 552
L4_VM_VMX_PROCBASED_CTLS_DFL1_REG, 552
l4_vm_vmx_ptr_load, 556
l4_vm_vmx_read, 556
l4_vm_vmx_read_16, 557
l4_vm_vmx_read_32, 558
l4_vm_vmx_read_64, 558
l4_vm_vmx_read_nat, 559
L4_VM_VMX_TRUE_ENTRY_CTLS_REG, 552
L4_VM_VMX_TRUE_EXIT_CTLS_REG, 552
L4_VM_VMX_TRUE_PINBASED_CTLS_REG, 552
L4_VM_VMX_TRUE_PROCBASED_CTLS_REG, 552
L4_VM_VMX_VMCS_CR2, 551
L4_VM_VMX_VMCS_ENUM_REG, 552
L4_VM_VMX_VMCS_MSR_CSTAR, 551
L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE, 551
L4_VM_VMX_VMCS_MSR_LSTAR, 551
L4_VM_VMX_VMCS_MSR_STAR, 551
L4_VM_VMX_VMCS_MSR_SYSCALL_MASK, 551
L4_VM_VMX_VMCS_MSR_TSC_AUX, 551
L4_VM_VMX_VMCS_XCR0, 551
l4_vm_vmx_write, 560
l4_vm_vmx_write_16, 561
l4_vm_vmx_write_32, 562
l4_vm_vmx_write_64, 562
l4_vm_vmx_write_nat, 563

W
L4Re::Dataspace::F, 1310
L4Re::Rm::F, 1385
W_bits
cxx::Bitmap_base, 719
wait
L4::Ipc::Istream, 978, 979
L4::Irq, 1070
L4virtio::Driver::Device, 1639
wait_for_event
L4vcpu::Vcpu, 1611
wait_for_next_used
L4virtio::Driver::Device, 1640
Wd_16bit
L4Re::Mmio_space, 1354
Wd_32bit
L4Re::Mmio_space, 1354
Wd_64bit
L4Re::Mmio_space, 1354
Wd_8bit
L4Re::Mmio_space, 1354
Word_bytes
L4::Ipc::Msg, 629
word_index
cxx::Bitmap_base, 724
write
L4::Vcon, 1253
write_bfm_t
L4virtio::Virtqueue::Desc::Flags, 1749
writev
L4Re::Vfs::Regular_file, 1487

X
L4Re::Rm::F, 1385
x86 Virtual Registers (UTCB), 604
L4_UTCB_BUF_REGS_OFFSET, 605
L4_utcb_consts_x86, 604
L4_UTCB_EXCEPTION_REGS_SIZE, 605
L4_UTCB_GENERIC_BUFFERS_SIZE, 605
L4_UTCB_GENERIC_DATA_SIZE, 605
L4_UTCB_INHERIT_FPU, 605
L4_UTCB_MSG_REGS_OFFSET, 605
L4_UTCB_OFFSET, 605
L4_UTCB_THREAD_REGS_OFFSET, 605
x86/l4/sys/cache.h, 2120, 2121
x86/l4/sys/consts.h, 2135
x86/l4/sys/l4int.h, 2228, 2229
x86/l4/sys/linkage.h, 1829
x86/l4/sys/segment.h, 1815, 1817
x86/l4/sys/utcb.h, 2292, 2293
x86/l4/util/apic.h, 1768, 1769
x86/l4/util/bitops_arch.h, 1837, 1838
x86/l4/util/cpu.h, 1843, 1845
x86/l4/util/idt.h, 1776, 1777
x86/l4/util/irq.h, 1935, 1937
x86/l4/util/l4_macros.h, 1848
x86/l4/util/mbi_argv.h, 1851, 1852
x86/l4/util/perform.h, 1783, 1784
x86/l4/util/port_io.h, 1823, 1825
x86/l4/util/rdtsc.h, 1793, 1795
x86/l4/util/spin.h, 1799, 1800
x86/l4/util/util.h, 1803, 1805

x86/l4f/l4/sys/ipc.h, 2208, 2209
x86/l4f/l4/sys/segment.h, 1814, 1815
x86/l4f/l4/util/port_io.h, 1820, 1822