L4Re – L4 Runtime Environment

Generated by Doxygen 1.8.13

Contents

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 9

4.2 Initial Environment and Application Bootstrapping 10

4.2.1 Configuring an application before startup 11

4.2.2 Connecting clients and servers 11

4.3 Memory management - Data Spaces and the Region Map 12

4.3.1 User-level paging 12

4.3.1.1 Data spaces 12

4.3.1.2 Virtual Memory Handling 12

4.3.1.3 Memory Allocation 13

4.4 Program Input and Output 13

4.5 Initial Memory Allocator and Factory 13

4.6 Application and Server Building Blocks 13

4.6.1 Creating Additional Application Threads . . . 14
4.6.2 Providing a Service . . . 14
4.7 Pthread Support . . . 14
4.8 Interface Definition Language . . . 15
4.8.1 Parameter types for RPC . . . 15
4.8.2 RPC Return Types . . . 17
4.8.3 RPC Method Declaration . . . 18
4.9 L4Re Build System . . . 18
4.9.1 Building L4Re . . . 19
4.9.2 Writing BID Make Files . . . 20
4.9.3 prog.mk - Application Role . . . 20
4.9.4 include.mk - Header File Role . . . 23
4.9.5 test.mk - Test Application Role . . . 24

5 L4Re Servers **29**

5.1 Moe, the Root-Task . . . 31
5.1.1 Memory Allocator, Generic Factory . . . 31
5.1.2 Name-Space Provider . . . 32
5.1.2.1 Boot FS . . . 32
5.1.3 Log Subsystem . . . 32
5.1.4 Scheduler Subsystem . . . 32
5.1.5 Command-Line Options . . . 32
5.2 Ned, the Init Process . . . 33
5.2.1 Lua Bindings for L4Re . . . 33
5.2.1.1 Capabilities in Lua . . . 33
5.2.1.2 Access to L4Re::Env Capabilities . . . 33
5.2.1.3 Constants . . . 34
5.2.1.4 Application Startup Details . . . 35
5.2.2 Command Line Options . . . 35
5.3 Io, the Io Server . . . 36
5.4 Uvmm, the virtual machine monitor . . . 41
5.5 Mag, the GUI Multiplexer . . . 41
5.6 Cons, the Console Multiplexer . . . 43

6 Deprecated List **45**

7 Module Index **47**

7.1 Modules 47

8 Namespace Index **51**

8.1 Namespace List 51

9 Hierarchical Index **53**

9.1 Class Hierarchy 53

10 Data Structure Index **63**

10.1 Data Structures 63

11 File Index **77**

11.1 File List 77

12 Module Documentation **89**

12.1 ARM Virtual Registers (UTCB) 89

12.1.1 Detailed Description 89

12.2 Atomic Instructions 90

12.2.1 Detailed Description 92

12.2.2 Function Documentation 92

12.2.2.1 l4util_add8() 92

12.2.2.2 l4util_add8_res() 92

12.2.2.3 l4util_atomic_add() 92

12.2.2.4 l4util_atomic_inc() 93

12.2.2.5 l4util_cmpxchg() 93

12.2.2.6 l4util_cmpxchg16() 94

12.2.2.7 l4util_cmpxchg32() 95

12.2.2.8 l4util_cmpxchg64() 96

12.2.2.9 l4util_cmpxchg8() 96

12.2.2.10 l4util_inc8() 97

12.2.2.11 l4util_inc8_res() 97

12.2.2.12 l4util_xchg() . . . 98
12.2.2.13 l4util_xchg16() . . . 98
12.2.2.14 l4util_xchg32() . . . 98
12.2.2.15 l4util_xchg8() . . . 99
12.3 Auxiliary data . . . 100
12.3.1 Detailed Description . . . 100
12.4 Base API . . . 101
12.4.1 Detailed Description . . . 103
12.5 Basic Macros . . . 104
12.5.1 Detailed Description . . . 105
12.5.2 Macro Definition Documentation . . . 105
12.5.2.1 L4_ALWAYS_INLINE . . . 105
12.5.2.2 L4_HIDDEN . . . 105
12.5.2.3 L4_NOTHROW . . . 106
12.6 Bit Manipulation . . . 107
12.6.1 Detailed Description . . . 107
12.6.2 Function Documentation . . . 107
12.6.2.1 l4util_bsf() . . . 108
12.6.2.2 l4util_bsr() . . . 108
12.6.2.3 l4util_btc() . . . 109
12.6.2.4 l4util_btr() . . . 109
12.6.2.5 l4util_bts() . . . 110
12.6.2.6 l4util_clear_bit() . . . 111
12.6.2.7 l4util_complement_bit() . . . 111
12.6.2.8 l4util_find_first_set_bit() . . . 112
12.6.2.9 l4util_find_first_zero_bit() . . . 112
12.6.2.10 l4util_next_power2() . . . 113
12.6.2.11 l4util_set_bit() . . . 113
12.6.2.12 l4util_test_bit() . . . 114
12.7 Bitmap graphics and fonts . . . 115

12.7.1 Detailed Description . . . 115
12.8 Buffer Registers (BRs) . . . 116
12.8.1 Detailed Description . . . 116
12.8.2 Enumeration Type Documentation . . . 116
12.8.2.1 l4_buffer_desc_consts_t . . . 116
12.9 C++ Exceptions . . . 118
12.9.1 Detailed Description . . . 118
12.10 C++ IPC Interface Definition. . . . 119
12.10.1 Detailed Description . . . 119
12.11 CPU related functions . . . 120
12.11.1 Detailed Description . . . 120
12.11.2 Function Documentation . . . 120
12.11.2.1 l4util_cpu_capabilities() . . . 120
12.11.2.2 l4util_cpu_capabilities_nocheck() . . . 121
12.11.2.3 l4util_cpu_has_cpuid() . . . 122
12.12 Cache Consistency . . . 123
12.12.1 Detailed Description . . . 123
12.12.2 Function Documentation . . . 123
12.12.2.1 l4_cache_clean_data() . . . 123
12.12.2.2 l4_cache_coherent() . . . 124
12.12.2.3 l4_cache_dma_coherent() . . . 124
12.12.2.4 l4_cache_flush_data() . . . 125
12.12.2.5 l4_cache_inv_data() . . . 125
12.13 Capabilities . . . 127
12.13.1 Detailed Description . . . 127
12.13.2 Enumeration Type Documentation . . . 128
12.13.2.1 l4_cap_consts_t . . . 128
12.13.2.2 l4_default_caps_t . . . 128
12.13.3 Function Documentation . . . 129
12.13.3.1 l4_capability_equal() . . . 129

12.13.3.2 l4_is_invalid_cap() . . . 129
12.13.3.3 l4_is_valid_cap() . . . 130
12.14 Capability allocator . . . 131
12.14.1 Detailed Description . . . 131
12.14.2 Function Documentation . . . 131
12.14.2.1 l4re_util_cap_last() . . . 131
12.15 Chunks . . . 132
12.15.1 Detailed Description . . . 132
12.15.2 Function Documentation . . . 132
12.15.2.1 l4shmc_add_chunk() . . . 132
12.15.2.2 l4shmc_chunk_capacity() . . . 133
12.15.2.3 l4shmc_chunk_ptr() . . . 133
12.15.2.4 l4shmc_chunk_signal() . . . 134
12.15.2.5 l4shmc_get_chunk() . . . 134
12.15.2.6 l4shmc_get_chunk_to() . . . 135
12.15.2.7 l4shmc_iterate_chunk() . . . 135
12.16 Comfortable Command Line Parsing . . . 136
12.16.1 Detailed Description . . . 136
12.16.2 Function Documentation . . . 136
12.16.2.1 parse_cmdline() . . . 137
12.17 Console API . . . 139
12.17.1 Detailed Description . . . 139
12.18 Consumer . . . 140
12.18.1 Detailed Description . . . 140
12.18.2 Function Documentation . . . 140
12.18.2.1 l4shmc_chunk_consumed() . . . 140
12.18.2.2 l4shmc_chunk_size() . . . 141
12.18.2.3 l4shmc_enable_chunk() . . . 141
12.18.2.4 l4shmc_is_chunk_ready() . . . 142
12.18.2.5 l4shmc_wait_chunk() . . . 142

12.18.2.6 l4shmc_wait_chunk_to() 142
12.18.2.7 l4shmc_wait_chunk_try() 143
12.19 Consumer 144
12.19.1 Detailed Description 144
12.19.2 Function Documentation 144
12.19.2.1 l4shmc_enable_signal() 144
12.19.2.2 l4shmc_wait_any() 145
12.19.2.3 l4shmc_wait_any_to() 145
12.19.2.4 l4shmc_wait_any_try() 146
12.19.2.5 l4shmc_wait_signal() 146
12.19.2.6 l4shmc_wait_signal_to() 146
12.19.2.7 l4shmc_wait_signal_try() 147
12.20 DMA API for L4Re 148
12.21 DMA Space Interface 149
12.21.1 Detailed Description 149
12.21.2 Typedef Documentation 149
12.21.2.1 l4re_dma_space_t 150
12.21.3 Function Documentation 150
12.21.3.1 l4re_dma_space_associate() 150
12.21.3.2 l4re_dma_space_disassociate() 151
12.21.3.3 l4re_dma_space_map() 151
12.21.3.4 l4re_dma_space_unmap() 152
12.22 Dataspace interface 153
12.22.1 Detailed Description 154
12.22.2 Enumeration Type Documentation 154
12.22.2.1 l4re_ds_map_flags 154
12.22.3 Function Documentation 154
12.22.3.1 l4re_ds_allocate() 154
12.22.3.2 l4re_ds_clear() 155
12.22.3.3 l4re_ds_copy_in() 155

12.22.3.4 l4re_ds_flags() . . . 155
12.22.3.5 l4re_ds_info() . . . 156
12.22.3.6 l4re_ds_phys() . . . 156
12.22.3.7 l4re_ds_size() . . . 156
12.23 Debug interface . . . 158
12.23.1 Detailed Description . . . 158
12.23.2 Function Documentation . . . 158
12.23.2.1 l4re_debug_obj_debug() . . . 158
12.24 Debugging API . . . 159
12.24.1 Detailed Description . . . 159
12.25 EDID parsing functionality . . . 160
12.25.1 Detailed Description . . . 160
12.25.2 Enumeration Type Documentation . . . 160
12.25.2.1 Libedid_consts . . . 160
12.25.3 Function Documentation . . . 161
12.25.3.1 libedid_check_header() . . . 161
12.25.3.2 libedid_checksum() . . . 161
12.25.3.3 libedid_dump() . . . 161
12.25.3.4 libedid_dump_standard_timings() . . . 162
12.25.3.5 libedid_num_ext_blocks() . . . 162
12.25.3.6 libedid_pnp_id() . . . 162
12.25.3.7 libedid_prefered_resolution() . . . 163
12.25.3.8 libedid_revision() . . . 163
12.25.3.9 libedid_version() . . . 163
12.26 ELF binary format . . . 165
12.26.1 Detailed Description . . . 177
12.26.2 Macro Definition Documentation . . . 177
12.26.2.1 DF_1_CONFALT . . . 177
12.26.2.2 DF_1_DIRECT . . . 178
12.26.2.3 DF_1_DISPRELDNE . . . 178

12.26.2.4 DF_1_DISPRELPND 178
12.26.2.5 DF_1_ENDFILTEE 178
12.26.2.6 DF_1_GLOBAL 178
12.26.2.7 DF_1_GROUP 179
12.26.2.8 DF_1_INTERPOSE 179
12.26.2.9 DF_1_LOADFLTR 179
12.26.2.10 DF_1_NODEFLIB 179
12.26.2.11 DF_1_NODELETE 179
12.26.2.12 DF_1_NODUMP 180
12.26.2.13 DF_1_NOOPEN 180
12.26.2.14 DF_1_NOW 180
12.26.2.15 DF_1_ORIGIN 180
12.26.2.16 DF_P1_GROUPPERM 180
12.26.2.17 DF_P1_LAZYLOAD 181
12.26.2.18 DT_HIPROC 181
12.26.2.19 DT_LOPROC 181
12.26.2.20 DT_NULL 181
12.26.2.21 EI_CLASS [1/2] 181
12.26.2.22 EI_CLASS [2/2] 182
12.26.2.23 EI_DATA [1/2] 182
12.26.2.24 EI_DATA [2/2] 182
12.26.2.25 EI_OSABI [1/2] 182
12.26.2.26 EI_OSABI [2/2] 183
12.26.2.27 EI_PAD [1/2] 183
12.26.2.28 EI_PAD [2/2] 183
12.26.2.29 EI_VERSION [1/2] 183
12.26.2.30 EI_VERSION [2/2] 184
12.26.2.31 ELFCLASSNONE [1/2] 184
12.26.2.32 ELFCLASSNONE [2/2] 184
12.26.2.33 ELFDATA2LSB [1/2] 184

12.26.2.34 ELFDATA2LSB [2/2] 185
12.26.2.35 ELFDATA2MSB [1/2] 185
12.26.2.36 ELFDATA2MSB [2/2] 185
12.26.2.37 ELFDATANONE [1/2] 185
12.26.2.38 ELFDATANONE [2/2] 186
12.26.2.39 ELFOSABI_AIX 186
12.26.2.40 ELFOSABI_FREEBSD 186
12.26.2.41 ELFOSABI_HPUX [1/2] 186
12.26.2.42 ELFOSABI_HPUX [2/2] 186
12.26.2.43 ELFOSABI_IRIX 187
12.26.2.44 ELFOSABI_LINUX 187
12.26.2.45 ELFOSABI_MODESTO 187
12.26.2.46 ELFOSABI_NETBSD 187
12.26.2.47 ELFOSABI_OPENBSD 187
12.26.2.48 ELFOSABI_SOLARIS 188
12.26.2.49 ELFOSABI_SYSV [1/2] 188
12.26.2.50 ELFOSABI_SYSV [2/2] 188
12.26.2.51 ELFOSABI_TRU64 188
12.26.2.52 EM_ARC 188
12.26.2.53 NT_VERSION 189
12.26.2.54 PT_GNU_EH_FRAME 189
12.26.2.55 PT_GNU_RELRO 189
12.26.2.56 PT_GNU_STACK 189
12.26.2.57 PT_HIOS 189
12.26.2.58 PT_HIPROC 190
12.26.2.59 PT_L4_AUX 190
12.26.2.60 PT_L4_KIP 190
12.26.2.61 PT_L4_STACK 190
12.26.2.62 PT_LOOS 190
12.26.2.63 PT_LOPROC 191

12.26.2.64 SHF_GROUP . . . 191
12.26.2.65 SHF_MASKOS . . . 191
12.26.2.66 SHF_TLS . . . 191
12.26.2.67 SHT_NUM . . . 191
12.27 Error Handling . . . 192
12.27.1 Detailed Description . . . 192
12.27.2 Enumeration Type Documentation . . . 193
12.27.2.1 l4_ipc_tcr_error_t . . . 193
12.27.3 Function Documentation . . . 193
12.27.3.1 l4_error() . . . 193
12.27.3.2 l4_ipc_error() . . . 195
12.27.3.3 l4_ipc_error_code() . . . 196
12.27.3.4 l4_ipc_is_rcv_error() . . . 197
12.27.3.5 l4_ipc_is_snd_error() . . . 197
12.28 Error codes . . . 199
12.28.1 Detailed Description . . . 199
12.28.2 Enumeration Type Documentation . . . 199
12.28.2.1 l4_error_code_t . . . 199
12.29 Event API . . . 201
12.29.1 Detailed Description . . . 201
12.30 Event interface . . . 202
12.30.1 Detailed Description . . . 202
12.30.2 Function Documentation . . . 202
12.30.2.1 l4re_event_get_axis_info() . . . 202
12.30.2.2 l4re_event_get_buffer() . . . 203
12.30.2.3 l4re_event_get_num_streams() . . . 203
12.30.2.4 l4re_event_get_stream_info() . . . 204
12.30.2.5 l4re_event_get_stream_info_for_id() . . . 204
12.31 Exception registers . . . 206
12.31.1 Detailed Description . . . 206

12.31.2 Function Documentation . . . 206
12.31.2.1 l4_utcb_exc() . . . 206
12.31.2.2 l4_utcb_exc_is_ex_regs_exception() . . . 207
12.31.2.3 l4_utcb_exc_is_pf() . . . 207
12.31.2.4 l4_utcb_exc_pc() . . . 207
12.31.2.5 l4_utcb_exc_pc_set() . . . 208
12.32 Extended vCPU support . . . 209
12.32.1 Detailed Description . . . 209
12.32.2 Function Documentation . . . 209
12.32.2.1 l4vcpu_ext_alloc() . . . 209
12.33 Factory . . . 211
12.33.1 Detailed Description . . . 212
12.33.2 Function Documentation . . . 212
12.33.2.1 l4_factory_create_factory() . . . 212
12.33.2.2 l4_factory_create_factory_u() . . . 213
12.33.2.3 l4_factory_create_gate() . . . 214
12.33.2.4 l4_factory_create_gate_u() . . . 215
12.33.2.5 l4_factory_create_irq() . . . 216
12.33.2.6 l4_factory_create_irq_u() . . . 217
12.33.2.7 l4_factory_create_task() . . . 218
12.33.2.8 l4_factory_create_task_u() . . . 219
12.33.2.9 l4_factory_create_thread() . . . 220
12.33.2.10 l4_factory_create_thread_u() . . . 221
12.33.2.11 l4_factory_create_vm() . . . 222
12.33.2.12 l4_factory_create_vm_u() . . . 223
12.34 Fiasco extensions . . . 224
12.34.1 Detailed Description . . . 225
12.34.2 Enumeration Type Documentation . . . 225
12.34.2.1 anonymous enum . . . 225
12.34.3 Function Documentation . . . 226

12.34.3.1 fiasco_gdt_get_entry_offset() . . . 226
12.34.3.2 fiasco_gdt_set() . . . 226
12.34.3.3 fiasco_ldt_set() . . . 227
12.34.3.4 fiasco_tbuf_get_status() . . . 227
12.34.3.5 fiasco_tbuf_get_status_phys() . . . 227
12.34.3.6 fiasco_tbuf_log() . . . 228
12.34.3.7 fiasco_tbuf_log_3val() . . . 228
12.34.3.8 fiasco_tbuf_log_binary() . . . 229
12.35 Fiasco real time scheduling extensions . . . 230
12.36 Fiasco-UX Virtual devices . . . 231
12.36.1 Detailed Description . . . 231
12.36.2 Enumeration Type Documentation . . . 231
12.36.2.1 l4_vhw_entry_type . . . 231
12.37 Flex pages . . . 233
12.37.1 Detailed Description . . . 235
12.37.2 Enumeration Type Documentation . . . 235
12.37.2.1 anonymous enum . . . 235
12.37.2.2 anonymous enum . . . 235
12.37.2.3 L4_cap_fpage_rights . . . 236
12.37.2.4 l4_fpage_cacheability_opt_t . . . 237
12.37.2.5 l4_fpage_consts . . . 237
12.37.2.6 L4_fpage_rights . . . 238
12.37.2.7 L4_obj_fpage_ctl . . . 238
12.37.3 Function Documentation . . . 239
12.37.3.1 l4_fpage() . . . 239
12.37.3.2 l4_fpage_all() . . . 239
12.37.3.3 l4_fpage_contains() . . . 240
12.37.3.4 l4_fpage_invalid() . . . 241
12.37.3.5 l4_fpage_ioport() . . . 241
12.37.3.6 l4_fpage_max_order() . . . 241

12.37.3.7 l4_fpage_memaddr() 242
12.37.3.8 l4_fpage_obj() 243
12.37.3.9 l4_fpage_page() 244
12.37.3.10 l4_fpage_rights() 244
12.37.3.11 l4_fpage_set_rights() 245
12.37.3.12 l4_fpage_size() 245
12.37.3.13 l4_fpage_type() 246
12.37.3.14 l4_iofpage() 246
12.37.3.15 l4_is_fpage_writable() 247
12.37.3.16 l4_obj_fpage() 248
12.38 Functions for rendering bitmap data in frame buffers 249
12.38.1 Detailed Description 249
12.38.2 Typedef Documentation 249
12.38.2.1 gfxbitmap_color_pix_t 250
12.38.2.2 gfxbitmap_color_t 250
12.38.3 Function Documentation 250
12.38.3.1 gfxbitmap_bmap() 250
12.38.3.2 gfxbitmap_convert_color() 251
12.38.3.3 gfxbitmap_copy() 251
12.38.3.4 gfxbitmap_fill() 251
12.38.3.5 gfxbitmap_set() 252
12.39 Functions for rendering bitmap fonts to frame buffers 253
12.39.1 Detailed Description 254
12.39.2 Enumeration Type Documentation 254
12.39.2.1 anonymous enum 254
12.39.3 Function Documentation 254
12.39.3.1 gfxbitmap_font_data() 254
12.39.3.2 gfxbitmap_font_get() 254
12.39.3.3 gfxbitmap_font_height() 255
12.39.3.4 gfxbitmap_font_init() 255

12.39.3.5 gfxbitmap_font_text() . . . 255
12.39.3.6 gfxbitmap_font_text_scale() . . . 256
12.39.3.7 gfxbitmap_font_width() . . . 257
12.40 Functions to manipulate the local IDT . . . 258
12.40.1 Detailed Description . . . 258
12.41 IA32 Port I/O API . . . 259
12.41.1 Detailed Description . . . 259
12.41.2 Function Documentation . . . 259
12.41.2.1 l4util_in16() . . . 260
12.41.2.2 l4util_in32() . . . 261
12.41.2.3 l4util_in8() . . . 261
12.41.2.4 l4util_ins16() . . . 262
12.41.2.5 l4util_ins32() . . . 262
12.41.2.6 l4util_ins8() . . . 263
12.41.2.7 l4util_out16() . . . 263
12.41.2.8 l4util_out32() . . . 263
12.41.2.9 l4util_out8() . . . 264
12.41.2.10 l4util_outs16() . . . 264
12.41.2.11 l4util_outs32() . . . 265
12.41.2.12 l4util_outs8() . . . 265
12.42 IO interface . . . 266
12.42.1 Detailed Description . . . 267
12.42.2 Typedef Documentation . . . 267
12.42.2.1 l4io_resource_t . . . 267
12.42.3 Enumeration Type Documentation . . . 267
12.42.3.1 l4io_device_types_t . . . 267
12.42.3.2 l4io_iomem_flags_t . . . 267
12.42.3.3 l4io_resource_types_t . . . 268
12.42.4 Function Documentation . . . 268
12.42.4.1 l4io_has_resource() . . . 268

12.42.4.2 l4io_lookup_device() . . . 269
12.42.4.3 l4io_lookup_resource() . . . 269
12.42.4.4 l4io_release_iomem() . . . 270
12.42.4.5 l4io_release_ioport() . . . 270
12.42.4.6 l4io_request_iomem() . . . 271
12.42.4.7 l4io_request_iomem_region() . . . 271
12.42.4.8 l4io_request_ioport() . . . 272
12.42.4.9 l4io_request_resource_iomem() . . . 272
12.42.4.10 l4io_search_iomem_region() . . . 273
12.43 IPC-Gate API . . . 274
12.43.1 Detailed Description . . . 274
12.43.2 Function Documentation . . . 275
12.43.2.1 l4_ipc_gate_bind_thread() . . . 275
12.43.2.2 l4_ipc_gate_get_infos() . . . 275
12.43.2.3 l4_rcv_ep_bind_thread() . . . 276
12.44 IRQ handling library . . . 277
12.44.1 Detailed Description . . . 277
12.45 IRQs . . . 278
12.45.1 Detailed Description . . . 279
12.45.2 Enumeration Type Documentation . . . 279
12.45.2.1 L4_irq_mode . . . 279
12.45.3 Function Documentation . . . 280
12.45.3.1 l4_irq_attach() . . . 280
12.45.3.2 l4_irq_attach_u() . . . 281
12.45.3.3 l4_irq_detach() . . . 281
12.45.3.4 l4_irq_detach_u() . . . 282
12.45.3.5 l4_irq_mux_chain() . . . 283
12.45.3.6 l4_irq_mux_chain_u() . . . 284
12.45.3.7 l4_irq_receive() . . . 285
12.45.3.8 l4_irq_receive_u() . . . 286

12.45.3.9 l4_irq_trigger() . . . 287
12.45.3.10 l4_irq_trigger_u() . . . 288
12.45.3.11 l4_irq_unmask() . . . 289
12.45.3.12 l4_irq_unmask_u() . . . 290
12.45.3.13 l4_irq_wait() . . . 290
12.45.3.14 l4_irq_wait_u() . . . 291
12.46 Initial Environment . . . 293
12.46.1 Detailed Description . . . 293
12.46.2 Function Documentation . . . 294
12.46.2.1 l4re_env() . . . 294
12.46.2.2 l4re_env_get_cap() . . . 294
12.46.2.3 l4re_env_get_cap_e() . . . 295
12.46.2.4 l4re_env_get_cap_l() . . . 296
12.46.2.5 l4re_kip() . . . 297
12.47 Integer Types . . . 298
12.47.1 Detailed Description . . . 299
12.47.2 Typedef Documentation . . . 299
12.47.2.1 l4_int16_t . . . 299
12.47.2.2 l4_int32_t . . . 300
12.47.2.3 l4_int64_t . . . 300
12.47.2.4 l4_int8_t . . . 300
12.47.2.5 l4_uint16_t . . . 300
12.47.2.6 l4_uint32_t . . . 300
12.47.2.7 l4_uint64_t . . . 300
12.47.2.8 l4_uint8_t . . . 300
12.48 Interface for asynchronous ISR handlers with a given IRQ capability. . . . 301
12.48.1 Detailed Description . . . 301
12.48.2 Function Documentation . . . 301
12.48.2.1 l4irq_request_cap() . . . 301
12.49 Interface for asynchronous ISR handlers. . . . 303

12.49.1 Detailed Description . . . 303
12.49.2 Function Documentation . . . 303
12.49.2.1 l4irq_release() . . . 303
12.49.2.2 l4irq_request() . . . 304
12.50 Interface using direct functionality. . . . 305
12.50.1 Detailed Description . . . 305
12.50.2 Function Documentation . . . 305
12.50.2.1 l4irq_attach_cap() . . . 305
12.50.2.2 l4irq_attach_cap_ft() . . . 306
12.50.2.3 l4irq_attach_thread_cap() . . . 306
12.50.2.4 l4irq_attach_thread_cap_ft() . . . 306
12.51 Interface using direct functionality. . . . 308
12.51.1 Detailed Description . . . 308
12.51.2 Function Documentation . . . 308
12.51.2.1 l4irq_attach() . . . 308
12.51.2.2 l4irq_attach_ft() . . . 309
12.51.2.3 l4irq_attach_thread() . . . 309
12.51.2.4 l4irq_attach_thread_ft() . . . 310
12.51.2.5 l4irq_detach() . . . 310
12.51.2.6 l4irq_unmask() . . . 311
12.51.2.7 l4irq_unmask_and_wait_any() . . . 311
12.51.2.8 l4irq_wait() . . . 311
12.51.2.9 l4irq_wait_any() . . . 313
12.52 Internal Helpers . . . 314
12.52.1 Detailed Description . . . 314
12.53 Internal constants . . . 315
12.53.1 Detailed Description . . . 316
12.54 Internal functions . . . 317
12.54.1 Detailed Description . . . 317
12.55 Interrupt controller . . . 318

12.55.1 Detailed Description . . . 319
12.55.2 Typedef Documentation . . . 319
12.55.2.1 l4_icu_info_t . . . 319
12.55.3 Enumeration Type Documentation . . . 320
12.55.3.1 L4_icu_flags . . . 320
12.55.4 Function Documentation . . . 320
12.55.4.1 l4_icu_bind() . . . 320
12.55.4.2 l4_icu_bind_u() . . . 321
12.55.4.3 l4_icu_info() . . . 322
12.55.4.4 l4_icu_info_u() . . . 323
12.55.4.5 l4_icu_mask() . . . 323
12.55.4.6 l4_icu_mask_u() . . . 324
12.55.4.7 l4_icu_msi_info() . . . 325
12.55.4.8 l4_icu_msi_info_u() . . . 326
12.55.4.9 l4_icu_set_mode() . . . 326
12.55.4.10 l4_icu_set_mode_u() . . . 327
12.55.4.11 l4_icu_unbind() . . . 328
12.55.4.12 l4_icu_unbind_u() . . . 329
12.55.4.13 l4_icu_unmask() . . . 330
12.55.4.14 l4_icu_unmask_u() . . . 330
12.56 Kernel Debugger . . . 332
12.56.1 Detailed Description . . . 332
12.56.2 Function Documentation . . . 332
12.56.2.1 l4_debugger_global_id() . . . 332
12.56.2.2 l4_debugger_kobj_to_id() . . . 333
12.56.2.3 l4_debugger_set_object_name() . . . 333
12.57 Kernel Interface Page . . . 335
12.57.1 Detailed Description . . . 336
12.57.2 Function Documentation . . . 336
12.57.2.1 l4_kernel_info_version_offset() . . . 336

12.57.2.2 l4_kip_clock() . . . 337
12.57.2.3 l4_kip_clock_lw() . . . 338
12.57.2.4 l4_kip_version() . . . 339
12.57.2.5 l4_kip_version_string() . . . 339
12.58 Kernel Interface Page API . . . 340
12.58.1 Detailed Description . . . 340
12.58.2 Macro Definition Documentation . . . 340
12.58.2.1 l4util_kip_for_each_feature . . . 340
12.58.3 Function Documentation . . . 341
12.58.3.1 l4util_kip_kernel_abi_version() . . . 341
12.58.3.2 l4util_kip_kernel_has_feature() . . . 341
12.58.3.3 l4util_kip_kernel_is_ux() . . . 341
12.58.3.4 l4util_memdesc_vm_high() . . . 342
12.59 Kernel Objects . . . 343
12.59.1 Detailed Description . . . 344
12.60 Kumem allocator utility . . . 345
12.61 Kumem utiltities . . . 346
12.61.1 Detailed Description . . . 346
12.61.2 Function Documentation . . . 346
12.61.2.1 kumem_alloc() . . . 346
12.62 L4 IPC Opcodes . . . 348
12.62.1 Detailed Description . . . 348
12.62.2 Enumeration Type Documentation . . . 348
12.62.2.1 L4_icu_opcode . . . 348
12.62.2.2 L4_ipc_gate_ops . . . 349
12.62.2.3 L4_platform_ctl_ops . . . 350
12.62.2.4 L4_task_ops . . . 350
12.62.2.5 L4_thread_ops . . . 350
12.62.2.6 L4_vcon_ops . . . 351
12.63 L4 VIRTIO Block Device . . . 352

12.63.1 Detailed Description . . . 352
12.63.2 Enumeration Type Documentation . . . 352
12.63.2.1 L4virtio_block_operations . . . 352
12.63.2.2 L4virtio_block_status . . . 353
12.64 L4 VIRTIO Interface . . . 354
12.64.1 Detailed Description . . . 354
12.65 L4 VIRTIO Transport Layer . . . 355
12.65.1 Detailed Description . . . 356
12.65.2 Typedef Documentation . . . 356
12.65.2.1 l4virtio_config_queue_t . . . 357
12.65.3 Enumeration Type Documentation . . . 357
12.65.3.1 L4_virtio_cmd . . . 357
12.65.3.2 L4_virtio_irq_status . . . 357
12.65.3.3 L4_virtio_opcodes . . . 358
12.65.3.4 L4virtio_device_ids . . . 358
12.65.3.5 L4virtio_device_status . . . 359
12.65.3.6 L4virtio_feature_bits . . . 359
12.65.4 Function Documentation . . . 359
12.65.4.1 l4virtio_config_queue() . . . 359
12.65.4.2 l4virtio_config_queues() . . . 360
12.65.4.3 l4virtio_device_config() . . . 361
12.65.4.4 l4virtio_register_ds() . . . 361
12.65.4.5 l4virtio_register_iface() . . . 362
12.65.4.6 l4virtio_set_status() . . . 363
12.66 L4 Vbus functions . . . 364
12.66.1 Detailed Description . . . 365
12.66.2 Enumeration Type Documentation . . . 365
12.66.2.1 L4vbus_dma_domain_assign_flags . . . 365
12.66.3 Function Documentation . . . 365
12.66.3.1 l4vbus_assign_dma_domain() . . . 365

12.66.3.2 l4vbus_get_device() . . . 366
12.66.3.3 l4vbus_get_device_by_hid() . . . 367
12.66.3.4 l4vbus_get_hid() . . . 368
12.66.3.5 l4vbus_get_next_device() . . . 368
12.66.3.6 l4vbus_get_resource() . . . 369
12.66.3.7 l4vbus_is_compatible() . . . 370
12.66.3.8 l4vbus_release_resource() . . . 371
12.66.3.9 l4vbus_request_resource() . . . 371
12.66.3.10 l4vbus_vicu_get_cap() . . . 372
12.67 L4 kernel object type information . . . 374
12.67.1 Detailed Description . . . 374
12.67.2 Function Documentation . . . 374
12.67.2.1 kobject_typeid() . . . 374
12.68 L4Re C Interface . . . 376
12.68.1 Detailed Description . . . 377
12.68.2 Function Documentation . . . 377
12.68.2.1 l4re_inhibitor_acquire() . . . 377
12.69 L4Re C++ Interface . . . 379
12.69.1 Detailed Description . . . 380
12.70 L4Re Capability API . . . 381
12.70.1 Detailed Description . . . 382
12.70.2 Function Documentation . . . 382
12.70.2.1 make_auto_cap() . . . 382
12.70.2.2 make_auto_del_cap() . . . 382
12.70.2.3 make_ref_cap() . . . 383
12.70.2.4 make_ref_del_cap() . . . 383
12.70.3 Variable Documentation . . . 383
12.70.3.1 cap_alloc . . . 383
12.71 L4Re ELF Auxiliary Information . . . 384
12.71.1 Detailed Description . . . 385

12.71.2 Macro Definition Documentation . . . 385
12.71.2.1 L4RE_ELF_AUX_ELEM . . . 385
12.71.2.2 L4RE_ELF_AUX_ELEM_T . . . 385
12.71.3 Enumeration Type Documentation . . . 386
12.71.3.1 anonymous enum . . . 386
12.72 L4Re Protocol identifiers . . . 387
12.72.1 Detailed Description . . . 387
12.73 L4Re Util C Interface . . . 388
12.74 L4Re Util C++ Interface . . . 389
12.74.1 Detailed Description . . . 389
12.75 L4vbus GPIO functions . . . 390
12.75.1 Detailed Description . . . 391
12.75.2 Enumeration Type Documentation . . . 391
12.75.2.1 L4vbus_gpio_generic_func . . . 391
12.75.2.2 L4vbus_gpio_pull_modes . . . 391
12.75.3 Function Documentation . . . 392
12.75.3.1 l4vbus_gpio_config_get() . . . 392
12.75.3.2 l4vbus_gpio_config_pad() . . . 393
12.75.3.3 l4vbus_gpio_config_pull() . . . 393
12.75.3.4 l4vbus_gpio_get() . . . 394
12.75.3.5 l4vbus_gpio_multi_config_pad() . . . 395
12.75.3.6 l4vbus_gpio_multi_get() . . . 395
12.75.3.7 l4vbus_gpio_multi_set() . . . 396
12.75.3.8 l4vbus_gpio_multi_setup() . . . 397
12.75.3.9 l4vbus_gpio_set() . . . 398
12.75.3.10 l4vbus_gpio_setup() . . . 399
12.75.3.11 l4vbus_gpio_to_irq() . . . 399
12.76 L4vbus PCI functions . . . 401
12.76.1 Detailed Description . . . 401
12.76.2 Function Documentation . . . 401

12.76.2.1 l4vbus_pci_cfg_read() . . . 401
12.76.2.2 l4vbus_pci_cfg_write() . . . 402
12.76.2.3 l4vbus_pci_irq_enable() . . . 403
12.76.2.4 l4vbus_pcidev_cfg_read() . . . 404
12.76.2.5 l4vbus_pcidev_cfg_write() . . . 405
12.76.2.6 l4vbus_pcidev_irq_enable() . . . 405
12.77 Log interface . . . 407
12.77.1 Detailed Description . . . 407
12.77.2 Function Documentation . . . 407
12.77.2.1 l4re_log_print() . . . 407
12.77.2.2 l4re_log_print_srv() . . . 408
12.77.2.3 l4re_log_printn() . . . 409
12.77.2.4 l4re_log_printn_srv() . . . 410
12.78 Logging interface . . . 411
12.78.1 Detailed Description . . . 411
12.79 Low-Level Thread Functions . . . 412
12.80 Machine Restarting Function . . . 413
12.80.1 Detailed Description . . . 413
12.81 Memory allocator . . . 414
12.81.1 Detailed Description . . . 414
12.81.2 Enumeration Type Documentation . . . 414
12.81.2.1 l4re_ma_flags . . . 415
12.81.3 Function Documentation . . . 415
12.81.3.1 l4re_ma_alloc() . . . 415
12.81.3.2 l4re_ma_alloc_align() . . . 416
12.81.3.3 l4re_ma_alloc_align_srv() . . . 417
12.81.3.4 l4re_ma_free() . . . 418
12.81.3.5 l4re_ma_free_srv() . . . 419
12.82 Memory descriptors (C version) . . . 420
12.82.1 Detailed Description . . . 421

12.82.2 Typedef Documentation . . . 421
12.82.2.1 l4_kernel_info_mem_desc_t . . . 421
12.82.3 Enumeration Type Documentation . . . 421
12.82.3.1 l4_mem_info_sub_type_t . . . 421
12.82.3.2 l4_mem_type_t . . . 422
12.82.4 Function Documentation . . . 422
12.82.4.1 l4_kernel_info_get_mem_desc_end() . . . 422
12.82.4.2 l4_kernel_info_get_mem_desc_is_virtual() . . . 422
12.82.4.3 l4_kernel_info_get_mem_desc_start() . . . 423
12.82.4.4 l4_kernel_info_get_mem_desc_subtype() . . . 423
12.82.4.5 l4_kernel_info_get_mem_desc_type() . . . 423
12.82.4.6 l4_kernel_info_get_num_mem_descs() . . . 424
12.82.4.7 l4_kernel_info_set_mem_desc() . . . 424
12.83 Memory operations. . . . 425
12.83.1 Detailed Description . . . 425
12.83.2 Enumeration Type Documentation . . . 425
12.83.2.1 L4_mem_op_widths . . . 425
12.83.3 Function Documentation . . . 426
12.83.3.1 l4_mem_read() . . . 426
12.83.3.2 l4_mem_write() . . . 427
12.84 Memory related . . . 428
12.84.1 Detailed Description . . . 429
12.84.2 Macro Definition Documentation . . . 429
12.84.2.1 L4_LOG2_PAGESIZE . . . 429
12.84.2.2 L4_LOG2_SUPERPAGESIZE . . . 429
12.84.2.3 L4_PAGEMASK . . . 430
12.84.2.4 L4_SUPERPAGEMASK . . . 430
12.84.2.5 L4_SUPERPAGESIZE . . . 430
12.84.3 Enumeration Type Documentation . . . 430
12.84.3.1 l4_addr_consts_t . . . 430

12.84.4 Function Documentation . . . 431
12.84.4.1 l4_round_page() . . . 431
12.84.4.2 l4_round_size() . . . 432
12.84.4.3 l4_trunc_page() . . . 433
12.84.4.4 l4_trunc_size() . . . 434
12.85 Message Items . . . 435
12.85.1 Detailed Description . . . 435
12.85.2 Enumeration Type Documentation . . . 435
12.85.2.1 l4_msg_item_consts_t . . . 435
12.85.3 Function Documentation . . . 436
12.85.3.1 l4_map_control() . . . 436
12.85.3.2 l4_map_obj_control() . . . 437
12.86 Message Registers (MRs) . . . 438
12.86.1 Detailed Description . . . 438
12.87 Message Tag . . . 439
12.87.1 Detailed Description . . . 440
12.87.2 Typedef Documentation . . . 440
12.87.2.1 l4_msgtag_t . . . 440
12.87.3 Enumeration Type Documentation . . . 441
12.87.3.1 l4_msgtag_flags . . . 441
12.87.3.2 l4_msgtag_protocol . . . 441
12.87.3.3 L4_platform_ctl_proto . . . 442
12.87.4 Function Documentation . . . 442
12.87.4.1 l4_msgtag() . . . 442
12.87.4.2 l4_msgtag_flags() . . . 444
12.87.4.3 l4_msgtag_has_error() . . . 445
12.87.4.4 l4_msgtag_is_exception() . . . 446
12.87.4.5 l4_msgtag_is_io_page_fault() . . . 447
12.87.4.6 l4_msgtag_is_page_fault() . . . 448
12.87.4.7 l4_msgtag_is_preemption() . . . 449

12.87.4.8 l4_msgtag_is_sigma0() . . . 450
12.87.4.9 l4_msgtag_is_sys_exception() . . . 451
12.87.4.10 l4_msgtag_items() . . . 452
12.87.4.11 l4_msgtag_label() . . . 453
12.87.4.12 l4_msgtag_words() . . . 454
12.88 Name-space API . . . 455
12.88.1 Detailed Description . . . 455
12.89 Namespace interface . . . 456
12.89.1 Detailed Description . . . 456
12.89.2 Enumeration Type Documentation . . . 456
12.89.2.1 l4re_ns_register_flags . . . 456
12.89.3 Function Documentation . . . 457
12.89.3.1 l4re_ns_query_srv() . . . 457
12.89.3.2 l4re_ns_query_to_srv() . . . 457
12.89.3.3 l4re_ns_register_obj_srv() . . . 459
12.90 Object Invocation . . . 461
12.90.1 Detailed Description . . . 462
12.90.2 Enumeration Type Documentation . . . 463
12.90.2.1 l4_syscall_flags_t . . . 463
12.90.3 Function Documentation . . . 463
12.90.3.1 l4_ipc() . . . 463
12.90.3.2 l4_ipc_call() . . . 465
12.90.3.3 l4_ipc_receive() . . . 466
12.90.3.4 l4_ipc_reply_and_wait() . . . 467
12.90.3.5 l4_ipc_send() . . . 468
12.90.3.6 l4_ipc_send_and_wait() . . . 470
12.90.3.7 l4_ipc_sleep() . . . 471
12.90.3.8 l4_ipc_wait() . . . 472
12.90.3.9 l4_sndfpage_add() . . . 473
12.91 Parent API . . . 474

12.91.1 Detailed Description . . . 474
12.92 Platform Control C API . . . 475
12.92.1 Detailed Description . . . 475
12.92.2 Function Documentation . . . 475
12.92.2.1 l4_platform_ctl_cpu_disable() . . . 475
12.92.2.2 l4_platform_ctl_cpu_enable() . . . 476
12.92.2.3 l4_platform_ctl_system_shutdown() . . . 476
12.92.2.4 l4_platform_ctl_system_suspend() . . . 477
12.93 Producer . . . 478
12.93.1 Detailed Description . . . 478
12.93.2 Function Documentation . . . 478
12.93.2.1 l4shmc_chunk_ready() . . . 478
12.93.2.2 l4shmc_chunk_ready_sig() . . . 479
12.93.2.3 l4shmc_chunk_try_to_take() . . . 479
12.93.2.4 l4shmc_is_chunk_clear() . . . 480
12.94 Producer . . . 481
12.94.1 Detailed Description . . . 481
12.94.2 Function Documentation . . . 481
12.94.2.1 l4shmc_trigger() . . . 481
12.95 Random number support . . . 482
12.95.1 Detailed Description . . . 482
12.95.2 Function Documentation . . . 482
12.95.2.1 l4util_rand() . . . 482
12.95.2.2 l4util_srand() . . . 482
12.96 Realtime API . . . 484
12.97 Region map API . . . 485
12.97.1 Detailed Description . . . 485
12.98 Region map interface . . . 486
12.98.1 Detailed Description . . . 487
12.98.2 Enumeration Type Documentation . . . 487

12.98.2.1 l4re_rm_flags_t . . . 487
12.98.3 Function Documentation . . . 487
12.98.3.1 l4re_rm_attach() . . . 487
12.98.3.2 l4re_rm_attach_srv() . . . 489
12.98.3.3 l4re_rm_detach() . . . 489
12.98.3.4 l4re_rm_detach_ds() . . . 490
12.98.3.5 l4re_rm_detach_ds_unmap() . . . 491
12.98.3.6 l4re_rm_detach_srv() . . . 492
12.98.3.7 l4re_rm_detach_unmap() . . . 493
12.98.3.8 l4re_rm_find() . . . 494
12.98.3.9 l4re_rm_find_srv() . . . 495
12.98.3.10 l4re_rm_free_area() . . . 495
12.98.3.11 l4re_rm_free_area_srv() . . . 496
12.98.3.12 l4re_rm_reserve_area() . . . 496
12.98.3.13 l4re_rm_reserve_area_srv() . . . 497
12.98.3.14 l4re_rm_show_lists() . . . 497
12.99 Scheduler . . . 498
12.99.1 Detailed Description . . . 499
12.99.2 Enumeration Type Documentation . . . 499
12.99.2.1 L4_scheduler_ops . . . 499
12.99.3 Function Documentation . . . 499
12.99.3.1 l4_sched_cpu_set() . . . 499
12.99.3.2 l4_scheduler_idle_time() . . . 500
12.99.3.3 l4_scheduler_info() . . . 501
12.99.3.4 l4_scheduler_is_online() . . . 502
12.99.3.5 l4_scheduler_run_thread() . . . 502
12.100 Server-Side IPC framework . . . 504
12.100.1 Detailed Description . . . 505
12.100.2 Enumeration Type Documentation . . . 505
12.100.2.1 Reply_mode . . . 505

12.101 Shared Memory Library . . . 506
12.101.1 Detailed Description . . . 507
12.101.2 Function Documentation . . . 507
12.101.2.1 l4shmc_area_overhead() . . . 507
12.101.2.2 l4shmc_area_size() . . . 507
12.101.2.3 l4shmc_area_size_free() . . . 508
12.101.2.4 l4shmc_attach() . . . 508
12.101.2.5 l4shmc_attach_to() . . . 509
12.101.2.6 l4shmc_chunk_overhead() . . . 509
12.101.2.7 l4shmc_connect_chunk_signal() . . . 510
12.101.2.8 l4shmc_create() . . . 511
12.102 Sigma0 API . . . 512
12.102.1 Detailed Description . . . 513
12.102.2 Enumeration Type Documentation . . . 513
12.102.2.1 l4sigma0_return_flags_t . . . 513
12.102.3 Function Documentation . . . 513
12.102.3.1 l4sigma0_debug_dump() . . . 513
12.102.3.2 l4sigma0_map_anypage() . . . 514
12.102.3.3 l4sigma0_map_errstr() . . . 514
12.102.3.4 l4sigma0_map_iomem() . . . 515
12.102.3.5 l4sigma0_map_kip() . . . 515
12.102.3.6 l4sigma0_map_mem() . . . 516
12.102.3.7 l4sigma0_map_tbuf() . . . 516
12.102.3.8 l4sigma0_new_client() . . . 517
12.103 Signals . . . 518
12.103.1 Detailed Description . . . 518
12.103.2 Function Documentation . . . 518
12.103.2.1 l4shmc_add_signal() . . . 518
12.103.2.2 l4shmc_attach_signal() . . . 519
12.103.2.3 l4shmc_attach_signal_to() . . . 519

12.103.2.4 l4shmc_check_magic() . . . 520
12.103.2.5 l4shmc_get_signal_to() . . . 520
12.103.2.6 l4shmc_signal_cap() . . . 521
12.104 Small C++ Template Library . . . 522
12.104.1 Detailed Description . . . 523
12.104.2 Function Documentation . . . 523
12.104.2.1 max() . . . 523
12.104.2.2 min() . . . 524
12.104.2.3 operator new() . . . 525
12.105 Task . . . 526
12.105.1 Detailed Description . . . 527
12.105.2 Enumeration Type Documentation . . . 527
12.105.2.1 l4_unmap_flags_t . . . 527
12.105.3 Function Documentation . . . 528
12.105.3.1 l4_task_add_ku_mem() . . . 528
12.105.3.2 l4_task_cap_equal() . . . 528
12.105.3.3 l4_task_cap_has_child() . . . 529
12.105.3.4 l4_task_cap_valid() . . . 529
12.105.3.5 l4_task_delete_obj() . . . 530
12.105.3.6 l4_task_map() . . . 530
12.105.3.7 l4_task_release_cap() . . . 531
12.105.3.8 l4_task_unmap() . . . 531
12.105.3.9 l4_task_unmap_batch() . . . 532
12.106 Thread . . . 534
12.106.1 Detailed Description . . . 535
12.106.2 Enumeration Type Documentation . . . 536
12.106.2.1 L4_thread_control_flags . . . 536
12.106.2.2 L4_thread_control_mr_indices . . . 536
12.106.2.3 L4_thread_ex_regs_flags . . . 538
12.106.3 Function Documentation . . . 538

12.106.3.14 l4_thread_arm_set_tpidruro() 538
12.106.3.24 l4_thread_ex_regs() 539
12.106.3.34 l4_thread_ex_regs_ret() 540
12.106.3.44 l4_thread_ex_regs_ret_u() 541
12.106.3.54 l4_thread_ex_regs_u() 542
12.106.3.64 l4_thread_modify_sender_add() 543
12.106.3.74 l4_thread_modify_sender_commit() 544
12.106.3.84 l4_thread_modify_sender_start() 544
12.106.3.94 l4_thread_register_del_irq() 544
12.106.3.10 l4_thread_stats_time() 545
12.106.3.11 l4_thread_switch() 545
12.106.3.12 l4_thread_vcpu_control() 546
12.106.3.13 l4_thread_vcpu_control_ext() 546
12.106.3.14 l4_thread_vcpu_control_ext_u() 547
12.106.3.15 l4_thread_vcpu_control_u() 548
12.106.3.16 l4_thread_vcpu_resume_commit() 549
12.106.3.17 l4_thread_vcpu_resume_start() 550
12.106.3.18 l4_thread_yield() 550
12.107 Thread Control Registers (TCRs) 551
12.107.1 Detailed Description 551
12.108 Thread control 552
12.108.1 Detailed Description 552
12.108.2 Function Documentation 553
12.108.2.1 l4_thread_control_alien() 553
12.108.2.2 l4_thread_control_bind() 553
12.108.2.3 l4_thread_control_commit() 554
12.108.2.4 l4_thread_control_exc_handler() 554
12.108.2.5 l4_thread_control_pager() 555
12.108.2.6 l4_thread_control_start() 555
12.108.2.7 l4_thread_control_ux_host_syscall() 556

12.109 Timeouts . . . 557
12.109.1 Detailed Description . . . 558
12.109.2 Macro Definition Documentation . . . 558
12.109.2.1 L4_IPC_TIMEOUT_0 . . . 558
12.109.3 Typedef Documentation . . . 559
12.109.3.1 l4_timeout_s . . . 559
12.109.3.2 l4_timeout_t . . . 559
12.109.4 Enumeration Type Documentation . . . 559
12.109.4.1 l4_timeout_abs_validity . . . 559
12.109.5 Function Documentation . . . 559
12.109.5.1 l4_ipc_timeout() . . . 559
12.109.5.2 l4_rcv_timeout() . . . 560
12.109.5.3 l4_snd_timeout() . . . 560
12.109.5.4 l4_timeout() . . . 561
12.109.5.5 l4_timeout_abs() . . . 561
12.109.5.6 l4_timeout_get() . . . 562
12.109.5.7 l4_timeout_is_absolute() . . . 563
12.109.5.8 l4_timeout_rel() . . . 564
12.109.5.9 l4_timeout_rel_get() . . . 564
12.109.5.10 l4_utcb_mr64_idx() . . . 565
12.110 Timestamp Counter . . . 566
12.110.1 Detailed Description . . . 567
12.110.2 Function Documentation . . . 567
12.110.2.1 l4_busy_wait_ns() . . . 567
12.110.2.2 l4_busy_wait_us() . . . 568
12.110.2.3 l4_calibrate_tsc() . . . 569
12.110.2.4 l4_get_hz() . . . 569
12.110.2.5 l4_ns_to_tsc() . . . 569
12.110.2.6 l4_rdpmc() . . . 570
12.110.2.7 l4_rdpmc_32() . . . 571

12.110.2.8 l4_rdtsc() . . . 571
12.110.2.9 l4_rdtsc_32() . . . 572
12.110.2.10 l4_tsc_init() . . . 572
12.110.2.11 l4_tsc_to_ns() . . . 573
12.110.2.12 l4_tsc_to_s_and_ns() . . . 573
12.110.2.13 l4_tsc_to_us() . . . 574
12.111 Utility Functions . . . 575
12.111.1 Detailed Description . . . 576
12.111.2 Function Documentation . . . 576
12.111.2.1 l4_sleep() . . . 576
12.111.2.2 l4_touch_ro() . . . 577
12.111.2.3 l4_touch_rw() . . . 578
12.111.2.4 l4_usleep() . . . 578
12.111.2.5 l4util_micros2l4to() . . . 579
12.111.2.6 l4util_splitlog2_hdl() . . . 579
12.111.2.7 l4util_splitlog2_size() . . . 580
12.112 VM API for SVM . . . 582
12.112.1 Detailed Description . . . 582
12.113 VM API for TZ . . . 583
12.113.1 Detailed Description . . . 583
12.114 VM API for VMX . . . 584
12.114.1 Detailed Description . . . 585
12.114.2 Enumeration Type Documentation . . . 585
12.114.2.1 anonymous enum . . . 585
12.114.2.2 L4_vm_vmx_caps_regs . . . 586
12.114.2.3 L4_vm_vmx_dfl1_regs . . . 587
12.114.3 Function Documentation . . . 587
12.114.3.1 l4_vm_vmx_clear() . . . 587
12.114.3.2 l4_vm_vmx_field_len() . . . 588
12.114.3.3 l4_vm_vmx_field_order() . . . 588

12.114.3.4 l4_vm_vmx_get_caps() . . . 589
12.114.3.5 l4_vm_vmx_get_caps_default1() . . . 589
12.114.3.6 l4_vm_vmx_get_cr2_index() . . . 590
12.114.3.7 l4_vm_vmx_ptr_load() . . . 590
12.114.3.8 l4_vm_vmx_read() . . . 591
12.114.3.9 l4_vm_vmx_read_16() . . . 592
12.114.3.10 l4_vm_vmx_read_32() . . . 593
12.114.3.11 l4_vm_vmx_read_64() . . . 593
12.114.3.12 l4_vm_vmx_read_nat() . . . 594
12.114.3.13 l4_vm_vmx_write() . . . 595
12.114.3.14 l4_vm_vmx_write_16() . . . 595
12.114.3.15 l4_vm_vmx_write_32() . . . 596
12.114.3.16 l4_vm_vmx_write_64() . . . 597
12.114.3.17 l4_vm_vmx_write_nat() . . . 597
12.115 Vbus API . . . 600
12.115.1 Detailed Description . . . 600
12.116 Video API . . . 601
12.116.1 Detailed Description . . . 602
12.116.2 Typedef Documentation . . . 602
12.116.2.1 l4re_video_view_t . . . 602
12.116.3 Enumeration Type Documentation . . . 603
12.116.3.1 l4re_video_goos_info_flags_t . . . 603
12.116.3.2 l4re_video_view_info_flags_t . . . 603
12.116.4 Function Documentation . . . 603
12.116.4.1 l4re_video_goos_create_buffer() . . . 604
12.116.4.2 l4re_video_goos_create_view() . . . 604
12.116.4.3 l4re_video_goos_delete_buffer() . . . 604
12.116.4.4 l4re_video_goos_delete_view() . . . 606
12.116.4.5 l4re_video_goos_get_static_buffer() . . . 606
12.116.4.6 l4re_video_goos_get_view() . . . 606

12.116.4.7 l4re_video_goos_info() . . . 607
12.116.4.8 l4re_video_goos_refresh() . . . 607
12.116.4.9 l4re_video_view_get_info() . . . 608
12.116.4.10 l4re_video_view_refresh() . . . 608
12.116.4.11 l4re_video_view_set_info() . . . 609
12.116.4.12 l4re_video_view_set_viewport() . . . 609
12.116.4.13 l4re_video_view_stack() . . . 609
12.117 Virtual Console . . . 611
12.117.1 Detailed Description . . . 612
12.117.2 Enumeration Type Documentation . . . 612
12.117.2.1 L4_vcon_i_flags . . . 612
12.117.2.2 L4_vcon_l_flags . . . 613
12.117.2.3 L4_vcon_o_flags . . . 613
12.117.2.4 L4_vcon_size_consts . . . 613
12.117.3 Function Documentation . . . 614
12.117.3.1 l4_vcon_get_attr() . . . 614
12.117.3.2 l4_vcon_get_attr_u() . . . 615
12.117.3.3 l4_vcon_read() . . . 615
12.117.3.4 l4_vcon_read_u() . . . 616
12.117.3.5 l4_vcon_read_with_flags() . . . 617
12.117.3.6 l4_vcon_send() . . . 618
12.117.3.7 l4_vcon_send_u() . . . 619
12.117.3.8 l4_vcon_set_attr() . . . 620
12.117.3.9 l4_vcon_set_attr_u() . . . 621
12.117.3.10 l4_vcon_write() . . . 621
12.117.3.11 l4_vcon_write_u() . . . 622
12.118 Virtual Machines . . . 624
12.118.1 Detailed Description . . . 624
12.119 Virtual Registers (UTCBs) . . . 625
12.119.1 Detailed Description . . . 626

12.119.2 Typedef Documentation . . . 626
12.119.2.1 l4_utcb_t . . . 626
12.119.3 Function Documentation . . . 627
12.119.3.1 l4_utcb_br() . . . 627
12.119.3.2 l4_utcb_mr() . . . 627
12.119.3.3 l4_utcb_tcr() . . . 628
12.120 amd64 Virtual Registers (UTCB) . . . 629
12.120.1 Detailed Description . . . 629
12.121 vCPU API . . . 630
12.121.1 Detailed Description . . . 631
12.121.2 Enumeration Type Documentation . . . 631
12.121.2.1 L4_vcpu_state_flags . . . 631
12.121.2.2 L4_vcpu_state_offset . . . 631
12.121.2.3 L4_vcpu_sticky_flags . . . 631
12.122 vCPU Support Library . . . 633
12.122.1 Detailed Description . . . 634
12.122.2 Function Documentation . . . 634
12.122.2.1 l4vcpu_irq_disable() . . . 634
12.122.2.2 l4vcpu_irq_disable_save() . . . 635
12.122.2.3 l4vcpu_irq_enable() . . . 635
12.122.2.4 l4vcpu_irq_restore() . . . 636
12.122.2.5 l4vcpu_is_irq_entry() . . . 637
12.122.2.6 l4vcpu_is_page_fault_entry() . . . 638
12.122.2.7 l4vcpu_print_state() . . . 639
12.122.2.8 l4vcpu_wait_for_event() . . . 639
12.123 x86 Virtual Registers (UTCB) . . . 641
12.123.1 Detailed Description . . . 641
12.123.2 Enumeration Type Documentation . . . 641
12.123.2.1 L4_utcb_consts_x86 . . . 641

13 Namespace Documentation 643

13.1 cxx Namespace Reference . . . 643

13.1.1 Detailed Description . . . 645

13.1.2 Typedef Documentation . . . 645

13.1.2.1 H_list_item . . . 645

13.2 cxx::Bits Namespace Reference . . . 645

13.2.1 Detailed Description . . . 646

13.3 L4 Namespace Reference . . . 646

13.3.1 Detailed Description . . . 650

13.3.2 Enumeration Type Documentation . . . 650

13.3.2.1 anonymous enum . . . 650

13.3.3 Function Documentation . . . 650

13.3.3.1 cap_cast() [1/2] . . . 650

13.3.3.2 cap_cast() [2/2] . . . 651

13.3.3.3 cap_dynamic_cast() . . . 652

13.3.3.4 cap_reinterpret_cast() [1/2] . . . 653

13.3.3.5 cap_reinterpret_cast() [2/2] . . . 654

13.3.3.6 throw_ipc_exception() [1/2] . . . 655

13.3.3.7 throw_ipc_exception() [2/2] . . . 656

13.4 L4::Ipc Namespace Reference . . . 657

13.4.1 Detailed Description . . . 659

13.4.2 Function Documentation . . . 659

13.4.2.1 buf_cp_in() . . . 660

13.4.2.2 buf_cp_out() . . . 660

13.4.2.3 buf_in() . . . 661

13.4.2.4 make_cap() . . . 661

13.4.2.5 make_cap_full() . . . 662

13.4.2.6 make_cap_rw() . . . 663

13.4.2.7 make_cap_rws() . . . 664

13.4.2.8 msg_ptr() . . . 664

13.4.2.9 read() . . . 665
13.4.2.10 str_cp_in() . . . 665
13.5 L4::Ipc::Msg Namespace Reference . . . 666
13.5.1 Detailed Description . . . 667
13.5.2 Enumeration Type Documentation . . . 667
13.5.2.1 anonymous enum . . . 667
13.5.3 Function Documentation . . . 668
13.5.3.1 align_to() [1/2] . . . 668
13.5.3.2 align_to() [2/2] . . . 669
13.5.3.3 check_size() [1/2] . . . 669
13.5.3.4 check_size() [2/2] . . . 670
13.5.3.5 msg_add() . . . 670
13.5.3.6 msg_get() . . . 671
13.6 L4::Ipc_svr Namespace Reference . . . 672
13.6.1 Detailed Description . . . 673
13.7 L4::Typeid Namespace Reference . . . 673
13.7.1 Detailed Description . . . 673
13.8 L4::Types Namespace Reference . . . 673
13.8.1 Detailed Description . . . 674
13.9 L4Re Namespace Reference . . . 674
13.9.1 Detailed Description . . . 676
13.9.2 Typedef Documentation . . . 676
13.9.2.1 Shared_cap . . . 676
13.9.2.2 shared_cap . . . 676
13.9.2.3 Shared_del_cap . . . 677
13.9.2.4 shared_del_cap . . . 677
13.9.2.5 Unique_cap . . . 678
13.9.2.6 unique_cap . . . 678
13.9.2.7 Unique_del_cap . . . 679
13.9.2.8 unique_del_cap . . . 679

13.9.3 Function Documentation . . . 680
13.9.3.1 chkcap() . . . 680
13.9.3.2 chkipc() . . . 681
13.9.3.3 chksys() [1/3] . . . 682
13.9.3.4 chksys() [2/3] . . . 683
13.9.3.5 chksys() [3/3] . . . 684
13.9.3.6 make_shared_cap() . . . 685
13.9.3.7 make_shared_del_cap() . . . 686
13.9.3.8 make_unique_cap() . . . 687
13.9.3.9 make_unique_del_cap() . . . 687
13.10 L4Re::Util Namespace Reference . . . 688
13.10.1 Detailed Description . . . 690
13.10.2 Typedef Documentation . . . 690
13.10.2.1 Shared_cap . . . 690
13.10.2.2 shared_cap . . . 691
13.10.2.3 Shared_del_cap . . . 692
13.10.2.4 shared_del_cap . . . 692
13.10.2.5 Unique_cap . . . 693
13.10.2.6 unique_cap . . . 694
13.10.2.7 Unique_del_cap . . . 694
13.10.2.8 unique_del_cap . . . 695
13.10.3 Function Documentation . . . 696
13.10.3.1 make_shared_cap() . . . 696
13.10.3.2 make_shared_del_cap() . . . 696
13.10.3.3 make_unique_cap() . . . 697
13.10.3.4 make_unique_del_cap() . . . 697
13.11 L4Re::Vfs Namespace Reference . . . 698
13.11.1 Detailed Description . . . 699
13.12 L4virtio Namespace Reference . . . 699
13.12.1 Detailed Description . . . 699

14 Data Structure Documentation 701

14.1 cxx::Auto_ptr< T > Class Template Reference . . . 701

14.1.1 Detailed Description . . . 702

14.1.2 Member Typedef Documentation . . . 702

14.1.2.1 Ref_type . . . 702

14.1.3 Constructor & Destructor Documentation . . . 703

14.1.3.1 Auto_ptr() [1/2] . . . 703

14.1.3.2 Auto_ptr() [2/2] . . . 703

14.1.3.3 ~Auto_ptr() . . . 703

14.1.4 Member Function Documentation . . . 703

14.1.4.1 get() . . . 704

14.1.4.2 operator Priv_type *() . . . 704

14.1.4.3 operator*() . . . 704

14.1.4.4 operator->() . . . 704

14.1.4.5 operator=() . . . 704

14.1.4.6 release() . . . 705

14.2 cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > Class Template Reference . . . 706

14.2.1 Detailed Description . . . 708

14.2.2 Constructor & Destructor Documentation . . . 708

14.2.2.1 Avl_map() . . . 708

14.2.3 Member Function Documentation . . . 709

14.2.3.1 insert() . . . 709

14.2.3.2 operator[]() [1/2] . . . 709

14.2.3.3 operator[]() [2/2] . . . 710

14.3 cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > Class Template Reference . . . 710

14.3.1 Detailed Description . . . 712

14.4 cxx::Avl_tree< Node, Get_key, Compare > Class Template Reference . . . 713

14.4.1 Detailed Description . . . 716

14.4.2 Member Function Documentation . . . 717

14.4.2.1 insert() . . . 717

14.4.2.2 remove() . . . 717
14.5 cxx::Avl_tree_node Class Reference . . . 718
14.5.1 Detailed Description . . . 720
14.6 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference . . . 721
14.6.1 Detailed Description . . . 722
14.6.2 Member Enumeration Documentation . . . 723
14.6.2.1 anonymous enum . . . 723
14.6.3 Member Function Documentation . . . 723
14.6.3.1 free_objects() . . . 723
14.6.3.2 total_objects() . . . 724
14.7 cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference . . . 724
14.7.1 Detailed Description . . . 726
14.7.2 Member Enumeration Documentation . . . 727
14.7.2.1 anonymous enum . . . 727
14.7.3 Member Function Documentation . . . 727
14.7.3.1 alloc() . . . 727
14.7.3.2 free() . . . 727
14.7.3.3 free_objects() . . . 728
14.7.3.4 total_objects() . . . 728
14.8 cxx::Bitfield< T, LSB, MSB > Class Template Reference . . . 729
14.8.1 Detailed Description . . . 730
14.8.2 Member Typedef Documentation . . . 730
14.8.2.1 Bits_type . . . 730
14.8.2.2 Ref . . . 731
14.8.2.3 Ref_unshifted . . . 731
14.8.2.4 Shift_type . . . 731
14.8.2.5 Val . . . 731
14.8.2.6 Val_unshifted . . . 731
14.8.3 Member Enumeration Documentation . . . 731
14.8.3.1 anonymous enum . . . 731

14.8.3.2 Masks . . . 732
14.8.4 Member Function Documentation . . . 732
14.8.4.1 get() . . . 732
14.8.4.2 get_unshifted() . . . 733
14.8.4.3 set() . . . 733
14.8.4.4 set_dirty() . . . 734
14.8.4.5 set_unshifted() . . . 735
14.8.4.6 set_unshifted_dirty() . . . 736
14.8.4.7 val() . . . 737
14.8.4.8 val_dirty() . . . 738
14.8.4.9 val_unshifted() . . . 738
14.9 cxx::Bitfield< T, LSB, MSB >::Value< TT > Class Template Reference . . . 739
14.9.1 Detailed Description . . . 740
14.10 cxx::Bitfield< T, LSB, MSB >::Value_base< TT > Class Template Reference . . . 740
14.10.1 Detailed Description . . . 741
14.11 cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Class Template Reference . . . 742
14.11.1 Detailed Description . . . 743
14.12 cxx::Bitmap< BITS > Class Template Reference . . . 743
14.12.1 Detailed Description . . . 745
14.12.2 Constructor & Destructor Documentation . . . 746
14.12.2.1 Bitmap() . . . 746
14.12.3 Member Function Documentation . . . 746
14.12.3.1 scan_zero() . . . 746
14.13 cxx::Bitmap_base Class Reference . . . 747
14.13.1 Detailed Description . . . 750
14.13.2 Member Enumeration Documentation . . . 750
14.13.2.1 anonymous enum . . . 750
14.13.3 Member Function Documentation . . . 751
14.13.3.1 bit() [1/2] . . . 751
14.13.3.2 bit() [2/2] . . . 752

14.13.3.3 bit_index() . . . 752
14.13.3.4 chars() . . . 753
14.13.3.5 clear_bit() . . . 753
14.13.3.6 operator[]() [1/2] . . . 754
14.13.3.7 operator[]() [2/2] . . . 755
14.13.3.8 scan_zero() . . . 755
14.13.3.9 set_bit() . . . 756
14.13.3.10 word_index() . . . 757
14.13.3.11 words() . . . 758
14.14 cxx::Bitmap_base::Bit Class Reference . . . 758
14.14.1 Detailed Description . . . 758
14.15 cxx::Bitmap_base::Char< BITS > Class Template Reference . . . 759
14.15.1 Detailed Description . . . 759
14.16 cxx::Bitmap_base::Word< BITS > Class Template Reference . . . 759
14.16.1 Detailed Description . . . 760
14.17 cxx::Bits::Avl_map_get_key< KEY_TYPE > Struct Template Reference . . . 761
14.17.1 Detailed Description . . . 761
14.18 cxx::Bits::Avl_set_get_key< KEY_TYPE > Struct Template Reference . . . 761
14.18.1 Detailed Description . . . 762
14.19 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > Class Template Reference 762
14.19.1 Detailed Description . . . 765
14.19.2 Member Enumeration Documentation . . . 766
14.19.2.1 anonymous enum . . . 766
14.19.3 Constructor & Destructor Documentation . . . 766
14.19.3.1 Base_avl_set() [1/2] . . . 766
14.19.3.2 Base_avl_set() [2/2] . . . 767
14.19.4 Member Function Documentation . . . 767
14.19.4.1 begin() [1/2] . . . 767
14.19.4.2 begin() [2/2] . . . 768
14.19.4.3 end() [1/2] . . . 768

14.19.4.4 end() [2/2] . . . 769
14.19.4.5 erase() . . . 769
14.19.4.6 find_node() . . . 769
14.19.4.7 insert() . . . 770
14.19.4.8 lower_bound_node() . . . 770
14.19.4.9 rbegin() [1/2] . . . 771
14.19.4.10 rbegin() [2/2] . . . 771
14.19.4.11 remove() . . . 771
14.19.4.12 rend() [1/2] . . . 772
14.19.4.13 rend() [2/2] . . . 772
14.20 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node Class Reference . 773
14.20.1 Detailed Description . . . 773
14.20.2 Member Function Documentation . . . 774
14.20.2.1 operator*() . . . 774
14.20.2.2 operator->() . . . 774
14.20.2.3 valid() . . . 774
14.21 cxx::Bits::Basic_list< POLICY > Class Template Reference . . . 775
14.21.1 Detailed Description . . . 776
14.21.2 Member Function Documentation . . . 776
14.21.2.1 clear() . . . 776
14.21.2.2 iter() . . . 777
14.22 cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference . . . 777
14.22.1 Detailed Description . . . 781
14.22.2 Member Function Documentation . . . 781
14.22.2.1 begin() [1/2] . . . 782
14.22.2.2 begin() [2/2] . . . 782
14.22.2.3 dir() [1/2] . . . 782
14.22.2.4 dir() [2/2] . . . 783
14.22.2.5 end() [1/2] . . . 784
14.22.2.6 end() [2/2] . . . 785

14.22.2.7 find() . . . 785
14.22.2.8 find_node() . . . 786
14.22.2.9 lower_bound_node() . . . 786
14.22.2.10 rbegin() [1/2] . . . 787
14.22.2.11 rbegin() [2/2] . . . 788
14.22.2.12 remove_all() . . . 788
14.22.2.13 remove_tree() . . . 789
14.22.2.14 rend() [1/2] . . . 790
14.22.2.15 rend() [2/2] . . . 790
14.23 cxx::Bits::Bst_node Class Reference . . . 791
14.23.1 Detailed Description . . . 792
14.24 cxx::Bits::Direction Struct Reference . . . 793
14.24.1 Detailed Description . . . 794
14.24.2 Member Enumeration Documentation . . . 794
14.24.2.1 Direction_e . . . 794
14.24.3 Member Function Documentation . . . 794
14.24.3.1 operator"!() . . . 794
14.25 cxx::Bits::Smart_ptr_list< ITEM > Class Template Reference . . . 795
14.25.1 Detailed Description . . . 796
14.25.2 Member Function Documentation . . . 796
14.25.2.1 pop_front() . . . 796
14.26 cxx::Bits::Smart_ptr_list_item< T, STORE_T > Class Template Reference . . . 797
14.26.1 Detailed Description . . . 797
14.27 cxx::H_list< T, POLICY > Class Template Reference . . . 798
14.27.1 Detailed Description . . . 801
14.27.2 Member Function Documentation . . . 801
14.27.2.1 erase() . . . 801
14.27.2.2 insert() . . . 802
14.27.2.3 insert_after() . . . 802
14.27.2.4 insert_before() . . . 803

14.27.2.5 iter() . . . 803
14.27.2.6 pop_front() . . . 804
14.27.2.7 remove() . . . 804
14.27.2.8 replace() . . . 804
14.28 cxx::H_list_item_t< ELEM_TYPE > Class Template Reference . . . 805
14.28.1 Detailed Description . . . 806
14.28.2 Constructor & Destructor Documentation . . . 806
14.28.2.1 H_list_item_t() . . . 806
14.28.2.2 ~H_list_item_t() . . . 807
14.29 cxx::H_list_t< T > Struct Template Reference . . . 807
14.29.1 Detailed Description . . . 809
14.30 cxx::List< D, Alloc > Class Template Reference . . . 810
14.30.1 Detailed Description . . . 811
14.30.2 Member Function Documentation . . . 811
14.30.2.1 items() . . . 811
14.30.2.2 operator[]() [1/2] . . . 811
14.30.2.3 operator[]() [2/2] . . . 811
14.30.2.4 push_back() . . . 812
14.30.2.5 push_front() . . . 812
14.30.2.6 remove() . . . 812
14.30.2.7 size() . . . 812
14.31 cxx::List< D, Alloc >::Iter Class Reference . . . 813
14.31.1 Detailed Description . . . 813
14.32 cxx::List_alloc Class Reference . . . 814
14.32.1 Detailed Description . . . 814
14.32.2 Constructor & Destructor Documentation . . . 814
14.32.2.1 List_alloc() . . . 815
14.32.3 Member Function Documentation . . . 815
14.32.3.1 alloc() . . . 815
14.32.3.2 alloc_max() . . . 816

14.32.3.3 avail() . . . 817
14.32.3.4 free() . . . 818
14.33 cxx::List_item Class Reference . . . 818
14.33.1 Detailed Description . . . 820
14.33.2 Member Function Documentation . . . 820
14.33.2.1 get_next_item() . . . 820
14.33.2.2 get_prev_item() . . . 820
14.33.2.3 insert_next_item() . . . 821
14.33.2.4 insert_prev_item() . . . 821
14.33.2.5 push_back() . . . 821
14.33.2.6 push_front() . . . 822
14.33.2.7 remove() . . . 822
14.33.2.8 remove_me() . . . 823
14.34 cxx::List_item::Iter Class Reference . . . 824
14.34.1 Detailed Description . . . 826
14.34.2 Member Function Documentation . . . 826
14.34.2.1 remove_me() . . . 826
14.35 cxx::List_item::T_iter< T, Poly > Class Template Reference . . . 827
14.35.1 Detailed Description . . . 829
14.36 cxx::Lt_functor< Obj > Struct Template Reference . . . 830
14.36.1 Detailed Description . . . 830
14.37 cxx::New_allocator< _Type > Class Template Reference . . . 830
14.37.1 Detailed Description . . . 831
14.38 cxx::Nothrow Class Reference . . . 831
14.38.1 Detailed Description . . . 831
14.39 cxx::Pair< First, Second > Struct Template Reference . . . 832
14.39.1 Detailed Description . . . 832
14.39.2 Constructor & Destructor Documentation . . . 833
14.39.2.1 Pair() . . . 833
14.40 cxx::Pair_first_compare< Cmp, Typ > Class Template Reference . . . 833

14.40.1 Detailed Description . . . 834
14.40.2 Constructor & Destructor Documentation . . . 834
14.40.2.1 Pair_first_compare() . . . 834
14.40.3 Member Function Documentation . . . 834
14.40.3.1 operator()() . . . 835
14.41 cxx::Ref_obj_list_item< T > Struct Template Reference . . . 835
14.41.1 Detailed Description . . . 836
14.42 cxx::Ref_ptr< T, CNT > Class Template Reference . . . 836
14.42.1 Detailed Description . . . 838
14.42.2 Constructor & Destructor Documentation . . . 839
14.42.2.1 Ref_ptr() [1/3] . . . 839
14.42.2.2 Ref_ptr() [2/3] . . . 840
14.42.2.3 Ref_ptr() [3/3] . . . 840
14.42.3 Member Function Documentation . . . 840
14.42.3.1 get() . . . 840
14.42.3.2 ptr() . . . 841
14.42.3.3 release() . . . 841
14.43 cxx::S_list< T, POLICY > Class Template Reference . . . 842
14.43.1 Detailed Description . . . 845
14.43.2 Member Function Documentation . . . 845
14.43.2.1 pop_front() . . . 845
14.44 cxx::Slab< Type, Slab_size, Max_free, Alloc > Class Template Reference . . . 845
14.44.1 Detailed Description . . . 848
14.44.2 Member Function Documentation . . . 848
14.44.2.1 alloc() . . . 848
14.44.2.2 free() . . . 848
14.45 cxx::Slab_static< Type, Slab_size, Max_free, Alloc > Class Template Reference . . . 849
14.45.1 Detailed Description . . . 851
14.45.2 Member Function Documentation . . . 852
14.45.2.1 alloc() . . . 852

14.46 cxx::static_vector< T, IDX > Class Template Reference . . . 852
14.46.1 Detailed Description . . . 853
14.47 cxx::String Class Reference . . . 854
14.47.1 Detailed Description . . . 856
14.47.2 Constructor & Destructor Documentation . . . 857
14.47.2.1 String() . . . 857
14.47.3 Member Function Documentation . . . 857
14.47.3.1 find() . . . 857
14.47.3.2 from_dec() . . . 858
14.47.3.3 from_hex() . . . 859
14.47.3.4 starts_with() . . . 860
14.48 cxx::Weak_ref< T > Class Template Reference . . . 861
14.48.1 Detailed Description . . . 862
14.49 cxx::Weak_ref_base Class Reference . . . 863
14.49.1 Detailed Description . . . 865
14.50 Elf32_Dyn Struct Reference . . . 866
14.50.1 Detailed Description . . . 866
14.50.2 Field Documentation . . . 866
14.50.2.1 d_val . . . 866
14.51 Elf32_Ehdr Struct Reference . . . 867
14.51.1 Detailed Description . . . 868
14.51.2 Field Documentation . . . 868
14.51.2.1 e_phnum . . . 868
14.51.2.2 e_shnum . . . 868
14.52 Elf32_Phdr Struct Reference . . . 869
14.52.1 Detailed Description . . . 869
14.53 Elf32_Shdr Struct Reference . . . 870
14.53.1 Detailed Description . . . 871
14.54 Elf32_Sym Struct Reference . . . 871
14.54.1 Detailed Description . . . 872

14.55 Elf64_Dyn Struct Reference . . . 872
14.55.1 Detailed Description . . . 872
14.55.2 Field Documentation . . . 873
14.55.2.1 d_val . . . 873
14.56 Elf64_Ehdr Struct Reference . . . 873
14.56.1 Detailed Description . . . 874
14.56.2 Field Documentation . . . 874
14.56.2.1 e_phnum . . . 874
14.56.2.2 e_shnum . . . 875
14.57 Elf64_Phdr Struct Reference . . . 875
14.57.1 Detailed Description . . . 876
14.58 Elf64_Shdr Struct Reference . . . 876
14.58.1 Detailed Description . . . 877
14.59 Elf64_Sym Struct Reference . . . 877
14.59.1 Detailed Description . . . 878
14.60 gfxbitmap_offset Struct Reference . . . 878
14.60.1 Detailed Description . . . 879
14.61 L4::Alloc_list Class Reference . . . 879
14.61.1 Detailed Description . . . 879
14.62 L4::Arm_smccc Class Reference . . . 880
14.62.1 Detailed Description . . . 880
14.62.2 Member Function Documentation . . . 880
14.62.2.1 call() . . . 880
14.63 L4::Base_exception Class Reference . . . 881
14.63.1 Detailed Description . . . 882
14.64 L4::Basic_registry Class Reference . . . 883
14.64.1 Detailed Description . . . 884
14.64.2 Member Function Documentation . . . 884
14.64.2.1 dispatch() . . . 884
14.64.2.2 find() . . . 884

14.65 L4::Bounds_error Class Reference . . . 885
14.65.1 Detailed Description . . . 887
14.66 L4::Cap< T > Class Template Reference . . . 888
14.66.1 Detailed Description . . . 890
14.66.2 Constructor & Destructor Documentation . . . 890
14.66.2.1 Cap() [1/4] . . . 890
14.66.2.2 Cap() [2/4] . . . 891
14.66.2.3 Cap() [3/4] . . . 891
14.66.2.4 Cap() [4/4] . . . 891
14.66.3 Member Function Documentation . . . 892
14.66.3.1 copy() . . . 892
14.66.3.2 move() . . . 892
14.67 L4::Cap_base Class Reference . . . 893
14.67.1 Detailed Description . . . 894
14.67.2 Member Enumeration Documentation . . . 895
14.67.2.1 Cap_type . . . 895
14.67.2.2 No_init_type . . . 895
14.67.3 Constructor & Destructor Documentation . . . 895
14.67.3.1 Cap_base() [1/2] . . . 895
14.67.3.2 Cap_base() [2/2] . . . 896
14.67.4 Member Function Documentation . . . 896
14.67.4.1 cap() . . . 896
14.67.4.2 copy() . . . 897
14.67.4.3 fpage() . . . 898
14.67.4.4 is_valid() . . . 899
14.67.4.5 move() . . . 900
14.67.4.6 snd_base() . . . 901
14.67.4.7 validate() [1/2] . . . 902
14.67.4.8 validate() [2/2] . . . 903
14.67.5 Field Documentation . . . 903

14.67.5.1 _c . . . 903
14.68 L4::Com_error Class Reference . . . 904
14.68.1 Detailed Description . . . 906
14.68.2 Constructor & Destructor Documentation . . . 906
14.68.2.1 Com_error() . . . 906
14.69 L4::Debugger Class Reference . . . 906
14.69.1 Detailed Description . . . 909
14.69.2 Member Function Documentation . . . 909
14.69.2.1 get_object_name() . . . 909
14.69.2.2 global_id() . . . 910
14.69.2.3 kobj_to_id() . . . 910
14.69.2.4 query_log_name() . . . 910
14.69.2.5 query_log_typeid() . . . 911
14.69.2.6 set_object_name() . . . 912
14.69.2.7 switch_log() . . . 912
14.70 L4::Element_already_exists Class Reference . . . 913
14.70.1 Detailed Description . . . 914
14.71 L4::Element_not_found Class Reference . . . 915
14.71.1 Detailed Description . . . 916
14.72 L4::Exception Class Reference . . . 917
14.72.1 Detailed Description . . . 917
14.72.2 Member Typedef Documentation . . . 917
14.72.2.1 Rpcs . . . 917
14.73 L4::Exception_tracer Class Reference . . . 918
14.73.1 Detailed Description . . . 919
14.74 L4::Factory Class Reference . . . 919
14.74.1 Detailed Description . . . 922
14.74.2 Member Function Documentation . . . 923
14.74.2.1 create() [1/2] . . . 923
14.74.2.2 create() [2/2] . . . 924

14.74.2.3 create_factory() . . . 925
14.74.2.4 create_gate() . . . 926
14.74.2.5 create_irq() . . . 927
14.74.2.6 create_task() . . . 928
14.74.2.7 create_thread() . . . 929
14.74.2.8 create_vm() . . . 930
14.75 L4::Factory::Lstr Struct Reference . . . 931
14.75.1 Detailed Description . . . 932
14.75.2 Constructor & Destructor Documentation . . . 932
14.75.2.1 Lstr() . . . 932
14.76 L4::Factory::Nil Struct Reference . . . 932
14.76.1 Detailed Description . . . 933
14.77 L4::Factory::S Class Reference . . . 933
14.77.1 Detailed Description . . . 934
14.77.2 Constructor & Destructor Documentation . . . 934
14.77.2.1 S() [1/2] . . . 934
14.77.2.2 S() [2/2] . . . 934
14.77.3 Member Function Documentation . . . 935
14.77.3.1 operator l4_msgtag_t() . . . 935
14.77.3.2 operator<<() [1/6] . . . 935
14.77.3.3 operator<<() [2/6] . . . 936
14.77.3.4 operator<<() [3/6] . . . 936
14.77.3.5 operator<<() [4/6] . . . 936
14.77.3.6 operator<<() [5/6] . . . 937
14.77.3.7 operator<<() [6/6] . . . 937
14.78 L4::Icu Class Reference . . . 938
14.78.1 Detailed Description . . . 940
14.78.2 Member Function Documentation . . . 940
14.78.2.1 bind() . . . 940
14.78.2.2 info() . . . 941

14.78.2.3 mask() . . . 942
14.78.2.4 msi_info() . . . 943
14.78.2.5 set_mode() . . . 944
14.78.2.6 unbind() . . . 944
14.79 L4::Icu::Info Class Reference . . . 945
14.79.1 Detailed Description . . . 946
14.80 L4::Invalid_capability Class Reference . . . 946
14.80.1 Detailed Description . . . 948
14.80.2 Constructor & Destructor Documentation . . . 949
14.80.2.1 Invalid_capability() . . . 949
14.80.3 Member Function Documentation . . . 949
14.80.3.1 cap() . . . 949
14.81 L4::Io_pager Class Reference . . . 950
14.81.1 Detailed Description . . . 951
14.81.2 Member Function Documentation . . . 951
14.81.2.1 io_page_fault() . . . 951
14.82 L4::Iommu Class Reference . . . 952
14.82.1 Detailed Description . . . 953
14.82.2 Member Function Documentation . . . 954
14.82.2.1 bind() . . . 954
14.82.2.2 unbind() . . . 954
14.83 L4::IOModifier Class Reference . . . 954
14.83.1 Detailed Description . . . 955
14.84 L4::Ipc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference . . . 955
14.84.1 Detailed Description . . . 957
14.85 L4::Ipc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference . . . 958
14.85.1 Detailed Description . . . 959
14.86 L4::Ipc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference . . . 959
14.86.1 Detailed Description . . . 961
14.87 L4::Ipc::As_value< T > Struct Template Reference . . . 961

14.87.1 Detailed Description 962
14.88 L4::Ipc::Buf_item Class Reference 962
14.88.1 Detailed Description 963
14.89 L4::Ipc::Call Struct Reference 963
14.89.1 Detailed Description 964
14.90 L4::Ipc::Call_t< RIGHTS > Struct Template Reference 964
14.90.1 Detailed Description 965
14.91 L4::Ipc::Call_zero_send_timeout Struct Reference 966
14.91.1 Detailed Description 967
14.92 L4::Ipc::Cap< T > Class Template Reference 967
14.92.1 Detailed Description 969
14.92.2 Member Enumeration Documentation 969
14.92.2.1 anonymous enum 969
14.92.3 Constructor & Destructor Documentation 969
14.92.3.1 Cap() 970
14.92.4 Member Function Documentation 970
14.92.4.1 from_ci() 970
14.93 L4::Ipc::Gen_fpage< T > Class Template Reference 970
14.93.1 Detailed Description 972
14.93.2 Member Function Documentation 972
14.93.2.1 cap_received() 972
14.93.2.2 id_received() 973
14.93.2.3 is_compound() 973
14.93.2.4 local_id_received() 973
14.94 L4::Ipc::In_out< T > Struct Template Reference 974
14.94.1 Detailed Description 974
14.95 L4::Ipc::Iostream Class Reference 974
14.95.1 Detailed Description 977
14.95.2 Constructor & Destructor Documentation 977
14.95.2.1 Iostream() 977

14.95.3 Member Function Documentation . . . 978
14.95.3.1 call() . . . 978
14.95.3.2 reply_and_wait() [1/2] . . . 979
14.95.3.3 reply_and_wait() [2/2] . . . 979
14.95.3.4 reset() . . . 980
14.96 L4::Ipc::Istream Class Reference . . . 981
14.96.1 Detailed Description . . . 984
14.96.2 Constructor & Destructor Documentation . . . 984
14.96.2.1 Istream() . . . 985
14.96.3 Member Function Documentation . . . 985
14.96.3.1 get() [1/3] . . . 985
14.96.3.2 get() [2/3] . . . 986
14.96.3.3 get() [3/3] . . . 986
14.96.3.4 receive() . . . 987
14.96.3.5 reset() . . . 988
14.96.3.6 skip() . . . 988
14.96.3.7 tag() [1/2] . . . 989
14.96.3.8 tag() [2/2] . . . 990
14.96.3.9 wait() [1/2] . . . 990
14.96.3.10 wait() [2/2] . . . 990
14.97 L4::Ipc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference . . . 991
14.97.1 Detailed Description . . . 992
14.98 L4::Ipc::Msg::Cls_buffer Struct Reference . . . 992
14.98.1 Detailed Description . . . 993
14.99 L4::Ipc::Msg::Cls_data Struct Reference . . . 994
14.99.1 Detailed Description . . . 994
14.100 L4::Ipc::Msg::Cls_item Struct Reference . . . 995
14.100.1 Detailed Description . . . 995
14.101 L4::Ipc::Msg::Dir_in Struct Reference . . . 996
14.101.1 Detailed Description . . . 996

14.102 L4::Ipc::Msg::Dir_out Struct Reference . . . 997
14.102.1 Detailed Description . . . 997
14.103 L4::Ipc::Msg::Do_in_data Struct Reference . . . 998
14.103.1 Detailed Description . . . 998
14.104 L4::Ipc::Msg::Do_in_items Struct Reference . . . 999
14.104.1 Detailed Description . . . 999
14.105 L4::Ipc::Msg::Do_out_data Struct Reference . . . 1000
14.105.1 Detailed Description . . . 1001
14.106 L4::Ipc::Msg::Do_out_items Struct Reference . . . 1001
14.106.1 Detailed Description . . . 1002
14.107 L4::Ipc::Msg::Do_rcv_buffers Struct Reference . . . 1002
14.107.1 Detailed Description . . . 1003
14.108 L4::Ipc::Msg::Elem< Array< A, LEN > &> Struct Template Reference . . . 1004
14.108.1 Detailed Description . . . 1004
14.109 L4::Ipc::Msg::Elem< Array< A, LEN > > Struct Template Reference . . . 1005
14.109.1 Detailed Description . . . 1005
14.110 L4::Ipc::Msg::Elem< Array_ref< A, LEN > &> Struct Template Reference . . . 1006
14.110.1 Detailed Description . . . 1006
14.111 L4::Ipc::Msg::Is_valid_rpc_type< T > Struct Template Reference . . . 1007
14.111.1 Detailed Description . . . 1008
14.112 L4::Ipc::Msg::Svr_arg_pack< IPC_TYPE > Struct Template Reference . . . 1009
14.112.1 Detailed Description . . . 1009
14.113 L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference . . . 1009
14.113.1 Detailed Description . . . 1010
14.114 L4::Ipc::Msg_ptr< T > Class Template Reference . . . 1010
14.114.1 Detailed Description . . . 1011
14.114.2 Constructor & Destructor Documentation . . . 1011
14.114.2.1 Msg_ptr() . . . 1011
14.115 L4::Ipc::Opt< T > Struct Template Reference . . . 1011
14.115.1 Detailed Description . . . 1013

14.116 L4::Ipc::Ostream Class Reference . . . 1013
14.116.1 Detailed Description . . . 1016
14.116.2 Member Function Documentation . . . 1016
14.116.2.1 put() [1/2] . . . 1016
14.116.2.2 put() [2/2] . . . 1017
14.116.2.3 send() . . . 1017
14.116.2.4 tag() [1/2] . . . 1018
14.116.2.5 tag() [2/2] . . . 1019
14.117 L4::Ipc::Out< T > Struct Template Reference . . . 1019
14.117.1 Detailed Description . . . 1019
14.118 L4::Ipc::Ret_array< T > Struct Template Reference . . . 1020
14.118.1 Detailed Description . . . 1020
14.119 L4::Ipc::Send_only Struct Reference . . . 1021
14.119.1 Detailed Description . . . 1021
14.120 L4::Ipc::Small_buf Class Reference . . . 1021
14.120.1 Detailed Description . . . 1022
14.120.2 Constructor & Destructor Documentation . . . 1022
14.120.2.1 Small_buf() [1/2] . . . 1022
14.120.2.2 Small_buf() [2/2] . . . 1022
14.121 L4::Ipc::Snd_item Class Reference . . . 1023
14.121.1 Detailed Description . . . 1023
14.122 L4::Ipc::Str_cp_in< T > Class Template Reference . . . 1023
14.122.1 Detailed Description . . . 1024
14.122.2 Constructor & Destructor Documentation . . . 1024
14.122.2.1 Str_cp_in() . . . 1024
14.123 L4::Ipc::Varg Class Reference . . . 1025
14.123.1 Detailed Description . . . 1026
14.123.2 Member Function Documentation . . . 1026
14.123.2.1 data() . . . 1026
14.123.2.2 get_value() . . . 1026

14.123.2.3 is_nil() . . . 1027
14.123.2.4 is_of() . . . 1027
14.123.2.5 is_of_int() . . . 1027
14.123.2.6 length() . . . 1028
14.123.2.7 tag() . . . 1028
14.123.2.8 type() . . . 1029
14.123.2.9 value() . . . 1029
14.124 L4::Ipc::Varg_list< MAX > Class Template Reference . . . 1030
14.124.1 Detailed Description . . . 1031
14.125 L4::Ipc::Varg_list_ref Class Reference . . . 1031
14.125.1 Detailed Description . . . 1032
14.125.2 Constructor & Destructor Documentation . . . 1032
14.125.2.1 Varg_list_ref() . . . 1032
14.126 L4::Ipc_gate Class Reference . . . 1033
14.126.1 Detailed Description . . . 1036
14.126.2 Member Function Documentation . . . 1036
14.126.2.1 get_infos() . . . 1036
14.127 L4::Ipc_svr::Br_manager_no_buffers Class Reference . . . 1036
14.127.1 Detailed Description . . . 1039
14.127.2 Member Function Documentation . . . 1039
14.127.2.1 alloc_buffer_demand() . . . 1039
14.128 L4::Ipc_svr::Compound_reply Struct Reference . . . 1040
14.128.1 Detailed Description . . . 1040
14.129 L4::Ipc_svr::Default_loop_hooks Struct Reference . . . 1041
14.129.1 Detailed Description . . . 1042
14.130 L4::Ipc_svr::Default_setup_wait Struct Reference . . . 1042
14.130.1 Detailed Description . . . 1043
14.131 L4::Ipc_svr::Default_timeout Struct Reference . . . 1043
14.131.1 Detailed Description . . . 1044
14.132 L4::Ipc_svr::Direct_dispatch< R > Struct Template Reference . . . 1044

14.132.1 Detailed Description . . . 1045
14.133 L4::Ipc_svr::Direct_dispatch< R * > Struct Template Reference . . . 1046
14.133.1 Detailed Description . . . 1046
14.134 L4::Ipc_svr::Exc_dispatch< R, Exc > Struct Template Reference . . . 1047
14.134.1 Detailed Description . . . 1048
14.135 L4::Ipc_svr::Ignore_errors Struct Reference . . . 1048
14.135.1 Detailed Description . . . 1049
14.136 L4::Ipc_svr::Server_iface Class Reference . . . 1049
14.136.1 Detailed Description . . . 1052
14.136.2 Member Function Documentation . . . 1052
14.136.2.1 add_timeout() . . . 1052
14.136.2.2 alloc_buffer_demand() . . . 1053
14.136.2.3 get_rcv_cap() . . . 1053
14.136.2.4 rcv_cap() [1/2] . . . 1054
14.136.2.5 rcv_cap() [2/2] . . . 1055
14.136.2.6 realloc_rcv_cap() . . . 1056
14.136.2.7 remove_timeout() . . . 1057
14.137 L4::Ipc_svr::Timed_work< HOOKS > Class Template Reference . . . 1058
14.137.1 Detailed Description . . . 1058
14.138 L4::Ipc_svr::Timeout Class Reference . . . 1058
14.138.1 Detailed Description . . . 1060
14.138.2 Member Function Documentation . . . 1060
14.138.2.1 expired() . . . 1060
14.138.2.2 timeout() . . . 1061
14.139 L4::Ipc_svr::Timeout_queue Class Reference . . . 1061
14.139.1 Detailed Description . . . 1062
14.139.2 Member Function Documentation . . . 1062
14.139.2.1 add() . . . 1062
14.139.2.2 handle_expired_timeouts() . . . 1063
14.139.2.3 next_timeout() . . . 1063

14.139.2.4 remove() . . . 1063
14.139.2.5 timeout_expired() . . . 1064
14.140 L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference . . . 1065
14.140.1 Detailed Description . . . 1067
14.140.2 Member Function Documentation . . . 1067
14.140.2.1 add_timeout() . . . 1067
14.140.2.2 remove_timeout() . . . 1068
14.141 L4::Irq Class Reference . . . 1068
14.141.1 Detailed Description . . . 1071
14.141.2 Member Function Documentation . . . 1071
14.141.2.1 attach() . . . 1071
14.141.2.2 detach() . . . 1072
14.141.2.3 receive() . . . 1073
14.141.2.4 unmask() . . . 1074
14.141.2.5 wait() . . . 1075
14.142 L4::Irq_eoi Class Reference . . . 1075
14.142.1 Detailed Description . . . 1077
14.142.2 Member Function Documentation . . . 1077
14.142.2.1 unmask() . . . 1077
14.143 L4::Irq_handler_object Struct Reference . . . 1078
14.143.1 Detailed Description . . . 1080
14.144 L4::Irq_mux Struct Reference . . . 1081
14.144.1 Detailed Description . . . 1083
14.144.2 Member Function Documentation . . . 1083
14.144.2.1 chain() . . . 1083
14.145 L4::Kip::Mem_desc Class Reference . . . 1084
14.145.1 Detailed Description . . . 1085
14.145.2 Member Enumeration Documentation . . . 1086
14.145.2.1 Info_sub_type . . . 1086
14.145.2.2 Mem_type . . . 1086

14.145.3 Constructor & Destructor Documentation . . . 1086
14.145.3.1 Mem_desc() . . . 1086
14.145.4 Member Function Documentation . . . 1087
14.145.4.1 all() [1/2] . . . 1087
14.145.4.2 all() [2/2] . . . 1088
14.145.4.3 count() [1/2] . . . 1088
14.145.4.4 count() [2/2] . . . 1089
14.145.4.5 end() . . . 1090
14.145.4.6 first() . . . 1090
14.145.4.7 is_virtual() . . . 1091
14.145.4.8 set() . . . 1091
14.145.4.9 size() . . . 1092
14.145.4.10 start() . . . 1092
14.145.4.11 sub_type() . . . 1093
14.145.4.12 type() . . . 1093
14.146 L4::Kobject Class Reference . . . 1093
14.146.1 Detailed Description . . . 1094
14.146.2 Member Function Documentation . . . 1094
14.146.2.1 cap() . . . 1094
14.146.2.2 dec_refcnt() . . . 1096
14.147 L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference . . . 1097
14.147.1 Detailed Description . . . 1098
14.147.2 Member Typedef Documentation . . . 1099
14.147.2.1 __Iface . . . 1099
14.147.2.2 __Iface_list . . . 1099
14.147.2.3 Class . . . 1100
14.147.3 Member Function Documentation . . . 1100
14.147.3.1 __check_protocols__() . . . 1100
14.147.3.2 c() . . . 1100
14.148 L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference 1101

14.148.1 Detailed Description . . . 1102
14.148.2 Member Typedef Documentation . . . 1103
14.148.2.1 __Iface . . . 1103
14.148.2.2 __Iface_list . . . 1103
14.148.2.3 Class . . . 1104
14.148.3 Member Function Documentation . . . 1104
14.148.3.1 __check_protocols__() . . . 1104
14.148.3.2 c() . . . 1104
14.149 L4::Kobject_demand< T > Struct Template Reference . . . 1104
14.149.1 Detailed Description . . . 1105
14.150 L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference . . . 1105
14.150.1 Detailed Description . . . 1106
14.151 L4::Kobject_typeid< T > Struct Template Reference . . . 1107
14.151.1 Detailed Description . . . 1107
14.151.2 Member Typedef Documentation . . . 1108
14.151.2.1 Demand . . . 1108
14.151.3 Member Function Documentation . . . 1108
14.151.3.1 demand() . . . 1108
14.151.3.2 d() . . . 1109
14.151.3.3 proto_dispatch() . . . 1109
14.152 L4::Kobject_typeid< void > Struct Template Reference . . . 1110
14.152.1 Detailed Description . . . 1111
14.153 L4::Kobject_x< Derived, ARGS > Struct Template Reference . . . 1111
14.153.1 Detailed Description . . . 1112
14.154 L4::Meta Class Reference . . . 1112
14.154.1 Detailed Description . . . 1114
14.154.2 Member Function Documentation . . . 1115
14.154.2.1 interface() . . . 1115
14.154.2.2 num_interfaces() . . . 1115
14.154.2.3 supports() . . . 1115

14.155 L4::Out_of_memory Class Reference . . . 1116
14.155.1 Detailed Description . . . 1119
14.156 L4::Pager Class Reference . . . 1119
14.156.1 Detailed Description . . . 1121
14.156.2 Member Function Documentation . . . 1121
14.156.2.1 page_fault() . . . 1122
14.157 L4::Platform_control Class Reference . . . 1122
14.157.1 Detailed Description . . . 1125
14.157.2 Member Enumeration Documentation . . . 1125
14.157.2.1 Opcode . . . 1125
14.157.3 Member Function Documentation . . . 1125
14.157.3.1 cpu_disable() . . . 1125
14.157.3.2 cpu_enable() . . . 1126
14.157.3.3 system_shutdown() . . . 1126
14.157.3.4 system_suspend() . . . 1126
14.158 L4::Poll_timeout_kipclock Class Reference . . . 1127
14.158.1 Detailed Description . . . 1128
14.158.2 Constructor & Destructor Documentation . . . 1128
14.158.2.1 Poll_timeout_kipclock() . . . 1128
14.158.3 Member Function Documentation . . . 1128
14.158.3.1 set() . . . 1128
14.158.3.2 test() . . . 1129
14.158.3.3 timed_out() . . . 1130
14.159 L4::Proto_t< P > Struct Template Reference . . . 1130
14.159.1 Detailed Description . . . 1131
14.160 L4::Rcv_endpoint Class Reference . . . 1132
14.160.1 Detailed Description . . . 1134
14.160.2 Member Function Documentation . . . 1135
14.160.2.1 bind_thread() . . . 1135
14.161 L4::Registry_iface Class Reference . . . 1136

14.161.1 Detailed Description . . . 1137
14.161.2 Member Function Documentation . . . 1137
14.161.2.1 register_irq_obj() . . . 1137
14.161.2.2 register_obj() [1/3] . . . 1138
14.161.2.3 register_obj() [2/3] . . . 1139
14.161.2.4 register_obj() [3/3] . . . 1140
14.161.2.5 unregister_obj() . . . 1140
14.162 L4::Runtime_error Class Reference . . . 1141
14.162.1 Detailed Description . . . 1142
14.162.2 Constructor & Destructor Documentation . . . 1143
14.162.2.1 Runtime_error() . . . 1143
14.162.3 Member Function Documentation . . . 1143
14.162.3.1 err_no() . . . 1143
14.162.3.2 extra_str() . . . 1143
14.163 L4::Scheduler Class Reference . . . 1144
14.163.1 Detailed Description . . . 1146
14.163.2 Member Function Documentation . . . 1146
14.163.2.1 idle_time() . . . 1146
14.163.2.2 info() . . . 1147
14.163.2.3 is_online() . . . 1148
14.163.2.4 run_thread() . . . 1148
14.164 L4::Semaphore Struct Reference . . . 1149
14.164.1 Detailed Description . . . 1152
14.164.2 Member Function Documentation . . . 1152
14.164.2.1 down() . . . 1152
14.164.2.2 up() . . . 1153
14.165 L4::Server< LOOP_HOOKS > Class Template Reference . . . 1154
14.165.1 Detailed Description . . . 1155
14.165.2 Constructor & Destructor Documentation . . . 1155
14.165.2.1 Server() . . . 1155

14.165.3 Member Function Documentation . . . 1156
14.165.3.1 internal_loop() . . . 1156
14.165.3.2 loop() . . . 1157
14.166 L4::Server_object Class Reference . . . 1157
14.166.1 Detailed Description . . . 1158
14.166.2 Member Function Documentation . . . 1158
14.166.2.1 dispatch() . . . 1158
14.167 L4::Server_object_t< IFACE, BASE > Struct Template Reference . . . 1159
14.167.1 Detailed Description . . . 1161
14.167.2 Member Function Documentation . . . 1161
14.167.2.1 dispatch_meta_request() . . . 1161
14.167.2.2 get_buffer_demand() . . . 1162
14.167.2.3 proto_dispatch() . . . 1162
14.168 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference . . . 1163
14.168.1 Detailed Description . . . 1164
14.169 L4::Smart_cap< T, SMART > Class Template Reference . . . 1165
14.169.1 Detailed Description . . . 1168
14.169.2 Constructor & Destructor Documentation . . . 1168
14.169.2.1 Smart_cap() . . . 1168
14.170 L4::String Class Reference . . . 1169
14.170.1 Detailed Description . . . 1169
14.171 L4::Task Class Reference . . . 1169
14.171.1 Detailed Description . . . 1172
14.171.2 Member Function Documentation . . . 1172
14.171.2.1 add_ku_mem() . . . 1172
14.171.2.2 cap_equal() . . . 1173
14.171.2.3 cap_has_child() . . . 1173
14.171.2.4 cap_valid() . . . 1174
14.171.2.5 delete_obj() . . . 1174
14.171.2.6 map() . . . 1175

14.171.2.7 release_cap() . . . 1175
14.171.2.8 unmap() . . . 1176
14.171.2.9 unmap_batch() . . . 1177
14.172 L4::Thread Class Reference . . . 1177
14.172.1 Detailed Description . . . 1180
14.172.2 Member Function Documentation . . . 1181
14.172.2.1 control() . . . 1181
14.172.2.2 ex_regs() [1/2] . . . 1181
14.172.2.3 ex_regs() [2/2] . . . 1182
14.172.2.4 modify_senders() . . . 1183
14.172.2.5 register_del_irq() . . . 1184
14.172.2.6 stats_time() . . . 1184
14.172.2.7 switch_to() . . . 1185
14.172.2.8 vcpu_control() . . . 1185
14.172.2.9 vcpu_control_ext() . . . 1186
14.172.2.10 vcpu_resume_commit() . . . 1187
14.172.2.11 vcpu_resume_start() . . . 1187
14.173 L4::Thread::Attr Class Reference . . . 1188
14.173.1 Detailed Description . . . 1189
14.173.2 Constructor & Destructor Documentation . . . 1189
14.173.2.1 Attr() . . . 1189
14.173.3 Member Function Documentation . . . 1189
14.173.3.1 bind() . . . 1189
14.173.3.2 exc_handler() [1/2] . . . 1190
14.173.3.3 exc_handler() [2/2] . . . 1190
14.173.3.4 pager() [1/2] . . . 1190
14.173.3.5 pager() [2/2] . . . 1191
14.173.3.6 ux_host_syscall() . . . 1191
14.174 L4::Thread::Modify_senders Class Reference . . . 1191
14.174.1 Detailed Description . . . 1192

14.174.2 Member Function Documentation . . . 1192
14.174.2.1 add() . . . 1192
14.175 L4::Triggerable Struct Reference . . . 1193
14.175.1 Detailed Description . . . 1195
14.175.2 Member Function Documentation . . . 1195
14.175.2.1 trigger() . . . 1196
14.176 L4::Type_info Struct Reference . . . 1197
14.176.1 Detailed Description . . . 1197
14.177 L4::Type_info::Demand Class Reference . . . 1198
14.177.1 Detailed Description . . . 1199
14.177.2 Constructor & Destructor Documentation . . . 1199
14.177.2.1 Demand() . . . 1200
14.177.3 Member Function Documentation . . . 1200
14.177.3.1 no_demand() . . . 1200
14.178 L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference . . . 1201
14.178.1 Detailed Description . . . 1202
14.178.2 Member Enumeration Documentation . . . 1203
14.178.2.1 anonymous enum . . . 1203
14.179 L4::Type_info::Demand_union_t< D1, D2 > Struct Template Reference . . . 1203
14.179.1 Detailed Description . . . 1205
14.180 L4::Typeid::Detail::_Rpcs< OPCODE, O, X > Struct Template Reference . . . 1206
14.180.1 Detailed Description . . . 1207
14.181 L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference . . . 1207
14.181.1 Detailed Description . . . 1208
14.182 L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... > Struct Template Reference . . . 1208
14.182.1 Detailed Description . . . 1209
14.183 L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference . . . 1209
14.183.1 Detailed Description . . . 1210
14.184 L4::Typeid::Detail::Rpcs_end Struct Reference . . . 1210
14.184.1 Detailed Description . . . 1211

14.185 L4::Typeid::P_dispatch< LIST > Struct Template Reference . . . 1211
14.185.1 Detailed Description . . . 1211
14.186 L4::Typeid::Raw_ipc< CLASS > Struct Template Reference . . . 1212
14.186.1 Detailed Description . . . 1212
14.187 L4::Typeid::Rpc_nocode< OPERATION > Struct Template Reference . . . 1212
14.187.1 Detailed Description . . . 1214
14.188 L4::Typeid::Rpcs< RPCS > Struct Template Reference . . . 1215
14.188.1 Detailed Description . . . 1216
14.189 L4::Typeid::Rpcs_code< OPCODE_TYPE > Struct Template Reference . . . 1217
14.189.1 Detailed Description . . . 1217
14.190 L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference . . . 1218
14.190.1 Detailed Description . . . 1219
14.191 L4::Typeid::Rpcs_sys< ARG > Struct Template Reference . . . 1220
14.191.1 Detailed Description . . . 1221
14.192 L4::Types::Bool< V > Struct Template Reference . . . 1222
14.192.1 Detailed Description . . . 1223
14.193 L4::Types::False Struct Reference . . . 1223
14.193.1 Detailed Description . . . 1225
14.194 L4::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference . . . 1225
14.194.1 Detailed Description . . . 1227
14.194.2 Member Enumeration Documentation . . . 1228
14.194.2.1 None_type . . . 1228
14.194.3 Constructor & Destructor Documentation . . . 1228
14.194.3.1 Flags() [1/2] . . . 1228
14.194.3.2 Flags() [2/2] . . . 1229
14.194.4 Member Function Documentation . . . 1229
14.194.4.1 clear() . . . 1229
14.194.4.2 from_raw() . . . 1230
14.195 L4::Types::Same< A, B > Struct Template Reference . . . 1230
14.195.1 Detailed Description . . . 1231

14.196 L4::Types::True Struct Reference . . . 1232
14.196.1 Detailed Description . . . 1234
14.197 L4::Unknown_error Class Reference . . . 1234
14.197.1 Detailed Description . . . 1236
14.198 L4::Vcon Class Reference . . . 1237
14.198.1 Detailed Description . . . 1239
14.198.2 Member Function Documentation . . . 1239
14.198.2.1 get_attr() . . . 1239
14.198.2.2 read() . . . 1240
14.198.2.3 read_with_flags() . . . 1241
14.198.2.4 send() . . . 1242
14.198.2.5 set_attr() . . . 1242
14.198.2.6 write() . . . 1243
14.199 L4::Vm Class Reference . . . 1244
14.199.1 Detailed Description . . . 1246
14.200 l4_buf_regs_t Struct Reference . . . 1247
14.200.1 Detailed Description . . . 1247
14.201 l4_exc_regs_t Struct Reference . . . 1248
14.201.1 Detailed Description . . . 1250
14.201.2 Field Documentation . . . 1250
14.201.2.1 flags . . . 1250
14.201.2.2 ss . . . 1250
14.202 l4_fpage_t Union Reference . . . 1251
14.202.1 Detailed Description . . . 1251
14.203 l4_icu_info_t Struct Reference . . . 1251
14.203.1 Detailed Description . . . 1253
14.203.2 Field Documentation . . . 1253
14.203.2.1 features . . . 1253
14.204 l4_icu_msi_info_t Struct Reference . . . 1253
14.204.1 Detailed Description . . . 1254

14.204.2 Field Documentation . . . 1254
14.204.2.1 msi_data . . . 1254
14.205 l4_kernel_info_mem_desc_t Struct Reference . . . 1254
14.205.1 Detailed Description . . . 1255
14.206 l4_kernel_info_t Struct Reference . . . 1255
14.206.1 Detailed Description . . . 1257
14.207 l4_msg_regs_t Union Reference . . . 1257
14.207.1 Detailed Description . . . 1258
14.208 l4_msgtag_t Struct Reference . . . 1258
14.208.1 Detailed Description . . . 1259
14.208.2 Member Function Documentation . . . 1260
14.208.2.1 flags() . . . 1260
14.209 l4_sched_cpu_set_t Struct Reference . . . 1260
14.209.1 Detailed Description . . . 1261
14.209.2 Member Function Documentation . . . 1261
14.209.2.1 granularity() . . . 1261
14.209.2.2 offset() . . . 1262
14.209.3 Field Documentation . . . 1262
14.209.3.1 gran_offset . . . 1262
14.210 l4_sched_param_t Struct Reference . . . 1263
14.210.1 Detailed Description . . . 1263
14.211 l4_snd_fpage_t Struct Reference . . . 1264
14.211.1 Detailed Description . . . 1264
14.212 l4_thread_regs_t Struct Reference . . . 1265
14.212.1 Detailed Description . . . 1265
14.213 l4_timeout_s Struct Reference . . . 1266
14.213.1 Detailed Description . . . 1266
14.214 l4_timeout_t Union Reference . . . 1267
14.214.1 Detailed Description . . . 1267
14.215 l4_tracebuffer_status_t Struct Reference . . . 1268

14.215.1 Detailed Description . . . 1269
14.215.2 Field Documentation . . . 1269
14.215.2.1 cnt_iobmap_tlb_flush . . . 1269
14.216 l4_tracebuffer_status_window_t Struct Reference . . . 1270
14.216.1 Detailed Description . . . 1270
14.217 l4_vcon_attr_t Struct Reference . . . 1271
14.217.1 Detailed Description . . . 1271
14.218 l4_vcpu_ipc_regs_t Struct Reference . . . 1272
14.218.1 Detailed Description . . . 1272
14.219 l4_vcpu_regs_t Struct Reference . . . 1273
14.219.1 Detailed Description . . . 1274
14.219.2 Field Documentation . . . 1274
14.219.2.1 ax . . . 1275
14.219.2.2 bp . . . 1275
14.219.2.3 bx . . . 1275
14.219.2.4 cx . . . 1275
14.219.2.5 di . . . 1276
14.219.2.6 dx . . . 1276
14.219.2.7 si . . . 1276
14.220 l4_vcpu_state_t Struct Reference . . . 1277
14.220.1 Detailed Description . . . 1279
14.221 l4_vhw_descriptor Struct Reference . . . 1279
14.221.1 Detailed Description . . . 1280
14.221.2 Field Documentation . . . 1281
14.221.2.1 count . . . 1281
14.221.2.2 descs . . . 1281
14.221.2.3 magic . . . 1281
14.221.2.4 version . . . 1282
14.222 l4_vhw_entry Struct Reference . . . 1282
14.222.1 Detailed Description . . . 1283

14.222.2 Field Documentation . . . 1283
14.222.2.1 fd . . . 1283
14.222.2.2 irq_no . . . 1283
14.222.2.3 mem_size . . . 1283
14.222.2.4 mem_start . . . 1284
14.222.2.5 provider_pid . . . 1284
14.222.2.6 type . . . 1284
14.223 l4_vm_svm_vmcb_control_area Struct Reference . . . 1285
14.223.1 Detailed Description . . . 1285
14.224 l4_vm_svm_vmcb_state_save_area Struct Reference . . . 1285
14.224.1 Detailed Description . . . 1286
14.225 l4_vm_svm_vmcb_state_save_area_seg Struct Reference . . . 1287
14.225.1 Detailed Description . . . 1287
14.226 l4_vm_svm_vmcb_t Struct Reference . . . 1287
14.226.1 Detailed Description . . . 1288
14.227 l4_vm_tz_state Struct Reference . . . 1289
14.227.1 Detailed Description . . . 1289
14.228 L4Re::Cap_alloc Class Reference . . . 1289
14.228.1 Detailed Description . . . 1290
14.228.2 Member Function Documentation . . . 1290
14.228.2.1 alloc() [1/2] . . . 1291
14.228.2.2 alloc() [2/2] . . . 1291
14.228.2.3 free() . . . 1292
14.228.2.4 get_cap_alloc() . . . 1293
14.229 L4Re::Console Class Reference . . . 1294
14.229.1 Detailed Description . . . 1295
14.230 L4Re::Dataspace Class Reference . . . 1296
14.230.1 Detailed Description . . . 1298
14.230.2 Member Enumeration Documentation . . . 1298
14.230.2.1 Map_flags . . . 1298

14.230.3 Member Function Documentation . . . 1299
14.230.3.1 allocate() . . . 1299
14.230.3.2 clear() . . . 1299
14.230.3.3 copy_in() . . . 1300
14.230.3.4 flags() . . . 1300
14.230.3.5 info() . . . 1302
14.230.3.6 map() . . . 1302
14.230.3.7 map_region() . . . 1303
14.230.3.8 phys() . . . 1304
14.230.3.9 size() . . . 1305
14.231 L4Re::Dataspace::Stats Struct Reference . . . 1305
14.231.1 Detailed Description . . . 1306
14.232 L4Re::Debug_obj Class Reference . . . 1306
14.232.1 Detailed Description . . . 1307
14.232.2 Member Function Documentation . . . 1308
14.232.2.1 debug() . . . 1308
14.233 L4Re::Dma_space Class Reference . . . 1308
14.233.1 Detailed Description . . . 1309
14.233.2 Member Typedef Documentation . . . 1309
14.233.2.1 Attributes . . . 1310
14.233.3 Member Enumeration Documentation . . . 1310
14.233.3.1 Attribute . . . 1310
14.233.3.2 Direction . . . 1310
14.233.3.3 Space_attrib . . . 1311
14.233.4 Member Function Documentation . . . 1311
14.233.4.1 associate() . . . 1311
14.233.4.2 disassociate() . . . 1312
14.233.4.3 map() . . . 1312
14.233.4.4 unmap() . . . 1313
14.234 L4Re::Env Class Reference . . . 1313

14.234.1 Detailed Description . . . 1315
14.234.2 Member Function Documentation . . . 1316
14.234.2.1 env() . . . 1316
14.234.2.2 factory() [1/2] . . . 1317
14.234.2.3 factory() [2/2] . . . 1317
14.234.2.4 first_free_cap() [1/2] . . . 1317
14.234.2.5 first_free_cap() [2/2] . . . 1317
14.234.2.6 first_free_utcb() [1/2] . . . 1318
14.234.2.7 first_free_utcb() [2/2] . . . 1318
14.234.2.8 get() . . . 1318
14.234.2.9 get_cap() [1/2] . . . 1319
14.234.2.10 get_cap() [2/2] . . . 1320
14.234.2.11 initial_caps() [1/2] . . . 1320
14.234.2.12 initial_caps() [2/2] . . . 1320
14.234.2.13 log() [1/2] . . . 1321
14.234.2.14 log() [2/2] . . . 1321
14.234.2.15 main_thread() [1/2] . . . 1321
14.234.2.16 main_thread() [2/2] . . . 1322
14.234.2.17 mem_alloc() [1/2] . . . 1322
14.234.2.18 mem_alloc() [2/2] . . . 1322
14.234.2.19 parent() [1/2] . . . 1323
14.234.2.20 parent() [2/2] . . . 1323
14.234.2.21 rm() [1/2] . . . 1323
14.234.2.22 rm() [2/2] . . . 1324
14.234.2.23 scheduler() [1/2] . . . 1324
14.234.2.24 scheduler() [2/2] . . . 1324
14.234.2.25 task() . . . 1325
14.234.2.26 utcb_area() [1/2] . . . 1325
14.234.2.27 utcb_area() [2/2] . . . 1325
14.235 L4Re::Event Class Reference . . . 1326

14.235.1 Detailed Description . . . 1329
14.235.2 Member Function Documentation . . . 1329
14.235.2.1 get_buffer() . . . 1329
14.236 L4Re::Event_buffer_t< PAYLOAD > Class Template Reference . . . 1330
14.236.1 Detailed Description . . . 1331
14.236.2 Constructor & Destructor Documentation . . . 1331
14.236.2.1 Event_buffer_t() . . . 1331
14.236.3 Member Function Documentation . . . 1332
14.236.3.1 next() . . . 1332
14.236.3.2 put() . . . 1332
14.237 L4Re::Event_buffer_t< PAYLOAD >::Event Struct Reference . . . 1333
14.237.1 Detailed Description . . . 1334
14.238 L4Re::Inhibitor Class Reference . . . 1334
14.238.1 Detailed Description . . . 1337
14.238.2 Member Enumeration Documentation . . . 1337
14.238.2.1 anonymous enum . . . 1337
14.238.3 Member Function Documentation . . . 1337
14.238.3.1 acquire() . . . 1337
14.238.3.2 next_lock_info() . . . 1338
14.238.3.3 release() . . . 1339
14.239 L4Re::Log Class Reference . . . 1339
14.239.1 Detailed Description . . . 1342
14.239.2 Member Function Documentation . . . 1342
14.239.2.1 print() . . . 1342
14.239.2.2 printn() . . . 1342
14.240 L4Re::Mem_alloc Class Reference . . . 1342
14.240.1 Detailed Description . . . 1345
14.240.2 Member Enumeration Documentation . . . 1345
14.240.2.1 Mem_alloc_flags . . . 1345
14.240.3 Member Function Documentation . . . 1346

14.240.3.1 alloc() . . . 1346
14.240.3.2 free() . . . 1347
14.241 L4Re::Mmio_space Struct Reference . . . 1347
14.241.1 Detailed Description . . . 1350
14.241.2 Member Enumeration Documentation . . . 1350
14.241.2.1 Access_width . . . 1350
14.241.3 Member Function Documentation . . . 1350
14.241.3.1 mmio_read() . . . 1350
14.241.3.2 mmio_write() . . . 1351
14.242 L4Re::Namespace Class Reference . . . 1351
14.242.1 Detailed Description . . . 1354
14.242.2 Member Enumeration Documentation . . . 1354
14.242.2.1 Query_result_flags . . . 1354
14.242.2.2 Register_flags . . . 1354
14.242.3 Member Function Documentation . . . 1355
14.242.3.1 query() [1/2] . . . 1355
14.242.3.2 query() [2/2] . . . 1356
14.242.3.3 register_obj() . . . 1357
14.242.3.4 unlink() . . . 1357
14.243 L4Re::Ned::Cmd_control Class Reference . . . 1358
14.243.1 Detailed Description . . . 1359
14.243.2 Member Function Documentation . . . 1359
14.243.2.1 execute() [1/2] . . . 1359
14.243.2.2 execute() [2/2] . . . 1359
14.244 L4Re::Parent Class Reference . . . 1360
14.244.1 Detailed Description . . . 1361
14.244.2 Member Function Documentation . . . 1362
14.244.2.1 signal() . . . 1362
14.245 L4Re::Rm Class Reference . . . 1362
14.245.1 Detailed Description . . . 1365

14.245.2 Member Enumeration Documentation . . . 1366
14.245.2.1 Attach_flags . . . 1366
14.245.2.2 Detach_flags . . . 1366
14.245.2.3 Detach_result . . . 1366
14.245.2.4 Region_flags . . . 1367
14.245.3 Member Function Documentation . . . 1367
14.245.3.1 attach() [1/2] . . . 1367
14.245.3.2 attach() [2/2] . . . 1369
14.245.3.3 detach() [1/3] . . . 1370
14.245.3.4 detach() [2/3] . . . 1371
14.245.3.5 detach() [3/3] . . . 1371
14.245.3.6 find() . . . 1372
14.245.3.7 free_area() . . . 1373
14.245.3.8 reserve_area() [1/2] . . . 1373
14.245.3.9 reserve_area() [2/2] . . . 1374
14.246 L4Re::Smart_cap_auto< Unmap_flags > Class Template Reference . . . 1375
14.246.1 Detailed Description . . . 1376
14.247 L4Re::Smart_count_cap< Unmap_flags > Class Template Reference . . . 1376
14.247.1 Detailed Description . . . 1377
14.248 L4Re::Util::Auto_cap< T > Struct Template Reference . . . 1377
14.248.1 Detailed Description . . . 1377
14.249 L4Re::Util::Auto_del_cap< T > Struct Template Reference . . . 1378
14.249.1 Detailed Description . . . 1378
14.250 L4Re::Util::Br_manager Class Reference . . . 1379
14.250.1 Detailed Description . . . 1382
14.250.2 Member Function Documentation . . . 1382
14.250.2.1 alloc_buffer_demand() . . . 1382
14.250.2.2 get_rcv_cap() . . . 1382
14.250.2.3 realloc_rcv_cap() . . . 1383
14.250.2.4 set_rcv_cap_flags() . . . 1384

14.251 L4Re::Util::Br_manager_hooks Struct Reference . . . 1384
14.251.1 Detailed Description . . . 1385
14.252 L4Re::Util::Br_manager_timeout_hooks Struct Reference . . . 1385
14.252.1 Detailed Description . . . 1387
14.253 L4Re::Util::Cap_alloc_base Class Reference . . . 1388
14.253.1 Detailed Description . . . 1388
14.254 L4Re::Util::Counter< COUNTER > Struct Template Reference . . . 1389
14.254.1 Detailed Description . . . 1389
14.255 L4Re::Util::Counting_cap_alloc< COUNTERTYPE > Class Template Reference . . . 1389
14.255.1 Detailed Description . . . 1391
14.255.2 Constructor & Destructor Documentation . . . 1391
14.255.2.1 Counting_cap_alloc() . . . 1391
14.255.3 Member Function Documentation . . . 1391
14.255.3.1 alloc() [1/2] . . . 1392
14.255.3.2 alloc() [2/2] . . . 1393
14.255.3.3 free() . . . 1393
14.255.3.4 release() . . . 1394
14.255.3.5 setup() . . . 1395
14.255.3.6 take() . . . 1396
14.256 L4Re::Util::Dataspace_svr Class Reference . . . 1396
14.256.1 Detailed Description . . . 1398
14.256.2 Member Function Documentation . . . 1398
14.256.2.1 allocate() . . . 1398
14.256.2.2 clear() . . . 1399
14.256.2.3 copy() . . . 1400
14.256.2.4 is_static() . . . 1401
14.256.2.5 map() . . . 1402
14.256.2.6 map_hook() . . . 1402
14.256.2.7 page_shift() . . . 1403
14.256.2.8 phys() . . . 1404

14.256.2.9 release() . . . 1405
14.256.2.10 take() . . . 1406
14.257 L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference . . . 1406
14.257.1 Detailed Description . . . 1408
14.257.2 Member Function Documentation . . . 1409
14.257.2.1 foreach_available_event() . . . 1409
14.257.2.2 process() . . . 1409
14.258 L4Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference . . . 1410
14.258.1 Detailed Description . . . 1412
14.258.2 Member Function Documentation . . . 1412
14.258.2.1 attach() . . . 1413
14.258.2.2 buf() . . . 1413
14.258.2.3 detach() . . . 1413
14.259 L4Re::Util::Event_t< PAYLOAD > Class Template Reference . . . 1414
14.259.1 Detailed Description . . . 1415
14.259.2 Member Enumeration Documentation . . . 1415
14.259.2.1 Mode . . . 1415
14.259.3 Member Function Documentation . . . 1415
14.259.3.1 buffer() . . . 1415
14.259.3.2 init() . . . 1416
14.259.3.3 init_poll() . . . 1417
14.259.3.4 irq() . . . 1417
14.260 L4Re::Util::Item_alloc_base Class Reference . . . 1418
14.260.1 Detailed Description . . . 1418
14.261 L4Re::Util::Names::Name Class Reference . . . 1419
14.261.1 Detailed Description . . . 1420
14.262 L4Re::Util::Names::Name_space Class Reference . . . 1420
14.262.1 Detailed Description . . . 1421
14.262.2 Member Function Documentation . . . 1422
14.262.2.1 alloc_dynamic_entry() . . . 1422

14.262.2.2 copy_receive_cap() . . . 1422
14.262.2.3 free_capability() . . . 1422
14.262.2.4 free_dynamic_entry() . . . 1423
14.262.2.5 free_epiface() . . . 1423
14.262.2.6 get_epiface() . . . 1423
14.263 L4Re::Util::Object_registry Class Reference . . . 1424
14.263.1 Detailed Description . . . 1426
14.263.2 Constructor & Destructor Documentation . . . 1426
14.263.2.1 Object_registry() [1/2] . . . 1426
14.263.2.2 Object_registry() [2/2] . . . 1426
14.263.3 Member Function Documentation . . . 1427
14.263.3.1 register_irq_obj() . . . 1427
14.263.3.2 register_obj() [1/3] . . . 1428
14.263.3.3 register_obj() [2/3] . . . 1428
14.263.3.4 register_obj() [3/3] . . . 1429
14.263.3.5 unregister_obj() . . . 1429
14.264 L4Re::Util::Ref_cap< T > Struct Template Reference . . . 1430
14.264.1 Detailed Description . . . 1430
14.265 L4Re::Util::Ref_del_cap< T > Struct Template Reference . . . 1431
14.265.1 Detailed Description . . . 1431
14.266 L4Re::Util::Registry_server< LOOP_HOOKS > Class Template Reference . . . 1432
14.266.1 Detailed Description . . . 1434
14.266.2 Constructor & Destructor Documentation . . . 1435
14.266.2.1 Registry_server() [1/2] . . . 1435
14.266.2.2 Registry_server() [2/2] . . . 1435
14.266.3 Member Function Documentation . . . 1435
14.266.3.1 registry() [1/2] . . . 1436
14.266.3.2 registry() [2/2] . . . 1436
14.267 L4Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference . . . 1436
14.267.1 Detailed Description . . . 1437

14.268 L4Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference . . . 1437
14.268.1 Detailed Description . . . 1437
14.269 L4Re::Util::Vcon_svr< SVR > Class Template Reference . . . 1438
14.269.1 Detailed Description . . . 1438
14.270 L4Re::Util::Video::Goos_svr Class Reference . . . 1439
14.270.1 Detailed Description . . . 1440
14.270.2 Member Function Documentation . . . 1440
14.270.2.1 get_fb() . . . 1440
14.270.2.2 init_infos() . . . 1441
14.270.2.3 refresh() . . . 1441
14.270.2.4 screen_info() . . . 1442
14.270.2.5 view_info() . . . 1442
14.271 L4Re::Vfs::Be_file Class Reference . . . 1443
14.271.1 Detailed Description . . . 1445
14.271.2 Member Function Documentation . . . 1446
14.271.2.1 data_space() . . . 1446
14.271.2.2 fstat64() . . . 1446
14.271.2.3 unlock_all_locks() . . . 1447
14.272 L4Re::Vfs::Be_file_system Class Reference . . . 1447
14.272.1 Detailed Description . . . 1448
14.272.2 Constructor & Destructor Documentation . . . 1449
14.272.2.1 Be_file_system() . . . 1449
14.272.2.2 ~Be_file_system() . . . 1449
14.272.3 Member Function Documentation . . . 1449
14.272.3.1 type() . . . 1449
14.273 L4Re::Vfs::Directory Class Reference . . . 1450
14.273.1 Detailed Description . . . 1451
14.273.2 Member Function Documentation . . . 1452
14.273.2.1 faccessat() . . . 1452
14.273.2.2 link() . . . 1452

14.273.2.3 mkdir() . . . 1453
14.273.2.4 rename() . . . 1453
14.273.2.5 rmdir() . . . 1454
14.273.2.6 symlink() . . . 1454
14.273.2.7 unlink() . . . 1454
14.274 L4Re::Vfs::File Class Reference . . . 1455
14.274.1 Detailed Description . . . 1456
14.275 L4Re::Vfs::File_system Class Reference . . . 1457
14.275.1 Detailed Description . . . 1458
14.275.2 Member Function Documentation . . . 1458
14.275.2.1 mount() . . . 1458
14.275.2.2 type() . . . 1459
14.276 L4Re::Vfs::Fs Class Reference . . . 1459
14.276.1 Detailed Description . . . 1461
14.276.2 Member Function Documentation . . . 1462
14.276.2.1 alloc_fd() . . . 1462
14.276.2.2 free_fd() . . . 1462
14.276.2.3 get_file() . . . 1463
14.276.2.4 mount() . . . 1463
14.276.2.5 set_fd() . . . 1463
14.277 L4Re::Vfs::Generic_file Class Reference . . . 1464
14.277.1 Detailed Description . . . 1466
14.277.2 Member Function Documentation . . . 1466
14.277.2.1 fchmod() . . . 1467
14.277.2.2 fstat64() . . . 1467
14.277.2.3 get_status_flags() . . . 1467
14.277.2.4 set_status_flags() . . . 1467
14.277.2.5 unlock_all_locks() . . . 1468
14.278 L4Re::Vfs::Mman Class Reference . . . 1469
14.278.1 Detailed Description . . . 1470

14.279 L4Re::Vfs::Ops Class Reference . . . 1470
14.279.1 Detailed Description . . . 1472
14.280 L4Re::Vfs::Regular_file Class Reference . . . 1473
14.280.1 Detailed Description . . . 1474
14.280.2 Member Function Documentation . . . 1475
14.280.2.1 data_space() . . . 1475
14.280.2.2 fdatasync() . . . 1475
14.280.2.3 fsync() . . . 1475
14.280.2.4 ftruncate64() . . . 1475
14.280.2.5 get_lock() . . . 1476
14.280.2.6 lseek64() . . . 1476
14.280.2.7 readv() . . . 1477
14.280.2.8 set_lock() . . . 1477
14.280.2.9 writev() . . . 1477
14.281 L4Re::Vfs::Special_file Class Reference . . . 1478
14.281.1 Detailed Description . . . 1480
14.281.2 Member Function Documentation . . . 1480
14.281.2.1 ioctl() . . . 1480
14.282 L4Re::Video::Color_component Class Reference . . . 1481
14.282.1 Detailed Description . . . 1482
14.282.2 Constructor & Destructor Documentation . . . 1482
14.282.2.1 Color_component() . . . 1482
14.282.3 Member Function Documentation . . . 1482
14.282.3.1 dump() . . . 1482
14.282.3.2 get() . . . 1483
14.282.3.3 operator==() . . . 1483
14.282.3.4 set() . . . 1483
14.282.3.5 shift() . . . 1484
14.282.3.6 size() . . . 1484
14.283 L4Re::Video::Goos Class Reference . . . 1485

14.283.1 Detailed Description . . . 1487
14.283.2 Member Enumeration Documentation . . . 1487
14.283.2.1 Flags . . . 1487
14.283.3 Member Function Documentation . . . 1488
14.283.3.1 create_buffer() . . . 1488
14.283.3.2 create_view() . . . 1488
14.283.3.3 delete_buffer() . . . 1489
14.283.3.4 delete_view() . . . 1489
14.283.3.5 get_static_buffer() . . . 1489
14.283.3.6 info() . . . 1490
14.283.3.7 view() . . . 1490
14.284 L4Re::Video::Goos::Info Struct Reference . . . 1491
14.284.1 Detailed Description . . . 1492
14.284.2 Member Function Documentation . . . 1492
14.284.2.1 auto_refresh() . . . 1492
14.285 L4Re::Video::Pixel_info Class Reference . . . 1493
14.285.1 Detailed Description . . . 1494
14.285.2 Constructor & Destructor Documentation . . . 1494
14.285.2.1 Pixel_info() [1/2] . . . 1494
14.285.2.2 Pixel_info() [2/2] . . . 1495
14.285.3 Member Function Documentation . . . 1495
14.285.3.1 a() [1/2] . . . 1495
14.285.3.2 a() [2/2] . . . 1495
14.285.3.3 b() [1/2] . . . 1496
14.285.3.4 b() [2/2] . . . 1496
14.285.3.5 bits_per_pixel() . . . 1496
14.285.3.6 bytes_per_pixel() [1/2] . . . 1497
14.285.3.7 bytes_per_pixel() [2/2] . . . 1497
14.285.3.8 dump() . . . 1497
14.285.3.9 g() [1/2] . . . 1498

14.285.3.10 () [2/2] . . . 1498
14.285.3.11 has_alpha() . . . 1499
14.285.3.12 operator==() . . . 1499
14.285.3.13 () [1/2] . . . 1500
14.285.3.14 () [2/2] . . . 1500
14.286 L4Re::Video::View Class Reference . . . 1500
14.286.1 Detailed Description . . . 1502
14.286.2 Member Enumeration Documentation . . . 1502
14.286.2.1 Flags . . . 1502
14.286.2.2 V_flags . . . 1502
14.286.3 Member Function Documentation . . . 1503
14.286.3.1 info() . . . 1503
14.286.3.2 refresh() . . . 1503
14.286.3.3 set_info() . . . 1504
14.286.3.4 set_viewport() . . . 1504
14.286.3.5 stack() . . . 1505
14.287 L4Re::Video::View::Info Struct Reference . . . 1505
14.287.1 Detailed Description . . . 1507
14.288 l4re_aux_t Struct Reference . . . 1508
14.288.1 Detailed Description . . . 1508
14.289 l4re_ds_stats_t Struct Reference . . . 1509
14.289.1 Detailed Description . . . 1509
14.290 l4re_elf_aux_mword_t Struct Reference . . . 1509
14.290.1 Detailed Description . . . 1510
14.291 l4re_elf_aux_t Struct Reference . . . 1510
14.291.1 Detailed Description . . . 1511
14.292 l4re_elf_aux_vma_t Struct Reference . . . 1511
14.292.1 Detailed Description . . . 1511
14.293 l4re_env_cap_entry_t Struct Reference . . . 1512
14.293.1 Detailed Description . . . 1512

14.293.2 Constructor & Destructor Documentation . . . 1512
14.293.2.1 l4re_env_cap_entry_t() . . . 1513
14.293.3 Field Documentation . . . 1513
14.293.3.1 flags . . . 1513
14.294 l4re_env_t Struct Reference . . . 1514
14.294.1 Detailed Description . . . 1515
14.295 l4re_event_t Struct Reference . . . 1515
14.295.1 Detailed Description . . . 1516
14.296 l4re_video_color_component_t Struct Reference . . . 1516
14.296.1 Detailed Description . . . 1517
14.297 l4re_video_goos_info_t Struct Reference . . . 1517
14.297.1 Detailed Description . . . 1518
14.298 l4re_video_pixel_info_t Struct Reference . . . 1518
14.298.1 Detailed Description . . . 1519
14.299 l4re_video_view_info_t Struct Reference . . . 1519
14.299.1 Detailed Description . . . 1521
14.300 l4re_video_view_t Struct Reference . . . 1521
14.300.1 Detailed Description . . . 1521
14.301 l4util_idt_desc_t Struct Reference . . . 1522
14.301.1 Detailed Description . . . 1522
14.302 l4util_idt_header_t Struct Reference . . . 1523
14.302.1 Detailed Description . . . 1523
14.303 l4util_mb_addr_range_t Struct Reference . . . 1524
14.303.1 Detailed Description . . . 1524
14.304 l4util_mb_apm_t Struct Reference . . . 1525
14.304.1 Detailed Description . . . 1525
14.305 l4util_mb_drive_t Struct Reference . . . 1525
14.305.1 Detailed Description . . . 1526
14.305.2 Field Documentation . . . 1526
14.305.2.1 drive_cylinders . . . 1526

14.305.2.2 drive_mode . . . 1527
14.305.2.3 drive_number . . . 1527
14.306 L4::util_mb_info_t Struct Reference . . . 1527
14.306.1 Detailed Description . . . 1529
14.307 L4::util_mb_mod_t Struct Reference . . . 1529
14.307.1 Detailed Description . . . 1529
14.307.2 Field Documentation . . . 1530
14.307.2.1 mod_end . . . 1530
14.307.2.2 mod_start . . . 1530
14.308 L4::util_mb_vbe_ctrl_t Struct Reference . . . 1530
14.308.1 Detailed Description . . . 1531
14.309 L4::util_mb_vbe_mode_t Struct Reference . . . 1531
14.309.1 Detailed Description . . . 1533
14.310 L4::vbus::Device Class Reference . . . 1534
14.310.1 Detailed Description . . . 1536
14.310.2 Member Function Documentation . . . 1536
14.310.2.1 bus_cap() . . . 1536
14.310.2.2 dev_handle() . . . 1537
14.310.2.3 device() . . . 1538
14.310.2.4 device_by_hid() . . . 1539
14.310.2.5 get_resource() . . . 1540
14.310.2.6 is_compatible() . . . 1541
14.310.2.7 next_device() . . . 1541
14.310.2.8 operator"!=() . . . 1542
14.310.2.9 operator==() . . . 1542
14.310.3 Field Documentation . . . 1542
14.310.3.1 _bus . . . 1543
14.311 L4::vbus::Gpio_module Class Reference . . . 1543
14.311.1 Detailed Description . . . 1546
14.311.2 Member Function Documentation . . . 1546

14.311.2.1 config_pad() . . . 1546
14.311.2.2 get() . . . 1547
14.311.2.3 pin() . . . 1547
14.311.2.4 set() . . . 1548
14.311.2.5 setup() . . . 1549
14.312 L4vbus::Gpio_module::Pin_slice Struct Reference . . . 1549
14.312.1 Detailed Description . . . 1550
14.313 L4vbus::Gpio_pin Class Reference . . . 1550
14.313.1 Detailed Description . . . 1553
14.313.2 Member Function Documentation . . . 1553
14.313.2.1 config_get() . . . 1553
14.313.2.2 config_pad() . . . 1554
14.313.2.3 config_pull() . . . 1554
14.313.2.4 get() . . . 1555
14.313.2.5 pin() . . . 1556
14.313.2.6 set() . . . 1556
14.313.2.7 setup() . . . 1556
14.313.2.8 to_irq() . . . 1557
14.314 L4vbus::Icu Class Reference . . . 1558
14.314.1 Detailed Description . . . 1560
14.315 L4vbus::Pci_dev Class Reference . . . 1560
14.315.1 Detailed Description . . . 1563
14.315.2 Member Function Documentation . . . 1563
14.315.2.1 cfg_read() . . . 1563
14.315.2.2 cfg_write() . . . 1564
14.315.2.3 irq_enable() . . . 1564
14.316 L4vbus::Pci_host_bridge Class Reference . . . 1565
14.316.1 Detailed Description . . . 1568
14.316.2 Member Function Documentation . . . 1568
14.316.2.1 cfg_read() . . . 1568

14.316.2.2 cfg_write() . . . 1569
14.316.2.3 irq_enable() . . . 1570
14.317 L4vbus::Pm< DEC > Class Template Reference . . . 1571
14.317.1 Detailed Description . . . 1572
14.318 L4vbus::Vbus Class Reference . . . 1572
14.318.1 Detailed Description . . . 1575
14.318.2 Member Function Documentation . . . 1575
14.318.2.1 assign_dma_domain() [1/2] . . . 1575
14.318.2.2 assign_dma_domain() [2/2] . . . 1576
14.318.2.3 release_resource() . . . 1577
14.318.2.4 request_resource() . . . 1578
14.318.2.5 root() . . . 1578
14.319 L4vbus_device_t Struct Reference . . . 1579
14.319.1 Detailed Description . . . 1579
14.320 L4vbus_resource_t Struct Reference . . . 1580
14.320.1 Detailed Description . . . 1580
14.321 L4vcpu::State Class Reference . . . 1581
14.321.1 Detailed Description . . . 1581
14.321.2 Constructor & Destructor Documentation . . . 1581
14.321.2.1 State() . . . 1581
14.321.3 Member Function Documentation . . . 1582
14.321.3.1 add() . . . 1582
14.321.3.2 clear() . . . 1582
14.321.3.3 set() . . . 1582
14.322 L4vcpu::Vcpu Class Reference . . . 1583
14.322.1 Detailed Description . . . 1587
14.322.2 Member Function Documentation . . . 1587
14.322.2.1 cast() [1/2] . . . 1587
14.322.2.2 cast() [2/2] . . . 1587
14.322.2.3 entry_ip() . . . 1588

14.322.2.4 entry_sp() . . . 1588
14.322.2.5 ext_alloc() . . . 1589
14.322.2.6 f() [1/2] . . . 1589
14.322.2.7 f() [2/2] . . . 1589
14.322.2.8 irq_disable_save() . . . 1590
14.322.2.9 irq_enable() . . . 1590
14.322.2.10 irq_restore() . . . 1591
14.322.2.11 is_irq_entry() . . . 1591
14.322.2.12 is_page_fault_entry() . . . 1592
14.322.2.13 l() [1/2] . . . 1593
14.322.2.14 l() [2/2] . . . 1593
14.322.2.15 saved_state() [1/2] . . . 1593
14.322.2.16 saved_state() [2/2] . . . 1594
14.322.2.17 state() [1/2] . . . 1594
14.322.2.18 state() [2/2] . . . 1594
14.322.2.19 task() . . . 1594
14.322.2.20 wait_for_event() . . . 1595
14.323 L4virtio::Driver::Virtqueue Class Reference . . . 1596
14.323.1 Detailed Description . . . 1598
14.323.2 Member Function Documentation . . . 1598
14.323.2.1 alloc_descriptor() . . . 1598
14.323.2.2 desc() . . . 1598
14.323.2.3 enqueue_descriptor() . . . 1599
14.323.2.4 find_next_used() . . . 1599
14.323.2.5 free_descriptor() . . . 1599
14.323.2.6 init_queue() [1/2] . . . 1600
14.323.2.7 init_queue() [2/2] . . . 1600
14.323.2.8 initialize_rings() . . . 1601
14.324 L4virtio::Ptr< T > Class Template Reference . . . 1601
14.324.1 Detailed Description . . . 1603

14.324.2 Member Enumeration Documentation . . . 1603
14.324.2.1 Invalid_type . . . 1603
14.324.3 Member Function Documentation . . . 1603
14.324.3.1 get() . . . 1603
14.324.3.2 is_valid() . . . 1604
14.325 L4virtio::Svr::Bad_descriptor Struct Reference . . . 1605
14.325.1 Detailed Description . . . 1606
14.325.2 Member Enumeration Documentation . . . 1606
14.325.2.1 Error . . . 1606
14.325.3 Constructor & Destructor Documentation . . . 1606
14.325.3.1 Bad_descriptor() . . . 1606
14.325.4 Member Function Documentation . . . 1607
14.325.4.1 message() . . . 1607
14.326 L4virtio::Svr::Block_dev< Ds_data > Class Template Reference . . . 1607
14.326.1 Detailed Description . . . 1610
14.326.2 Constructor & Destructor Documentation . . . 1610
14.326.2.1 Block_dev() . . . 1610
14.326.3 Member Function Documentation . . . 1611
14.326.3.1 finalize_request() . . . 1611
14.326.3.2 process_request() . . . 1611
14.326.3.3 register_obj() . . . 1612
14.326.3.4 reset_client() . . . 1613
14.326.3.5 set_blk_size() . . . 1613
14.326.3.6 set_size_max() . . . 1614
14.326.3.7 set_topology() . . . 1614
14.327 L4virtio::Svr::Block_request< Ds_data > Class Template Reference . . . 1614
14.327.1 Detailed Description . . . 1615
14.327.2 Member Function Documentation . . . 1615
14.327.2.1 data_size() . . . 1615
14.327.2.2 next_block() . . . 1616

14.328 L4virtio::Svr::Data_buffer Struct Reference . . . 1617
14.328.1 Detailed Description . . . 1618
14.328.2 Constructor & Destructor Documentation . . . 1618
14.328.2.1 Data_buffer() . . . 1618
14.328.3 Member Function Documentation . . . 1618
14.328.3.1 copy_to() . . . 1618
14.328.3.2 done() . . . 1619
14.328.3.3 set() . . . 1619
14.328.3.4 skip() . . . 1620
14.329 L4virtio::Svr::Dev_config Class Reference . . . 1621
14.329.1 Detailed Description . . . 1622
14.329.2 Constructor & Destructor Documentation . . . 1622
14.329.2.1 Dev_config() . . . 1622
14.329.3 Member Function Documentation . . . 1623
14.329.3.1 change_queue_config() . . . 1623
14.329.3.2 ds() . . . 1624
14.329.3.3 get_cmd() . . . 1625
14.329.3.4 guest_features() . . . 1626
14.329.3.5 hdr() . . . 1626
14.329.3.6 qconfig() . . . 1627
14.329.3.7 reset_cmd() . . . 1628
14.329.3.8 reset_queue() . . . 1629
14.329.3.9 set_failed() . . . 1630
14.329.3.10 set_status() . . . 1630
14.329.3.11 status() . . . 1631
14.330 L4virtio::Svr::Dev_features Struct Reference . . . 1632
14.330.1 Detailed Description . . . 1633
14.330.2 Member Typedef Documentation . . . 1633
14.330.2.1 ring_event_idx_bfm_t . . . 1633
14.330.2.2 ring_indirect_desc_bfm_t . . . 1634

14.330.3 Member Function Documentation . . . 1634
14.330.3.1 ring_event_idx() [1/2] . . . 1634
14.330.3.2 ring_event_idx() [2/2] . . . 1634
14.330.3.3 ring_indirect_desc() [1/2] . . . 1634
14.330.3.4 ring_indirect_desc() [2/2] . . . 1634
14.331 L4virtio::Svr::Dev_status Struct Reference . . . 1635
14.331.1 Detailed Description . . . 1636
14.331.2 Member Typedef Documentation . . . 1636
14.331.2.1 acked_bfm_t . . . 1636
14.331.2.2 driver_bfm_t . . . 1637
14.331.2.3 driver_ok_bfm_t . . . 1637
14.331.2.4 failed_bfm_t . . . 1637
14.331.2.5 features_ok_bfm_t . . . 1637
14.331.3 Member Function Documentation . . . 1637
14.331.3.1 acked() [1/2] . . . 1638
14.331.3.2 acked() [2/2] . . . 1638
14.331.3.3 driver() [1/2] . . . 1638
14.331.3.4 driver() [2/2] . . . 1639
14.331.3.5 driver_ok() [1/2] . . . 1639
14.331.3.6 driver_ok() [2/2] . . . 1639
14.331.3.7 failed() [1/2] . . . 1640
14.331.3.8 failed() [2/2] . . . 1640
14.331.3.9 features_ok() [1/2] . . . 1640
14.331.3.10 features_ok() [2/2] . . . 1641
14.331.3.11 running() . . . 1641
14.332 L4virtio::Svr::Device_t< DATA > Class Template Reference . . . 1642
14.332.1 Detailed Description . . . 1644
14.332.2 Member Function Documentation . . . 1644
14.332.2.1 device_error() . . . 1644
14.332.2.2 handle_mem_cmd_write() . . . 1644

14.332.2.3 init_mem_info() . . . 1644
14.332.2.4 reset_queue_config() . . . 1645
14.332.2.5 setup_queue() . . . 1645
14.333 L4virtio::Svr::Driver_mem_list_t< DATA > Class Template Reference . . . 1646
14.333.1 Detailed Description . . . 1648
14.333.2 Member Function Documentation . . . 1648
14.333.2.1 add() . . . 1648
14.333.2.2 find() . . . 1649
14.333.2.3 full() . . . 1650
14.333.2.4 init() . . . 1650
14.333.2.5 load_desc() [1/3] . . . 1651
14.333.2.6 load_desc() [2/3] . . . 1651
14.333.2.7 load_desc() [3/3] . . . 1652
14.333.2.8 remove() . . . 1653
14.334 L4virtio::Svr::Driver_mem_region_t< DATA > Class Template Reference . . . 1653
14.334.1 Detailed Description . . . 1655
14.334.2 Constructor & Destructor Documentation . . . 1656
14.334.2.1 Driver_mem_region_t() . . . 1656
14.334.3 Member Function Documentation . . . 1656
14.334.3.1 contains() . . . 1656
14.334.3.2 drv_base() . . . 1657
14.334.3.3 ds() . . . 1657
14.334.3.4 ds_offset() . . . 1657
14.334.3.5 empty() . . . 1658
14.334.3.6 flags() . . . 1658
14.334.3.7 is_writable() . . . 1658
14.334.3.8 local() . . . 1658
14.334.3.9 local_base() . . . 1659
14.334.3.10 size() . . . 1660
14.335 L4virtio::Svr::Request_processor Class Reference . . . 1660

14.335.1 Detailed Description . . . 1661
14.335.2 Member Function Documentation . . . 1661
14.335.2.1 current_flags() . . . 1661
14.335.2.2 has_more() . . . 1662
14.335.2.3 next() . . . 1662
14.335.2.4 start() [1/2] . . . 1664
14.335.2.5 start() [2/2] . . . 1665
14.336 L4virtio::Svr::Virtqueue Class Reference . . . 1666
14.336.1 Detailed Description . . . 1669
14.336.2 Member Function Documentation . . . 1669
14.336.2.1 consumed() . . . 1669
14.336.2.2 desc() . . . 1670
14.336.2.3 desc_avail() . . . 1671
14.336.2.4 disable_notify() . . . 1671
14.336.2.5 enable_notify() . . . 1671
14.336.2.6 next_avail() . . . 1672
14.337 L4virtio::Svr::Virtqueue::Head_desc Class Reference . . . 1673
14.337.1 Detailed Description . . . 1673
14.337.2 Member Function Documentation . . . 1673
14.337.2.1 desc() . . . 1674
14.337.2.2 operator Null_ptr_check const *() . . . 1674
14.337.2.3 valid() . . . 1674
14.338 L4virtio::Virtqueue Class Reference . . . 1675
14.338.1 Detailed Description . . . 1678
14.338.2 Member Function Documentation . . . 1678
14.338.2.1 avail_align() . . . 1678
14.338.2.2 avail_size() . . . 1678
14.338.2.3 desc_align() . . . 1679
14.338.2.4 desc_size() . . . 1679
14.338.2.5 disable() . . . 1680

14.338.2.6 dump() . . . 1681
14.338.2.7 get_avail_idx() . . . 1681
14.338.2.8 get_tail_avail_idx() . . . 1682
14.338.2.9 no_notify_guest() . . . 1682
14.338.2.10 no_notify_host() [1/2] . . . 1683
14.338.2.11 no_notify_host() [2/2] . . . 1683
14.338.2.12 num() . . . 1684
14.338.2.13 ready() . . . 1684
14.338.2.14 setup() . . . 1685
14.338.2.15 setup_simple() . . . 1685
14.338.2.16 total_size() [1/2] . . . 1686
14.338.2.17 total_size() [2/2] . . . 1686
14.338.2.18 used_align() . . . 1687
14.338.2.19 used_size() . . . 1687
14.339 L4virtio::Virtqueue::Avail Class Reference . . . 1688
14.339.1 Detailed Description . . . 1689
14.340 L4virtio::Virtqueue::Avail::Flags Struct Reference . . . 1689
14.340.1 Detailed Description . . . 1690
14.340.2 Member Typedef Documentation . . . 1690
14.340.2.1 no_irq_bfm_t . . . 1690
14.340.3 Member Function Documentation . . . 1690
14.340.3.1 no_irq() [1/2] . . . 1690
14.340.3.2 no_irq() [2/2] . . . 1691
14.341 L4virtio::Virtqueue::Desc Class Reference . . . 1691
14.341.1 Detailed Description . . . 1693
14.342 L4virtio::Virtqueue::Desc::Flags Struct Reference . . . 1693
14.342.1 Detailed Description . . . 1694
14.342.2 Member Typedef Documentation . . . 1694
14.342.2.1 indirect_bfm_t . . . 1695
14.342.2.2 next_bfm_t . . . 1695

14.342.2.3 write_bfm_t . . . 1695
14.342.3 Member Function Documentation . . . 1695
14.342.3.1 indirect() [1/2] . . . 1695
14.342.3.2 indirect() [2/2] . . . 1696
14.342.3.3 next() [1/2] . . . 1696
14.342.3.4 next() [2/2] . . . 1696
14.342.3.5 write() [1/2] . . . 1696
14.342.3.6 write() [2/2] . . . 1697
14.343 L4virtio::Virtqueue::Used Class Reference . . . 1697
14.343.1 Detailed Description . . . 1698
14.344 L4virtio::Virtqueue::Used::Flags Struct Reference . . . 1698
14.344.1 Detailed Description . . . 1699
14.344.2 Member Typedef Documentation . . . 1699
14.344.2.1 no_notify_bfm_t . . . 1699
14.344.3 Member Function Documentation . . . 1699
14.344.3.1 no_notify() [1/2] . . . 1700
14.344.3.2 no_notify() [2/2] . . . 1700
14.345 L4virtio::Virtqueue::Used_elem Struct Reference . . . 1700
14.345.1 Detailed Description . . . 1701
14.345.2 Constructor & Destructor Documentation . . . 1701
14.345.2.1 Used_elem() . . . 1701
14.346 L4virtio_block_config_t Struct Reference . . . 1701
14.346.1 Detailed Description . . . 1702
14.346.2 Field Documentation . . . 1702
14.346.2.1 blk_size . . . 1702
14.347 L4virtio_block_header_t Struct Reference . . . 1703
14.347.1 Detailed Description . . . 1703
14.348 L4virtio_config_hdr_t Struct Reference . . . 1704
14.348.1 Detailed Description . . . 1705
14.348.2 Field Documentation . . . 1705
14.348.2.1 magic . . . 1705
14.348.2.2 status . . . 1705
14.349 L4virtio_config_queue_t Struct Reference . . . 1706
14.349.1 Detailed Description . . . 1707

15 File Documentation 1709

15.1 amd64/l4/util/apic.h File Reference . . . 1709

15.1.1 Detailed Description . . . 1709

15.2 apic.h . . . 1710

15.3 x86/l4/util/apic.h File Reference . . . 1714

15.3.1 Detailed Description . . . 1715

15.4 apic.h . . . 1715

15.5 amd64/l4/util/idt.h File Reference . . . 1720

15.5.1 Detailed Description . . . 1721

15.6 idt.h . . . 1721

15.7 x86/l4/util/idt.h File Reference . . . 1722

15.7.1 Detailed Description . . . 1722

15.8 idt.h . . . 1723

15.9 amd64/l4/util/perform.h File Reference . . . 1723

15.9.1 Detailed Description . . . 1724

15.10 perform.h . . . 1724

15.11 x86/l4/util/perform.h File Reference . . . 1729

15.11.1 Detailed Description . . . 1730

15.12 perform.h . . . 1730

15.13 amd64/l4/util/rdtsc.h File Reference . . . 1735

15.13.1 Detailed Description . . . 1737

15.14 rdtsc.h . . . 1737

15.15 x86/l4/util/rdtsc.h File Reference . . . 1740

15.15.1 Detailed Description . . . 1741

15.16 rdtsc.h . . . 1741

15.17 amd64/l4/util/spin.h File Reference . . . 1745

15.17.1 Detailed Description . . . 1745

15.18 spin.h . . . 1745

15.19 x86/l4/util/spin.h File Reference . . . 1746

15.19.1 Detailed Description . . . 1746

15.20 spin.h . . . 1746
15.21 amd64/l4/util/util.h File Reference . . . 1747
15.21.1 Detailed Description . . . 1748
15.21.2 Function Documentation . . . 1748
15.21.2.1 l4_sleep() . . . 1748
15.21.2.2 l4util_micros2l4to() . . . 1749
15.22 util.h . . . 1749
15.23 x86/l4/util/util.h File Reference . . . 1750
15.23.1 Detailed Description . . . 1751
15.23.2 Function Documentation . . . 1751
15.23.2.1 l4_sleep() . . . 1751
15.23.2.2 l4util_micros2l4to() . . . 1751
15.24 util.h . . . 1752
15.25 amd64/l4f/l4/sys/segment.h File Reference . . . 1752
15.25.1 Detailed Description . . . 1753
15.25.2 Function Documentation . . . 1753
15.25.2.1 fiasco_amd64_set_fs() . . . 1753
15.25.2.2 fiasco_amd64_set_segment_base() . . . 1754
15.26 segment.h . . . 1754
15.27 amd64/l4/sys/segment.h File Reference . . . 1755
15.27.1 Detailed Description . . . 1757
15.27.2 Enumeration Type Documentation . . . 1757
15.27.2.1 L4_task_ldt_x86_consts . . . 1757
15.27.3 Function Documentation . . . 1757
15.27.3.1 fiasco_amd64_segment_info() . . . 1757
15.27.3.2 fiasco_amd64_set_fs() . . . 1758
15.27.3.3 fiasco_amd64_set_segment_base() . . . 1758
15.28 segment.h . . . 1759
15.29 x86/l4f/l4/sys/segment.h File Reference . . . 1760
15.29.1 Detailed Description . . . 1761

15.30segment.h . . . 1761
15.31x86/l4/sys/segment.h File Reference . . . 1762
15.31.1 Detailed Description . . . 1763
15.31.2 Enumeration Type Documentation . . . 1763
15.31.2.1 L4_task_ldt_x86_consts . . . 1763
15.32segment.h . . . 1763
15.33amd64/l4f/l4/util/port_io.h File Reference . . . 1764
15.33.1 Detailed Description . . . 1765
15.34port_io.h . . . 1765
15.35amd64/l4/util/port_io.h File Reference . . . 1765
15.35.1 Detailed Description . . . 1766
15.36port_io.h . . . 1766
15.37x86/l4f/l4/util/port_io.h File Reference . . . 1766
15.37.1 Detailed Description . . . 1767
15.37.2 Function Documentation . . . 1768
15.37.2.1 l4util_ioport_map() . . . 1768
15.38port_io.h . . . 1769
15.39x86/l4/util/port_io.h File Reference . . . 1769
15.39.1 Detailed Description . . . 1771
15.40port_io.h . . . 1771
15.41amd64/l4f/l4/util/setjmp.h File Reference . . . 1773
15.41.1 Detailed Description . . . 1774
15.41.2 Function Documentation . . . 1774
15.41.2.1 l4_thread_longjmp() . . . 1774
15.41.2.2 l4_thread_setjmp() . . . 1775
15.42setjmp.h . . . 1775
15.43x86/l4f/l4/util/setjmp.h File Reference . . . 1776
15.43.1 Detailed Description . . . 1777
15.43.2 Function Documentation . . . 1777
15.43.2.1 l4_thread_longjmp() . . . 1777

15.43.2.2 l4_thread_setjmp() . . . 1778
15.44 setjmp.h . . . 1778
15.45 arm/l4/sys/linkage.h File Reference . . . 1779
15.45.1 Detailed Description . . . 1779
15.46 linkage.h . . . 1779
15.47 amd64/l4/sys/linkage.h File Reference . . . 1780
15.47.1 Detailed Description . . . 1780
15.48 linkage.h . . . 1780
15.49 x86/l4/sys/linkage.h File Reference . . . 1780
15.49.1 Detailed Description . . . 1781
15.50 linkage.h . . . 1781
15.51 arm/l4/sys/mem_op.h File Reference . . . 1781
15.51.1 Detailed Description . . . 1782
15.52 mem_op.h . . . 1783
15.53 arm/l4/sys/vm.h File Reference . . . 1784
15.53.1 Detailed Description . . . 1784
15.54 vm.h . . . 1784
15.55 arm/l4/util/bitops_arch.h File Reference . . . 1785
15.55.1 Detailed Description . . . 1785
15.56 bitops_arch.h . . . 1786
15.57 amd64/l4/util/bitops_arch.h File Reference . . . 1786
15.57.1 Detailed Description . . . 1786
15.58 bitops_arch.h . . . 1786
15.59 x86/l4/util/bitops_arch.h File Reference . . . 1789
15.59.1 Detailed Description . . . 1790
15.60 bitops_arch.h . . . 1790
15.61 arm/l4/util/cpu.h File Reference . . . 1793
15.61.1 Detailed Description . . . 1793
15.62 cpu.h . . . 1794
15.63 amd64/l4/util/cpu.h File Reference . . . 1794

15.63.1 Detailed Description . . . 1795
15.64 cpu.h . . . 1795
15.65 x86/l4/util/cpu.h File Reference . . . 1796
15.65.1 Detailed Description . . . 1797
15.66 cpu.h . . . 1797
15.67 arm/l4/util/l4_macros.h File Reference . . . 1798
15.67.1 Detailed Description . . . 1798
15.68 l4_macros.h . . . 1798
15.69 amd64/l4/util/l4_macros.h File Reference . . . 1799
15.69.1 Detailed Description . . . 1799
15.70 l4_macros.h . . . 1799
15.71 l4/util/l4_macros.h File Reference . . . 1799
15.71.1 Detailed Description . . . 1800
15.72 l4_macros.h . . . 1800
15.73 x86/l4/util/l4_macros.h File Reference . . . 1800
15.73.1 Detailed Description . . . 1800
15.74 l4_macros.h . . . 1801
15.75 arm/l4/util/mbi_argv.h File Reference . . . 1801
15.75.1 Detailed Description . . . 1801
15.76 mbi_argv.h . . . 1802
15.77 amd64/l4/util/mbi_argv.h File Reference . . . 1802
15.77.1 Detailed Description . . . 1802
15.78 mbi_argv.h . . . 1803
15.79 x86/l4/util/mbi_argv.h File Reference . . . 1803
15.79.1 Detailed Description . . . 1804
15.80 mbi_argv.h . . . 1804
15.81 arm/l4/util/stack_impl.h File Reference . . . 1805
15.81.1 Detailed Description . . . 1805
15.81.2 Function Documentation . . . 1805
15.81.2.1 l4util_stack_get_sp() . . . 1805

15.82 stack_impl.h . . . 1806
15.83 amd64/l4/util/stack_impl.h File Reference . . . 1806
15.83.1 Detailed Description . . . 1806
15.83.2 Function Documentation . . . 1806
15.83.2.1 l4util_stack_get_sp() . . . 1806
15.84 stack_impl.h . . . 1807
15.85 x86/l4/util/stack_impl.h File Reference . . . 1807
15.85.1 Detailed Description . . . 1807
15.85.2 Function Documentation . . . 1808
15.85.2.1 l4util_stack_get_sp() . . . 1808
15.86 stack_impl.h . . . 1808
15.87 arm/l4f/l4/sys/syscall_defs.h File Reference . . . 1808
15.87.1 Detailed Description . . . 1809
15.88 syscall_defs.h . . . 1809
15.89 contrib/libio-io/l4/io/io.h File Reference . . . 1809
15.89.1 Function Documentation . . . 1811
15.89.1.1 l4io_get_root_device() . . . 1811
15.89.1.2 l4io_iterate_devices() . . . 1811
15.89.1.3 l4io_request_all_ioports() . . . 1812
15.89.1.4 l4io_request_icu() . . . 1812
15.90 io.h . . . 1812
15.91 l4/sys/types.h File Reference . . . 1813
15.91.1 Detailed Description . . . 1816
15.91.2 Function Documentation . . . 1816
15.91.2.1 l4_capability_next() . . . 1816
15.92 types.h . . . 1816
15.93 l4/cxx/avl_map File Reference . . . 1819
15.93.1 Detailed Description . . . 1820
15.94 avl_map . . . 1820
15.95 l4/cxx/avl_set File Reference . . . 1822

15.95.1 Detailed Description . . . 1823
15.96 avl_set . . . 1823
15.97 l4/cxx/avl_tree File Reference . . . 1826
15.97.1 Detailed Description . . . 1828
15.98 avl_tree . . . 1828
15.99 l4/cxx/basic_ostream File Reference . . . 1832
15.99.1 Detailed Description . . . 1833
15.100 basic_ostream . . . 1833
15.101 l4/cxx/basic_vector.h File Reference . . . 1836
15.101.1 Detailed Description . . . 1836
15.102 basic_vector.h . . . 1837
15.103 l4/cxx/bits/bst.h File Reference . . . 1837
15.103.1 Detailed Description . . . 1839
15.104 bst.h . . . 1839
15.105 l4/cxx/bits/bst_base.h File Reference . . . 1841
15.105.1 Detailed Description . . . 1843
15.106 bst_base.h . . . 1843
15.107 l4/cxx/bits/bst_iter.h File Reference . . . 1844
15.107.1 Detailed Description . . . 1845
15.108 bst_iter.h . . . 1846
15.109 l4/cxx/exceptions File Reference . . . 1847
15.109.1 Detailed Description . . . 1849
15.110 exceptions . . . 1849
15.111 l4/cxx/iostream File Reference . . . 1852
15.111.1 Detailed Description . . . 1853
15.112 iostream . . . 1853
15.113 l4/cxx/ipc_helper File Reference . . . 1853
15.113.1 Detailed Description . . . 1854
15.114 ipc_helper . . . 1855
15.115 l4/cxx/ipc_stream File Reference . . . 1855

15.115.1 Detailed Description . . . 1858
15.115.2 Function Documentation . . . 1858
15.115.2.1 operator<<() [1/4] . . . 1858
15.115.2.2 operator<<() [2/4] . . . 1859
15.115.2.3 operator<<() [3/4] . . . 1860
15.115.2.4 operator<<() [4/4] . . . 1860
15.115.2.5 operator>>() [1/6] . . . 1861
15.115.2.6 operator>>() [2/6] . . . 1862
15.115.2.7 operator>>() [3/6] . . . 1863
15.115.2.8 operator>>() [4/6] . . . 1863
15.115.2.9 operator>>() [5/6] . . . 1864
15.115.2.10 operator>>() [6/6] . . . 1865
15.116 pc_stream . . . 1866
15.117 l4/cxx/l4iostream File Reference . . . 1876
15.117.1 Detailed Description . . . 1877
15.118 l4iostream . . . 1877
15.119 l4/cxx/l4types.h File Reference . . . 1878
15.119.1 Detailed Description . . . 1878
15.120 l4types.h . . . 1879
15.121 l4/cxx/main_thread File Reference . . . 1879
15.121.1 Detailed Description . . . 1880
15.122 main_thread . . . 1880
15.123 l4/cxx/pair File Reference . . . 1881
15.123.1 Detailed Description . . . 1882
15.124 pair . . . 1882
15.125 l4/cxx/std_exc_io File Reference . . . 1883
15.125.1 Detailed Description . . . 1883
15.126 std_exc_io . . . 1884
15.127 l4/cxx/string.h File Reference . . . 1884
15.127.1 Detailed Description . . . 1885

15.128 string.h . . . 1886
15.129 l4/irq/irq.h File Reference . . . 1886
15.129.1 Detailed Description . . . 1888
15.130 irq.h . . . 1888
15.131 arm/l4/util/irq.h File Reference . . . 1889
15.131.1 Detailed Description . . . 1889
15.132 irq.h . . . 1890
15.133 amd64/l4/util/irq.h File Reference . . . 1890
15.133.1 Detailed Description . . . 1891
15.133.2 Function Documentation . . . 1891
15.133.2.1 l4util_irq_acknowledge() . . . 1891
15.134 irq.h . . . 1892
15.135 x86/l4/util/irq.h File Reference . . . 1893
15.135.1 Detailed Description . . . 1893
15.135.2 Function Documentation . . . 1894
15.135.2.1 l4util_irq_acknowledge() . . . 1894
15.136 irq.h . . . 1894
15.137 l4/sys/irq.h File Reference . . . 1895
15.137.1 Detailed Description . . . 1897
15.138 irq.h . . . 1897
15.139 l4/libedid/edid.h File Reference . . . 1900
15.140 edid.h . . . 1901
15.141 l4/libgfxbitmap/bitmap.h File Reference . . . 1902
15.141.1 Detailed Description . . . 1903
15.141.2 Macro Definition Documentation . . . 1903
15.141.2.1 pSLIM_BMAP_START_LSB . . . 1904
15.142 bitmap.h . . . 1904
15.143 l4/libgfxbitmap/font.h File Reference . . . 1905
15.143.1 Detailed Description . . . 1906
15.144 font.h . . . 1906

15.145 L4/libgfxbitmap/support File Reference . . . 1907
15.145.1 Detailed Description . . . 1907
15.146 support . . . 1908
15.147 L4/re/c/dataspace.h File Reference . . . 1908
15.147.1 Detailed Description . . . 1909
15.148 dataspace.h . . . 1910
15.149 L4/re/c/debug.h File Reference . . . 1911
15.149.1 Detailed Description . . . 1911
15.150 debug.h . . . 1911
15.151 L4/re/c/dma_space.h File Reference . . . 1912
15.151.1 Detailed Description . . . 1913
15.151.2 Typedef Documentation . . . 1913
15.151.2.1 l4re_dma_space_dma_addr_t . . . 1913
15.151.3 Enumeration Type Documentation . . . 1913
15.151.3.1 l4re_dma_space_direction . . . 1913
15.151.3.2 l4re_dma_space_space_attribs . . . 1914
15.152 dma_space.h . . . 1914
15.153 L4/re/c/event.h File Reference . . . 1915
15.153.1 Detailed Description . . . 1916
15.154 event.h . . . 1917
15.155 L4/re/event.h File Reference . . . 1917
15.155.1 Detailed Description . . . 1918
15.156 event.h . . . 1918
15.157 L4/re/c/log.h File Reference . . . 1919
15.157.1 Detailed Description . . . 1920
15.158 log.h . . . 1920
15.159 L4/re/c/mem_alloc.h File Reference . . . 1921
15.159.1 Detailed Description . . . 1922
15.160 mem_alloc.h . . . 1923
15.161 L4/re/c/namespace.h File Reference . . . 1924

15.161.1 Detailed Description . . . 1925
15.162 namespace.h . . . 1925
15.163 l4/re/c/rm.h File Reference . . . 1926
15.163.1 Detailed Description . . . 1927
15.164 rm.h . . . 1927
15.165 l4/re/c/util/cap_alloc.h File Reference . . . 1930
15.165.1 Detailed Description . . . 1930
15.166 cap_alloc.h . . . 1931
15.167 l4/re/c/util/kumem_alloc.h File Reference . . . 1931
15.167.1 Detailed Description . . . 1932
15.167.2 Function Documentation . . . 1932
15.167.2.1 l4re_util_kumem_alloc() . . . 1932
15.168 kumem_alloc.h . . . 1933
15.169 l4/re/c/util/video/goos_fb.h File Reference . . . 1933
15.169.1 Detailed Description . . . 1934
15.170 goos_fb.h . . . 1934
15.171 l4/re/c/video/colors.h File Reference . . . 1935
15.171.1 Detailed Description . . . 1936
15.172 colors.h . . . 1937
15.173 l4/re/c/video/goos.h File Reference . . . 1937
15.173.1 Detailed Description . . . 1939
15.174 goos.h . . . 1939
15.175 l4/re/c/video/view.h File Reference . . . 1940
15.175.1 Detailed Description . . . 1942
15.176 view.h . . . 1943
15.177 l4/re/cap_alloc File Reference . . . 1944
15.177.1 Detailed Description . . . 1945
15.178 cap_alloc . . . 1945
15.179 l4/re/util/cap_alloc File Reference . . . 1947
15.179.1 Detailed Description . . . 1949

15.180 ap_alloc . . . 1949
15.181 l4/re/consts File Reference . . . 1951
15.181.1 Detailed Description . . . 1952
15.182 consts . . . 1952
15.183 l4/re/dataspace File Reference . . . 1953
15.183.1 Detailed Description . . . 1954
15.184 dataspace . . . 1954
15.185 l4/re/dataspace-sys.h File Reference . . . 1955
15.185.1 Detailed Description . . . 1956
15.186 dataspace-sys.h . . . 1956
15.187 l4/re/debug File Reference . . . 1956
15.187.1 Detailed Description . . . 1957
15.188 debug . . . 1958
15.189 l4/re/dma_space File Reference . . . 1958
15.190 dma_space . . . 1960
15.191 l4/re/elf_aux.h File Reference . . . 1961
15.191.1 Detailed Description . . . 1962
15.192 elf_aux.h . . . 1962
15.193 l4/re/env File Reference . . . 1963
15.193.1 Detailed Description . . . 1964
15.194 env . . . 1964
15.195 l4/re/env.h File Reference . . . 1965
15.195.1 Detailed Description . . . 1967
15.195.2 Typedef Documentation . . . 1967
15.195.2.1 l4re_env_t . . . 1967
15.196 env.h . . . 1967
15.197 l4/re/error_helper File Reference . . . 1969
15.197.1 Detailed Description . . . 1971
15.198 error_helper . . . 1971
15.199 l4/re/util/event File Reference . . . 1972

15.200 event . . . 1973
15.201 l4/re/impl/dataspace_impl.h File Reference . . . 1975
15.201.1 Detailed Description . . . 1976
15.202 dataspace_impl.h . . . 1976
15.203 l4/re/impl/mem_alloc_impl.h File Reference . . . 1977
15.203.1 Detailed Description . . . 1978
15.204 mem_alloc_impl.h . . . 1978
15.205 l4/re/impl/namespace_impl.h File Reference . . . 1979
15.205.1 Detailed Description . . . 1980
15.206 namespace_impl.h . . . 1980
15.207 l4/re/impl/rm_impl.h File Reference . . . 1981
15.207.1 Detailed Description . . . 1982
15.208 rm_impl.h . . . 1982
15.209 l4/re/l4aux.h File Reference . . . 1984
15.209.1 Detailed Description . . . 1984
15.210 l4aux.h . . . 1985
15.211 l4/re/log File Reference . . . 1985
15.211.1 Detailed Description . . . 1987
15.212 log . . . 1987
15.213 l4/re/log-sys.h File Reference . . . 1987
15.213.1 Detailed Description . . . 1988
15.214 log-sys.h . . . 1988
15.215 l4/re/mem_alloc File Reference . . . 1988
15.215.1 Detailed Description . . . 1990
15.216 mem_alloc . . . 1990
15.217 l4/re/mem_alloc-sys.h File Reference . . . 1990
15.217.1 Detailed Description . . . 1991
15.218 mem_alloc-sys.h . . . 1991
15.219 l4/re/namespace File Reference . . . 1992
15.219.1 Detailed Description . . . 1993

15.220 namespace . . . 1993
15.221 L4/re/namespace-sys.h File Reference . . . 1994
15.221.1 Detailed Description . . . 1995
15.222 namespace-sys.h . . . 1995
15.223 L4/re/parent File Reference . . . 1995
15.223.1 Detailed Description . . . 1997
15.224 parent . . . 1997
15.225 L4/re/parent-sys.h File Reference . . . 1997
15.225.1 Detailed Description . . . 1998
15.226 parent-sys.h . . . 1998
15.227 L4/re/protocols.h File Reference . . . 1998
15.227.1 Detailed Description . . . 1999
15.227.2 Enumeration Type Documentation . . . 1999
15.227.2.1 L4re_protocols . . . 1999
15.228 protocols.h . . . 1999
15.229 L4/re/rm File Reference . . . 2000
15.229.1 Detailed Description . . . 2001
15.230 rm . . . 2001
15.231 L4/re/rm-sys.h File Reference . . . 2005
15.231.1 Detailed Description . . . 2006
15.232 rm-sys.h . . . 2006
15.233 L4/re/shared_cap File Reference . . . 2006
15.233.1 Detailed Description . . . 2008
15.234 shared_cap . . . 2008
15.235 L4/re/util/shared_cap File Reference . . . 2009
15.235.1 Detailed Description . . . 2010
15.236 shared_cap . . . 2010
15.237 L4/re/unique_cap File Reference . . . 2011
15.237.1 Detailed Description . . . 2013
15.238 unique_cap . . . 2013

15.239 l4/re/util/unique_cap File Reference . . . 2013
15.239.1 Detailed Description . . . 2015
15.240 unique_cap . . . 2015
15.241 l4/re/util/bitmap_cap_alloc File Reference . . . 2016
15.241.1 Detailed Description . . . 2017
15.242 bitmap_cap_alloc . . . 2017
15.243 l4/re/util/cap File Reference . . . 2018
15.243.1 Detailed Description . . . 2019
15.244 cap . . . 2020
15.245 l4/re/util/cap_alloc_impl.h File Reference . . . 2020
15.245.1 Detailed Description . . . 2021
15.246 cap_alloc_impl.h . . . 2021
15.247 l4/re/util/counting_cap_alloc File Reference . . . 2022
15.247.1 Detailed Description . . . 2023
15.248 counting_cap_alloc . . . 2023
15.249 l4/re/util/item_alloc File Reference . . . 2025
15.249.1 Detailed Description . . . 2026
15.250 item_alloc . . . 2026
15.251 l4/re/util/kumem_alloc File Reference . . . 2027
15.251.1 Detailed Description . . . 2028
15.252 kumem_alloc . . . 2028
15.253 l4/re/util/region_mapping File Reference . . . 2029
15.253.1 Detailed Description . . . 2030
15.254 region_mapping . . . 2030
15.255 l4/re/video/goos-sys.h File Reference . . . 2035
15.255.1 Detailed Description . . . 2036
15.256 goos-sys.h . . . 2036
15.257 l4/shmc/shmc.h File Reference . . . 2036
15.257.1 Detailed Description . . . 2038
15.258 shmc.h . . . 2039

15.259 l4/sigma0/sigma0.h File Reference . . . 2041
15.259.1 Detailed Description . . . 2043
15.260 sigma0.h . . . 2043
15.261 l4/sys/__kernel_object_impl.h File Reference . . . 2044
15.261.1 Detailed Description . . . 2045
15.262 __kernel_object_impl.h . . . 2045
15.263 l4/sys/__ktrace-impl.h File Reference . . . 2046
15.263.1 Detailed Description . . . 2047
15.264 __ktrace-impl.h . . . 2047
15.265 l4/sys/__typeinfo.h File Reference . . . 2048
15.265.1 Detailed Description . . . 2051
15.266 __typeinfo.h . . . 2051
15.267 l4/sys/cache.h File Reference . . . 2061
15.267.1 Detailed Description . . . 2062
15.268 cache.h . . . 2062
15.269 arm/l4/sys/cache.h File Reference . . . 2063
15.269.1 Detailed Description . . . 2064
15.270 cache.h . . . 2064
15.271 amd64/l4/sys/cache.h File Reference . . . 2066
15.271.1 Detailed Description . . . 2066
15.272 cache.h . . . 2066
15.273 x86/l4/sys/cache.h File Reference . . . 2067
15.273.1 Detailed Description . . . 2067
15.274 cache.h . . . 2068
15.275 l4/sys/capability File Reference . . . 2068
15.275.1 Detailed Description . . . 2070
15.275.2 Macro Definition Documentation . . . 2070
15.275.2.1 L4_DISABLE_COPY . . . 2070
15.276 capability . . . 2070
15.277 l4/sys/compiler.h File Reference . . . 2072

15.277.1 Detailed Description . . . 2073
15.278 compiler.h . . . 2073
15.279 l4/sys/consts.h File Reference . . . 2075
15.279.1 Detailed Description . . . 2077
15.280 consts.h . . . 2077
15.281 arm/l4/sys/consts.h File Reference . . . 2079
15.281.1 Detailed Description . . . 2079
15.282 consts.h . . . 2080
15.283 amd64/l4/sys/consts.h File Reference . . . 2080
15.283.1 Detailed Description . . . 2080
15.284 consts.h . . . 2081
15.285 x86/l4/sys/consts.h File Reference . . . 2081
15.285.1 Detailed Description . . . 2081
15.286 consts.h . . . 2082
15.287 l4/re/consts.h File Reference . . . 2082
15.287.1 Detailed Description . . . 2083
15.288 consts.h . . . 2083
15.289 l4/sys/cxx/ipc_client File Reference . . . 2083
15.289.1 Macro Definition Documentation . . . 2085
15.289.1.1 L4_RPC_DEF . . . 2085
15.290 ipc_client . . . 2085
15.291 l4/sys/cxx/ipc_iface File Reference . . . 2086
15.291.1 Detailed Description . . . 2088
15.291.2 Macro Definition Documentation . . . 2088
15.291.2.1 L4_INLINE_RPC . . . 2088
15.291.2.2 L4_INLINE_RPC_NF . . . 2089
15.291.2.3 L4_INLINE_RPC_NF_OP . . . 2089
15.291.2.4 L4_INLINE_RPC_OP . . . 2090
15.291.2.5 L4_RPC . . . 2091
15.291.2.6 L4_RPC_NF . . . 2091

15.291.2.7 L4_RPC_NF_OP . . . 2092
15.291.2.8 L4_RPC_OP . . . 2092
15.292 ipc_iface . . . 2093
15.293 l4/cxx/ipc_server File Reference . . . 2098
15.293.1 Detailed Description . . . 2099
15.294 ipc_server . . . 2099
15.295 l4/sys/cxx/ipc_types File Reference . . . 2100
15.296 ipc_types . . . 2102
15.297 l4/sys/cxx/smart_capability_1x File Reference . . . 2109
15.298 smart_capability_1x . . . 2110
15.299 l4/sys/cxx/types File Reference . . . 2113
15.300 types . . . 2114
15.301 l4/sys/debugger File Reference . . . 2115
15.301.1 Detailed Description . . . 2116
15.302 debugger . . . 2116
15.303 l4/sys/debugger.h File Reference . . . 2117
15.303.1 Detailed Description . . . 2119
15.303.2 Function Documentation . . . 2119
15.303.2.1 l4_debugger_get_object_name() . . . 2119
15.303.2.2 l4_debugger_query_log_name() . . . 2119
15.303.2.3 l4_debugger_query_log_typeid() . . . 2120
15.303.2.4 l4_debugger_switch_log() . . . 2121
15.304 debugger.h . . . 2121
15.305 l4/sys/err.h File Reference . . . 2124
15.305.1 Detailed Description . . . 2125
15.306 err.h . . . 2126
15.307 l4/sys/exception File Reference . . . 2126
15.307.1 Detailed Description . . . 2127
15.308 exception . . . 2128
15.309 l4/sys/factory File Reference . . . 2128

15.309.1 Detailed Description . . . 2130
15.310 factory . . . 2130
15.311 l4/sys/factory.h File Reference . . . 2132
15.311.1 Detailed Description . . . 2133
15.312 factory.h . . . 2133
15.313 l4/sys/icu File Reference . . . 2138
15.313.1 Detailed Description . . . 2139
15.314 icu . . . 2139
15.315 l4/sys/icu.h File Reference . . . 2139
15.315.1 Detailed Description . . . 2142
15.316 icu.h . . . 2142
15.317 l4/sys/ipc.h File Reference . . . 2145
15.317.1 Detailed Description . . . 2147
15.317.2 Function Documentation . . . 2147
15.317.2.1 l4_ipc_to_errno() . . . 2147
15.318 ipc.h . . . 2148
15.319 arm/l4f/l4/sys/ipc.h File Reference . . . 2151
15.319.1 Detailed Description . . . 2152
15.320 ipc.h . . . 2152
15.321 x86/l4f/l4/sys/ipc.h File Reference . . . 2153
15.321.1 Detailed Description . . . 2154
15.322 ipc.h . . . 2154
15.323 l4/sys/ipc_gate File Reference . . . 2154
15.323.1 Detailed Description . . . 2156
15.324 ipc_gate . . . 2156
15.325 l4/sys/ipc_gate.h File Reference . . . 2156
15.325.1 Detailed Description . . . 2158
15.326 ipc_gate.h . . . 2158
15.327 l4/sys/irq File Reference . . . 2159
15.327.1 Detailed Description . . . 2161

15.328 irq . . . 2161
15.329 l4/sys/kernel_object.h File Reference . . . 2163
15.329.1 Detailed Description . . . 2164
15.330 kernel_object.h . . . 2164
15.331 l4/sys/kip File Reference . . . 2165
15.331.1 Detailed Description . . . 2166
15.332 kip . . . 2167
15.333 l4/sys/ktrace.h File Reference . . . 2168
15.333.1 Detailed Description . . . 2170
15.334 ktrace.h . . . 2170
15.335 l4/sys/l4int.h File Reference . . . 2171
15.335.1 Detailed Description . . . 2172
15.336 l4int.h . . . 2172
15.337 arm/l4/sys/l4int.h File Reference . . . 2173
15.337.1 Detailed Description . . . 2173
15.338 l4int.h . . . 2173
15.339 amd64/l4/sys/l4int.h File Reference . . . 2173
15.339.1 Detailed Description . . . 2174
15.340 l4int.h . . . 2174
15.341 x86/l4/sys/l4int.h File Reference . . . 2174
15.341.1 Detailed Description . . . 2175
15.342 l4int.h . . . 2175
15.343 l4/sys/memdesc.h File Reference . . . 2175
15.343.1 Detailed Description . . . 2177
15.344 memdesc.h . . . 2177
15.345 l4/sys/meta File Reference . . . 2179
15.345.1 Detailed Description . . . 2181
15.346 meta . . . 2181
15.347 l4/sys/pager File Reference . . . 2181
15.347.1 Detailed Description . . . 2183

15.348 pager . . . 2183
15.349 l4/sys/platform_control File Reference . . . 2183
15.349.1 Detailed Description . . . 2184
15.350 platform_control . . . 2185
15.351 l4/sys/platform_control.h File Reference . . . 2185
15.351.1 Detailed Description . . . 2187
15.352 platform_control.h . . . 2187
15.353 l4/sys/rcv_endpoint File Reference . . . 2189
15.353.1 Detailed Description . . . 2190
15.354 rcv_endpoint . . . 2190
15.355 l4/sys/rcv_endpoint.h File Reference . . . 2191
15.355.1 Detailed Description . . . 2192
15.355.2 Enumeration Type Documentation . . . 2192
15.355.2.1 L4_rcv_ep_ops . . . 2192
15.356 rcv_endpoint.h . . . 2193
15.357 l4/sys/scheduler File Reference . . . 2193
15.357.1 Detailed Description . . . 2194
15.358 scheduler . . . 2195
15.359 l4/sys/scheduler.h File Reference . . . 2195
15.359.1 Detailed Description . . . 2197
15.360 scheduler.h . . . 2197
15.361 l4/sys/semaphore File Reference . . . 2200
15.361.1 Detailed Description . . . 2202
15.362 semaphore . . . 2202
15.363 l4/sys/smart_capability File Reference . . . 2202
15.363.1 Detailed Description . . . 2204
15.364 smart_capability . . . 2204
15.365 l4/sys/task File Reference . . . 2206
15.365.1 Detailed Description . . . 2208
15.366 task . . . 2208

15.367 l4/sys/task.h File Reference . . . 2209
15.367.1 Detailed Description . . . 2210
15.368 task.h . . . 2210
15.369 l4/sys/thread File Reference . . . 2214
15.369.1 Detailed Description . . . 2215
15.370 thread . . . 2215
15.371 l4/cxx/thread File Reference . . . 2217
15.371.1 Detailed Description . . . 2218
15.372 thread . . . 2218
15.373 l4/sys/typeinfo_svr File Reference . . . 2219
15.373.1 Detailed Description . . . 2221
15.374 typeinfo_svr . . . 2221
15.375 l4/sys/utcb.h File Reference . . . 2221
15.375.1 Detailed Description . . . 2223
15.376 utcb.h . . . 2223
15.377 arm/l4/sys/utcb.h File Reference . . . 2226
15.377.1 Detailed Description . . . 2227
15.378 utcb.h . . . 2227
15.379 amd64/l4/sys/utcb.h File Reference . . . 2228
15.379.1 Detailed Description . . . 2230
15.380 utcb.h . . . 2230
15.381 x86/l4/sys/utcb.h File Reference . . . 2231
15.381.1 Detailed Description . . . 2233
15.382 utcb.h . . . 2233
15.383 l4/sys/vcon File Reference . . . 2234
15.383.1 Detailed Description . . . 2236
15.384 vcon . . . 2236
15.385 l4/sys/vcon.h File Reference . . . 2236
15.385.1 Detailed Description . . . 2238
15.385.2 Enumeration Type Documentation . . . 2239

15.385.2.1 L4_vcon_read_flags . . . 2239
15.386 vcon.h . . . 2239
15.387 l4/sys/vhw.h File Reference . . . 2242
15.387.1 Detailed Description . . . 2243
15.388 vhw.h . . . 2244
15.389 l4/sys/vm File Reference . . . 2245
15.389.1 Detailed Description . . . 2246
15.390 vm . . . 2246
15.391 l4/util/alloc.h File Reference . . . 2246
15.391.1 Detailed Description . . . 2247
15.392 alloc.h . . . 2247
15.393 l4/cxx/alloc.h File Reference . . . 2248
15.393.1 Detailed Description . . . 2248
15.394 alloc.h . . . 2248
15.395 l4/util/assert.h File Reference . . . 2249
15.395.1 Detailed Description . . . 2250
15.396 assert.h . . . 2250
15.397 l4/sys/assert.h File Reference . . . 2251
15.397.1 Detailed Description . . . 2252
15.397.2 Macro Definition Documentation . . . 2253
15.397.2.1 l4_assert . . . 2253
15.398 assert.h . . . 2253
15.399 l4/util/atomic.h File Reference . . . 2254
15.399.1 Detailed Description . . . 2256
15.400 atomic.h . . . 2257
15.401 arm/l4/sys/atomic.h File Reference . . . 2261
15.401.1 Detailed Description . . . 2261
15.402 atomic.h . . . 2262
15.403 l4/cxx/atomic.h File Reference . . . 2262
15.403.1 Detailed Description . . . 2263

15.404 atomic.h . . . 2263
15.405 l4/util/backtrace.h File Reference . . . 2264
15.405.1 Detailed Description . . . 2264
15.405.2 Function Documentation . . . 2265
15.405.2.1 l4util_backtrace() . . . 2265
15.406 backtrace.h . . . 2265
15.407 l4/util/base64.h File Reference . . . 2265
15.407.1 Detailed Description . . . 2266
15.408 base64.h . . . 2267
15.409 l4/util/bitops.h File Reference . . . 2267
15.409.1 Detailed Description . . . 2269
15.410 bitops.h . . . 2269
15.411 l4/util/elf.h File Reference . . . 2272
15.411.1 Detailed Description . . . 2285
15.412 elf.h . . . 2285
15.413 l4/util/getopt.h File Reference . . . 2295
15.413.1 Detailed Description . . . 2295
15.414 getopt.h . . . 2295
15.415 l4/util/keymap.h File Reference . . . 2296
15.415.1 Detailed Description . . . 2297
15.416 keymap.h . . . 2297
15.417 l4/util/kip.h File Reference . . . 2297
15.418 kip.h . . . 2298
15.419 l4/sys/kip.h File Reference . . . 2299
15.419.1 Detailed Description . . . 2300
15.420 kip.h . . . 2300
15.421 l4/util/kprintf.h File Reference . . . 2302
15.421.1 Detailed Description . . . 2302
15.422 kprintf.h . . . 2303
15.423 l4/util/list_alloc.h File Reference . . . 2303

15.423.1 Detailed Description . . . 2304
15.423.2 Function Documentation . . . 2304
15.423.2.1 l4la_alloc() . . . 2304
15.423.2.2 l4la_avail() . . . 2304
15.423.2.3 l4la_dump() . . . 2305
15.423.2.4 l4la_free() . . . 2305
15.423.2.5 l4la_init() . . . 2305
15.424 list_alloc.h . . . 2306
15.425 l4/util/lock.h File Reference . . . 2306
15.425.1 Detailed Description . . . 2307
15.426 lock.h . . . 2307
15.427 l4/util/mb_info.h File Reference . . . 2308
15.427.1 Detailed Description . . . 2310
15.427.2 Macro Definition Documentation . . . 2310
15.427.2.1 L4UTIL_MB_MEMORY . . . 2310
15.427.2.2 MB_ARD_MEMORY . . . 2311
15.427.2.3 MB_ART_MEMORY . . . 2311
15.428 mb_info.h . . . 2311
15.429 l4/util/parse_cmd.h File Reference . . . 2315
15.429.1 Detailed Description . . . 2315
15.430 parse_cmd.h . . . 2316
15.431 l4/util/rand.h File Reference . . . 2316
15.431.1 Detailed Description . . . 2317
15.432 rand.h . . . 2317
15.433 l4/util/reboot.h File Reference . . . 2318
15.433.1 Detailed Description . . . 2318
15.434 reboot.h . . . 2318
15.435 l4/util/sll.h File Reference . . . 2319
15.435.1 Detailed Description . . . 2319
15.436 sll.h . . . 2319

15.437 l4/util/splitlog2.h File Reference . . . 2323
15.437.1 Detailed Description . . . 2323
15.438 splitlog2.h . . . 2324
15.439 l4/util/stack.h File Reference . . . 2324
15.439.1 Detailed Description . . . 2325
15.439.2 Function Documentation . . . 2325
15.439.2.1 l4util_stack_get_sp() . . . 2326
15.440 stack.h . . . 2326
15.441 l4/util/thread.h File Reference . . . 2326
15.441.1 Detailed Description . . . 2327
15.441.2 Macro Definition Documentation . . . 2328
15.441.2.1 __L4UTIL_THREAD_FUNC . . . 2328
15.442 thread.h . . . 2328
15.443 l4/sys/thread.h File Reference . . . 2329
15.443.1 Detailed Description . . . 2331
15.444 thread.h . . . 2331
15.445 arm/l4/sys/thread.h File Reference . . . 2339
15.445.1 Detailed Description . . . 2339
15.446 thread.h . . . 2339
15.447 l4/vbus/vbus.h File Reference . . . 2340
15.447.1 Detailed Description . . . 2341
15.447.2 Enumeration Type Documentation . . . 2341
15.447.2.1 anonymous enum . . . 2341
15.448 vbus.h . . . 2342
15.449 l4/vbus/vbus_types.h File Reference . . . 2343
15.449.1 Detailed Description . . . 2344
15.449.2 Enumeration Type Documentation . . . 2344
15.449.2.1 l4vbus_resource_type_t . . . 2344
15.450 vbus_types.h . . . 2344

16 Example Documentation 2347

16.1 examples/clntsrv/client.cc . . . 2347

16.2 examples/clntsrv/clntsrv.cfg . . . 2348

16.3 examples/clntsrv/server.cc . . . 2348

16.4 examples/libs/l4re/c++/mem_alloc/ma+rm.cc . . . 2349

16.5 examples/libs/l4re/c++/shared_ds/ds_clnt.cc . . . 2350

16.6 examples/libs/l4re/c++/shared_ds/ds_srv.cc . . . 2351

16.7 examples/libs/l4re/c++/shared_ds/shared_ds.cfg . . . 2353

16.8 examples/libs/l4re/c/ma+rm.c . . . 2354

16.9 examples/libs/l4re/streammap/client.cc . . . 2355

16.10 examples/libs/l4re/streammap/server.cc . . . 2356

16.11 examples/libs/l4re/streammap/streammap.cfg . . . 2357

16.12 examples/libs/libirq/async_isr.c . . . 2357

16.13 examples/libs/libirq/loop.c . . . 2358

16.14 examples/libs/shmc/prodcons.c . . . 2359

16.15 examples/sys/aliens/main.c . . . 2360

16.16 examples/sys/ipc/ipc.cfg . . . 2364

16.17 examples/sys/ipc/ipc_example.c . . . 2364

16.18 examples/sys/isr/main.c . . . 2366

16.19 examples/sys/migrate/thread_migrate.cc . . . 2367

16.20 examples/sys/migrate/thread_migrate.cfg . . . 2368

16.21 examples/sys/singlestep/main.c . . . 2369

16.22 examples/sys/start-with-exc/main.c . . . 2371

16.23 examples/sys/utcb-ipc/main.c . . . 2373

16.24 examples/sys/ux-vhw/main.c . . . 2374

16.25 hello/server/src/main.c . . . 2375

16.26 tmpfs/lib/src/fs.cc . . . 2375

Index 2383

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

- Capabilities and Naming
- 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

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 be accessed only 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 objects available. This set of objects is predetermined by the creator of a new application process and granted directly to into 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 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 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.

4.2.1 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");
```

4.2.2 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 the client and the server. 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.

4.3 Memory management - Data Spaces and the Region Map

4.3.1 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 grant the memory in a secure way, often referred to as *memory* mapping.

The described mechanism can be used to construct a memory 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.

4.3.1.1 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 the some portion are provided and further pages are inserted by the pager as a result of page faults.

4.3.1.2 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 to attach and detach data spaces to an address space as well as to reserve areas of virtual memory. Since the region-map object possesses all knowledge about virtual memory layout of a task, it also serves as an application's default pager.

4.3.1.3 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.

4.4 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

4.5 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 allocation 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

4.6 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.

4.6.1 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.

4.6.2 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.

4.7 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, &pattr, 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...
```

4.8 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 on of the RPC macros (see below) and then all the functions need to be listed in the Rpcs type.

4.8.1 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.

```
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> &

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 U↩ TCB (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>> &

4.8.2 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.

4.8.3 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...)`.

4.9 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.

4.9.1 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.

4.9.2 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

4.9.3 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.

4.9.4 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.

4.9.5 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_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_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.

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 2, 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.

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>, L4_LX_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/
```

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).

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.

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.

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.

- 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.

- 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 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.

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

Moe consists of the following subsystems:

- Name-Space Provider — provides instances of name spaces
- Boot FS — provides access to the files loaded during platform boot (e.g., linked into the boot image or loaded via GRUB boot loader)
- Log Subsystem — provides tagged log output for applications
- Scheduler Subsystem — provides simple scheduler objects for scheduling policy enforcement
- Memory Allocator, Generic Factory — provides allocation of physical RAM as data spaces, as well as allocation of the other L4Re objects provided by Moe

5.1.1 Memory Allocator, Generic Factory

The generic factory in Moe is responsible for all kinds of dynamic object allocation. The interface is a combination of L4::Factory and, for traditional reasons, L4Re::Mem_alloc.

The generic factory interface 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

The memory allocator in Moe is the alternative interface for allocating memory (RAM) in terms of L4Re::Dataspace-s (see also L4Re::Mem_alloc).

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)

5.1.2 Name-Space Provider

Moe provides a name spaces 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.

5.1.2.1 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.

5.1.3 Log Subsystem

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

5.1.4 Scheduler Subsystem

The scheduler subsystem provides a simple scheduler proxy.

5.1.5 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.

5.2 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.

5.2.1 Lua Bindings for L4Re

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

5.2.1.1 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.

5.2.1.2 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.

5.2.1.3 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,
  Irq_muxer  = -19,
  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
}
```

5.2.1.4 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.

5.2.2 Command Line Options

Ned's command line syntax is:

```
[--interactive|-i] [--cmdcap|-c CAP] [--noexit] [--execute|-e STATEMENT] <lua script> [options passed to lua s
```

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.

- Interactive Option: **interactive**, **i**

  Start ned in interactive mode. After running the initial lua script, ned will present a prompt, where Lua statements can be typed in interactively.

  Must not be used together with server mode (-c).

- Command Capability Option: **cmdcap**, **c**

  Start ned in server mode. After running the initial Lua script, ned will accept Lua statements via IPC on the given capability (see L4Re::Ned::Cmd_control).

  Must not be used together with interactive mode (-i).

- No exit Option: **noexit**

  Ned does not terminate if only Return is entered at ned's prompt.

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

  Execute the Lua statement `STATEMENT`.

5.3 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 a 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

Common Name	Description	PCI ID string
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
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`.

5.4 Uvmm, the virtual machine monitor

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 availablity 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.

5.5 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. GEOM↩ETRY 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");
```

5.6 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.

At the moment, cons only implements the L4::Vcon protocol for both, clients and frontends.

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 <promt 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.

Chapter 6

Deprecated List

Global L4::Factory::create_irq (Cap< Irq >const &target_cap, l4_utcb_t *utcb=l4_utcb())

Use `create()` with `Cap<Irq>` as argument instead.

Global L4::Factory::create_thread (Cap< Thread > const &target_cap, l4_utcb_t *utcb=l4_utcb())

Use `create()` with `Cap<Thread>` as argument instead.

Global L4::Factory::create_vm (Cap< Vm >const &target_cap, l4_utcb_t *utcb=l4_utcb())

Use `create()` with `Cap<Vm>` as argument instead.

Class L4::lpc_svr::Timed_work< HOOKS >

Use L4::lpc_svr::Timeout_queue_hooks

Global L4::Irq::attach (l4_umword_t label, Cap< Thread > const &thread=Cap< Thread >::Invalid, l4_↩utcb_t *utcb=l4_utcb())

Use bind_thread().

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

Use L4::Irq_eoi::unmask()

Global L4::Task::cap_has_child (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb())

Do not use. Undetermined future, might be removed.

Global l4_task_cap_has_child (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW

Do not use. Undetermined future, might be removed.

Global L4Re::Dataspace::phys (l4_addr_t offset, l4_addr_t &phys_addr, l4_size_t &phys_size)

Use L4Re::Dma_space instead. This function will be removed in future releases.

Global L4Re::Mem_alloc::free (L4::Cap< Dataspace > mem) const

This function is an empty stub which remains here for backward compatibility only. Use `L4::Task↩::unmap(mem, L4_FP_DELETE_OBJ)` or other similar means to remove a dataspace.

Class L4Re::Util::Auto_cap< T >

Use L4Re::Util::Unique_cap.

Class L4Re::Util::Auto_del_cap< T >

Use L4Re::Util::Unique_cap.

Global L4Re::Util::make_auto_cap ()

Use L4Re::Util::make_unique_cap().

Global L4Re::Util::make_auto_del_cap ()

Use L4Re::Util::make_unique_del_cap().

Global l4re_ma_free (l4re_ds_t const mem) L4_NOTHROW

This function is deprecated. Use l4_task_unmap() or similar means to remove a dataspace.

Global l4re_ma_free_srv (l4_cap_idx_t srv, l4re_ds_t const mem) L4_NOTHROW

This function is deprecated. Use l4_task_unmap() or similar means to remove a dataspace.

Chapter 7

Module Index

7.1 Modules

Here is a list of all modules:

- Base API . . . 101
  - Basic Macros . . . 104
  - C++ IPC Interface Definition. . . . 119
    - Internal Helpers . . . 314
  - Cache Consistency . . . 123
  - Capabilities . . . 127
  - Error codes . . . 199
  - Fiasco extensions . . . 224
    - Fiasco real time scheduling extensions . . . 230
    - Kernel Debugger . . . 332
  - Flex pages . . . 233
  - Integer Types . . . 298
  - Kernel Interface Page . . . 335
    - Fiasco-UX Virtual devices . . . 231
    - Memory descriptors (C version) . . . 420
  - Kernel Objects . . . 343
    - Factory . . . 211
    - IPC-Gate API . . . 274
    - IRQs . . . 278
    - Interrupt controller . . . 318
    - L4 kernel object type information . . . 374
    - Platform Control C API . . . 475
    - Scheduler . . . 498
    - Task . . . 526
    - Thread . . . 534
      - Thread control . . . 552
      - vCPU API . . . 630
    - Virtual Console . . . 611
    - Virtual Machines . . . 624
      - VM API for SVM . . . 582
      - VM API for TZ . . . 583
      - VM API for VMX . . . 584
  - Memory operations. . . . 425
  - Memory related . . . 428

Object Invocation . . . 461
Error Handling . . . 192
Message Items . . . 435
Message Tag . . . 439
Realtime API . . . 484
Timeouts . . . 557
Virtual Registers (UTCBs) . . . 625
ARM Virtual Registers (UTCB) . . . 89
Buffer Registers (BRs) . . . 116
Message Registers (MRs) . . . 438
Exception registers . . . 206
Thread Control Registers (TCRs) . . . 551
amd64 Virtual Registers (UTCB) . . . 629
x86 Virtual Registers (UTCB) . . . 641
DMA API for L4Re . . . 148
EDID parsing functionality . . . 160
IO interface . . . 266
IRQ handling library . . . 277
Interface for asynchronous ISR handlers. . . . 303
Interface for asynchronous ISR handlers with a given IRQ capability. . . . 301
Interface using direct functionality. . . . 308
Interface using direct functionality. . . . 305
L4 IPC Opcodes . . . 348
L4 VIRTIO Interface . . . 354
L4 VIRTIO Block Device . . . 352
L4 VIRTIO Transport Layer . . . 355
L4 Vbus functions . . . 364
L4vbus GPIO functions . . . 390
L4vbus PCI functions . . . 401
L4Re C Interface . . . 376
Capability allocator . . . 131
DMA Space Interface . . . 149
Dataspace interface . . . 153
Debug interface . . . 158
Event interface . . . 202
Initial Environment . . . 293
Kumem allocator utility . . . 345
L4Re Util C Interface . . . 388
Log interface . . . 407
Memory allocator . . . 414
Namespace interface . . . 456
Region map interface . . . 486
Video API . . . 601
L4Re C++ Interface . . . 379
Auxiliary data . . . 100
C++ Exceptions . . . 118
Console API . . . 139
Debugging API . . . 159
Event API . . . 201
L4Re ELF Auxiliary Information . . . 384
L4Re Protocol identifiers . . . 387
L4Re Util C++ Interface . . . 389
Kumem utiltities . . . 346
L4Re Capability API . . . 381
Logging interface . . . 411
Name-space API . . . 455

Parent API . . . 474
Region map API . . . 485
Vbus API . . . 600
Server-Side IPC framework . . . 504
Shared Memory Library . . . 506
Chunks . . . 132
Consumer . . . 140
Producer . . . 478
Signals . . . 518
Consumer . . . 144
Producer . . . 481
Sigma0 API . . . 512
Internal constants . . . 315
Small C++ Template Library . . . 522
Utility Functions . . . 575
Atomic Instructions . . . 90
Bit Manipulation . . . 107
Bitmap graphics and fonts . . . 115
Functions for rendering bitmap data in frame buffers . . . 249
Functions for rendering bitmap fonts to frame buffers . . . 253
CPU related functions . . . 120
Comfortable Command Line Parsing . . . 136
ELF binary format . . . 165
Functions to manipulate the local IDT . . . 258
IA32 Port I/O API . . . 259
Internal functions . . . 317
Kernel Interface Page API . . . 340
Low-Level Thread Functions . . . 412
Machine Restarting Function . . . 413
Random number support . . . 482
Timestamp Counter . . . 566
vCPU Support Library . . . 633
Extended vCPU support . . . 209

Chapter 8

Namespace Index

8.1 Namespace List

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

cxx
Our C++ library 643
cxx::Bits
Internal helpers for the cxx package 645
L4
L4 low-level kernel interface 646
L4::Ipc
IPC related functionality 657
L4::Ipc::Msg
IPC Message related functionality 666
L4::Ipc_svr
Helper classes for L4::Server instantiation 672
L4::Typeid
Definition of interface data-type helpers 673
L4::Types
L4 basic type helpers for C++ 673
L4Re
L4Re C++ Interfaces 674
L4Re::Util
Documentation of the L4 Runtime Environment utility functionality in C++ 688
L4Re::Vfs
Virtual file system for interfaces POSIX libc 698
L4virtio
L4-VIRTIO Transport C++ API 699

Chapter 9

Hierarchical Index

9.1 Class Hierarchy

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

L4::Ipc::Array_ref< CHAR, LEN > 959
L4::Ipc::Array< CHAR, LEN > 955
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, Bits::Avl_set_get_key< ITEM_TYPE > > . 762
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > 710
cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > > 762
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > 706
cxx::Bits::Base_avl_set< Pair< Region, Hdlr >, cxx::Lt_functor< Region >, Alloc, Bits::Avl_map_get_key< Region > > 762
cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc > 706
cxx::Base_slab< sizeof(Type), Slab_size, Max_free, Alloc > 721
cxx::Slab< Type, Slab_size, Max_free, Alloc > 845
cxx::Base_slab_static< sizeof(Type), Slab_size, Max_free, Alloc > 724
cxx::Slab_static< Type, Slab_size, Max_free, Alloc > 849
cxx::Bits::Basic_list< Bits::Basic_list_policy< Observer, H_list_item > > 775
cxx::H_list< Observer > 798
cxx::Bits::Basic_list< Bits::Basic_list_policy< Slab_i, H_list_item > > 775
cxx::H_list< Slab_i > 798
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item > > 775
cxx::H_list< T > 798
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item_t< T > > > 775
cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > > 798
cxx::H_list_t< T > 807
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > > 775
cxx::S_list< T > 842
cxx::Bits::Basic_list< Bits::Basic_list_policy< Timeout, H_list_item > > 775
cxx::H_list< Timeout > 798
cxx::Bits::Basic_list< Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > > . 775
cxx::H_list< Weak_ref_base, Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > > 798
cxx::H_list_t< Weak_ref_base > 807

L4::Types::Bool< __Iface_conflict< I, I2 >::value||_Iface_conflict< I, LIST::type >::value > 1222
L4::Types::Bool< false > 1222
L4::Types::False 1223
L4::Types::Same< A, B > 1230
L4::Types::Bool< I1::Proto !=PROTO_EMPTY &&I1::Proto==I2::Proto > 1222
L4::Types::Bool< Iface_conflict< I, L2::type >::value||_Conflict< L1::type, L2::type >::value > 1222
L4::Types::Bool< true > 1222
L4::Types::True 1232
L4::Ipc::Msg::Is_valid_rpc_type< A *> 1007
L4::Ipc::Msg::Is_valid_rpc_type< T > 1007
L4::Types::Bool< Typeid::Conflict< L1::type, L2::type >::value||Conflict< L1, LIST... >::value||Conflict< L2, LIST... >::value > 1222
L4::Types::Bool< Typeid::Iface_conflict< I::type, L::type >::value||Iface_conflict< I, LIST... >::value > . 1222
cxx::Bits::Bst< _Node, Bits::Avl_map_get_key< KEY_TYPE >, COMPARE< KEY_TYPE > > 777
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< KEY_TYPE >, COMPARE< KEY_TYPE > > 713
cxx::Bits::Bst< _Node, Bits::Avl_map_get_key< Region >, cxx::Lt_functor< Region > > 777
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< Region >, cxx::Lt_functor< Region > > 713
cxx::Bits::Bst< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, COMPARE > 777
cxx::Avl_tree< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, COMPARE > 713
cxx::Bits::Bst< _Node, GET_KEY, COMPARE > 777
cxx::Avl_tree< _Node, GET_KEY, COMPARE > 713
cxx::Bits::Bst< Entry, Names_get_key, Lt_functor< typename Names_get_key ::Key_type > > 777
cxx::Avl_tree< Entry, Names_get_key > 713
L4::Ipc::Msg::Clnt_val_ops< Detail::_Plain< T >::type, DIR, CLASS > 991
L4::Ipc::Msg::Clnt_val_ops< Detail::_Plain< typename _Elem< A * > ::arg_type >::type, typename Direction< A * >::type, typename Class< typename Detail::_Plain< A * >::type >::type > . . 991
L4::Ipc::Msg::Clnt_val_ops< Detail::_Plain< typename _Elem< A const * > ::arg_type >::type, typename Direction< A const * >::type, typename Class< typename Detail::_Plain< A const * >::type >::type > 991
L4::Ipc::Msg::Clnt_val_ops< Detail::_Plain< typename _Elem< Array< A, LEN > & > ::arg_type >::type, typename Direction< Array< A, LEN > & >::type, typename Class< typename Detail::_Plain< Array< A, LEN > & >::type >::type > 991
L4::Ipc::Msg::Clnt_val_ops< Detail::_Plain< typename _Elem< Array< A, LEN > > ::arg_type >::type, typename Direction< Array< A, LEN > >::type, typename Class< typename Detail::_Plain< Array< A, LEN > >::type >::type > 991
L4::Ipc::Msg::Clnt_val_ops< Detail::_Plain< typename _Elem< String< A, LEN > & > ::arg_type >:↩ :type, typename Direction< String< A, LEN > & >::type, typename Class< typename Detail↩ ::_Plain< String< A, LEN > & >::type >::type > 991
L4::Ipc::Msg::Clnt_val_ops< Detail::_Plain< typename _Elem< T > ::arg_type >::type, typename Direction< T >::type, typename Class< typename Detail::_Plain< T >::type >::type > 991
cxx::Auto_ptr< T > 701
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > 721
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > 724
cxx::Bitfield< T, LSB, MSB > 729
cxx::Bitfield< T, LSB, MSB >::Value_base< TT > 740
cxx::Bitfield< T, LSB, MSB >::Value< TT > 739
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > 742
cxx::Bitmap_base 747
cxx::Bitmap< BITS > 743
cxx::Bitmap_base::Bit 758
cxx::Bitmap_base::Char< BITS > 759
cxx::Bitmap_base::Word< BITS > 759
cxx::Bits::Avl_map_get_key< KEY_TYPE > 761
cxx::Bits::Avl_set_get_key< KEY_TYPE > 761
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > 762
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node 773

cxx::Bits::Basic_list< POLICY > . . . 775
cxx::H_list< T, POLICY > . . . 798
cxx::S_list< T, POLICY > . . . 842
cxx::Bits::Bst< Node, Get_key, Compare > . . . 777
cxx::Avl_tree< Node, Get_key, Compare > . . . 713
cxx::Bits::Bst_node . . . 791
cxx::Avl_tree_node . . . 718
cxx::Bits::Direction . . . 793
cxx::Bits::Smart_ptr_list< ITEM > . . . 795
cxx::Bits::Smart_ptr_list_item< T, STORE_T > . . . 797
cxx::Ref_obj_list_item< T > . . . 835
cxx::H_list_item_t< ELEM_TYPE > . . . 805
L4::Ipc_svr::Timeout . . . 1058
cxx::List< D, Alloc > . . . 810
cxx::List< D, Alloc >::Iter . . . 813
cxx::List_alloc . . . 814
cxx::List_item . . . 818
cxx::List_item::Iter . . . 824
cxx::List_item::T_iter< T, Poly > . . . 827
cxx::List_item::T_iter< E > . . . 827
cxx::Lt_functor< Obj > . . . 830
cxx::New_allocator< _Type > . . . 830
cxx::Nothrow . . . 831
cxx::Pair< First, Second > . . . 832
cxx::Pair_first_compare< Cmp, Typ > . . . 833
cxx::Ref_ptr< T, CNT > . . . 836
cxx::static_vector< T, IDX > . . . 852
cxx::String . . . 854
L4Re::Util::Names::Name . . . 1419
L4virtio::Svr::Device_t< Ds_data > . . . 1642
L4virtio::Svr::Block_dev< Ds_data > . . . 1607
L4virtio::Svr::Driver_mem_list_t< Ds_data > . . . 1646
L4virtio::Svr::Driver_mem_region_t< Ds_data > . . . 1653
Elf32_Dyn . . . 866
Elf32_Ehdr . . . 867
Elf32_Phdr . . . 869
Elf32_Shdr . . . 870
Elf32_Sym . . . 871
Elf64_Dyn . . . 872
Elf64_Ehdr . . . 873
Elf64_Phdr . . . 875
Elf64_Shdr . . . 876
Elf64_Sym . . . 877
gfxbitmap_offset . . . 878
cxx::H_list_item_t< Weak_ref_base > . . . 805
cxx::Weak_ref_base . . . 863
cxx::Weak_ref< T > . . . 861
L4::Kobject_2t< Console, Video::Goos, Event, L4::PROTO_EMPTY > . . . 1097
L4Re::Console . . . 1294
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 . . . 952
L4::Alloc_list . . . 879
L4::Arm_smccc . . . 880
L4::Basic_registry . . . 883

L4Re::Util::Object_registry . . . 1424
L4::Cap_base . . . 893
L4::Cap< L4::Factory > . . . 888
L4::Cap< L4::Irq > . . . 888
L4::Cap< L4::Semaphore > . . . 888
L4::Cap< L4::Thread > . . . 888
L4::Cap< L4::Vcon > . . . 888
L4::Cap< L4Re::Dataspace > . . . 888
L4::Cap< L4Re::Namespace > . . . 888
L4::Cap< L4Re::Rm > . . . 888
L4::Cap< L4Re::Video::Goos > . . . 888
L4::Cap< L4vbus::Vbus > . . . 888
L4::Cap< void > . . . 888
L4::Cap< T > . . . 888
L4::Smart_cap< T, SMART > . . . 1165
L4::Exception . . . 917
L4::Exception_tracer . . . 918
L4::Base_exception . . . 881
L4::Invalid_capability . . . 946
L4::Runtime_error . . . 1141
L4::Bounds_error . . . 885
L4::Com_error . . . 904
L4::Element_already_exists . . . 913
L4::Element_not_found . . . 915
L4::Out_of_memory . . . 1116
L4::Unknown_error . . . 1234
L4::Factory::Lstr . . . 931
L4::Factory::Nil . . . 932
L4::Factory::S . . . 933
L4::Io_pager . . . 950
L4::Kobject_t< Pager, Io_pager, L4_PROTO_PAGE_FAULT > . . . 1105
L4::Pager . . . 1119
L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > > . . . 1105
L4Re::Rm . . . 1362
L4::IOModifier . . . 954
L4::Ipc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > . . . 958
L4::Ipc::Array_ref< ELEM_TYPE, LEN_TYPE > . . . 959
L4::Ipc::Array< ELEM_TYPE, LEN_TYPE > . . . 955
L4::Ipc::As_value< T > . . . 961
L4::Ipc::Buf_item . . . 962
L4::Ipc::Call . . . 963
L4::Ipc::Call_t< RIGHTS > . . . 964
L4::Ipc::Call_zero_send_timeout . . . 966
L4::Ipc::Cap< T > . . . 967
L4::Ipc::Gen_fpage< T > . . . 970
L4::Ipc::In_out< T > . . . 974
L4::Ipc::Istream . . . 981
L4::Ipc::Iostream . . . 974
L4::Ipc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS > . . . 991
L4::Ipc::Msg::Cls_buffer . . . 992
L4::Ipc::Msg::Do_rcv_buffers . . . 1002
L4::Ipc::Msg::Cls_data . . . 994
L4::Ipc::Msg::Do_in_data . . . 998
L4::Ipc::Msg::Do_out_data . . . 1000
L4::Ipc::Msg::Cls_item . . . 995

L4::Ipc::Msg::Do_in_items . . . 999
L4::Ipc::Msg::Do_out_items . . . 1001
L4::Ipc::Msg::Dir_in . . . 996
L4::Ipc::Msg::Do_in_data . . . 998
L4::Ipc::Msg::Do_in_items . . . 999
L4::Ipc::Msg::Do_rcv_buffers . . . 1002
L4::Ipc::Msg::Dir_out . . . 997
L4::Ipc::Msg::Do_out_data . . . 1000
L4::Ipc::Msg::Do_out_items . . . 1001
L4::Ipc::Msg::Elem< Array< A, LEN > &> . . . 1004
L4::Ipc::Msg::Elem< Array< A, LEN > > . . . 1005
L4::Ipc::Msg::Elem< Array_ref< A, LEN > &> . . . 1006
L4::Ipc::Msg::Svr_arg_pack< IPC_TYPE > . . . 1009
L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > . . . 1009
L4::Ipc::Msg_ptr< T > . . . 1010
L4::Ipc::Opt< T > . . . 1011
L4::Ipc::Ostream . . . 1013
L4::Ipc::Iostream . . . 974
L4::Ipc::Out< T > . . . 1019
L4::Ipc::Ret_array< T > . . . 1020
L4::Ipc::Send_only . . . 1021
L4::Ipc::Small_buf . . . 1021
L4::Ipc::Snd_item . . . 1023
L4::Ipc::Str_cp_in< T > . . . 1023
L4::Ipc::Varg . . . 1025
L4::Ipc::Varg_list_ref . . . 1031
L4::Ipc::Varg_list< MAX > . . . 1030
L4::Ipc_svr::Compound_reply . . . 1040
L4::Ipc_svr::Default_loop_hooks . . . 1041
L4Re::Util::Br_manager_hooks . . . 1384
L4::Ipc_svr::Default_setup_wait . . . 1042
L4::Ipc_svr::Default_timeout . . . 1043
L4::Ipc_svr::Default_loop_hooks . . . 1041
L4Re::Util::Br_manager_hooks . . . 1384
L4::Ipc_svr::Direct_dispatch< R > . . . 1044
L4::Ipc_svr::Exc_dispatch< R, Exc > . . . 1047
L4::Ipc_svr::Direct_dispatch< R * > . . . 1046
L4::Ipc_svr::Ignore_errors . . . 1048
L4::Ipc_svr::Default_loop_hooks . . . 1041
L4Re::Util::Br_manager_hooks . . . 1384
L4Re::Util::Br_manager_timeout_hooks . . . 1385
L4::Ipc_svr::Server_iface . . . 1049
L4::Ipc_svr::Br_manager_no_buffers . . . 1036
L4::Ipc_svr::Default_loop_hooks . . . 1041
L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > . . . 1065
L4Re::Util::Br_manager . . . 1379
L4Re::Util::Br_manager_hooks . . . 1384
L4::Ipc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager > . . . 1065
L4Re::Util::Br_manager_timeout_hooks . . . 1385
L4::Ipc_svr::Timed_work< HOOKS > . . . 1058
L4::Ipc_svr::Timeout_queue . . . 1061
L4::Irq_eoi . . . 1075
L4::Kobject_t< Icu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > > . . . 1105
L4::Icu . . . 938
L4::Kobject_t< Event, L4::Icu, L4RE_PROTO_EVENT > . . . 1105

L4Re::Event . . . 1326
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event > . . . 1101
L4vbus::Vbus . . . 1572
L4::Kobject_t< Scheduler, Icu, L4_PROTO_SCHEDULER, Type_info::Demand_t< 1 > > . . . 1105
L4::Scheduler . . . 1144
L4::Kobject_t< Vcon, Icu, L4_PROTO_LOG > . . . 1105
L4::Vcon . . . 1237
L4::Kobject_t< Log, L4::Vcon, L4::PROTO_EMPTY > . . . 1105
L4Re::Log . . . 1339
L4::Kobject_t< Triggerable, Irq_eoi, L4_PROTO_IRQ > . . . 1105
L4::Triggerable . . . 1193
L4::Kobject_2t< Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER > . . . 1097
L4::Irq . . . 1068
L4::Kobject_t< Irq_mux, Triggerable, L4_PROTO_IRQ_MUX > . . . 1105
L4::Irq_mux . . . 1081
L4::Kobject_t< Semaphore, Triggerable, L4_PROTO_SEMAPHORE > . . . 1105
L4::Semaphore . . . 1149
L4::Kip::Mem_desc . . . 1084
L4::Kobject . . . 1093
L4::Kobject_t< Arm_smccc, L4::Kobject, PROTO, Type_info::Demand_t<> > . . . 1105
L4::Kobject_t< Cmd_control, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> > . . . 1105
L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > > . . . 1105
L4Re::Dataspace . . . 1296
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event > . . . 1101
L4::Kobject_t< Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG > . . . 1105
L4Re::Debug_obj . . . 1306
L4::Kobject_t< Debugger, Kobject, L4_PROTO_DEBUGGER > . . . 1105
L4::Debugger . . . 906
L4::Kobject_t< Derived, L4::Kobject, PROTO, S_DEMAND > . . . 1105
L4::Kobject_t< Device, L4::Kobject, L4VIRTIO_PROTOCOL, L4::Type_info::Demand_t< 1 > > . . . 1105
L4::Kobject_t< Dma_space, L4::Kobject, PROTO, L4::Type_info::Demand_t< 1 > > . . . 1105
L4::Kobject_t< Exception, L4::Kobject, PROTO, Type_info::Demand_t<> > . . . 1105
L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY > . . . 1105
L4::Factory . . . 919
L4::Kobject_t< Mem_alloc, L4::Factory, L4::PROTO_EMPTY > . . . 1105
L4Re::Mem_alloc . . . 1342
L4::Kobject_t< Goos, L4::Kobject, L4RE_PROTO_GOOS > . . . 1105
L4Re::Video::Goos . . . 1485
L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR > . . . 1105
L4Re::Inhibitor . . . 1334
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event > . . . 1101
L4::Kobject_t< Io_pager, L4::Kobject, PROTO, Type_info::Demand_t<> > . . . 1105
L4::Kobject_t< Irq_eoi, L4::Kobject, PROTO, Type_info::Demand_t<> > . . . 1105
L4::Kobject_t< Meta, Kobject, L4_PROTO_META > . . . 1105
L4::Meta . . . 1112
L4::Kobject_t< Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE > . . . 1105
L4Re::Mmio_space . . . 1347
L4::Kobject_t< Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE, L4::Type_info::Demand_t< 1 > > . . . 1105
L4Re::Namespace . . . 1351
L4::Kobject_t< Parent, L4::Kobject, L4RE_PROTO_PARENT > . . . 1105
L4Re::Parent . . . 1360
L4::Kobject_t< Platform_control, Kobject, L4_PROTO_PLATFORM_CTL > . . . 1105
L4::Platform_control . . . 1122

L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > > . . . 1105
L4::Rcv_endpoint . . . 1132
L4::Kobject_2t< Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER > . . . 1097
L4::Kobject_t< Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > > 1105
L4::Ipc_gate . . . 1033
L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > > . . . 1105
L4::Task . . . 1169
L4::Kobject_t< Vm, Task, L4_PROTO_VM > . . . 1105
L4::Vm . . . 1244
L4::Kobject_t< Thread, Kobject, L4_PROTO_THREAD, Type_info::Demand_t< 1 > > . . . 1105
L4::Thread . . . 1177
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 > . . . 1104
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 . . . 1136
L4Re::Util::Object_registry . . . 1424
L4::Server< LOOP_HOOKS > . . . 1154
L4Re::Util::Registry_server< LOOP_HOOKS > . . . 1432
L4::Server_object . . . 1157
L4::Server_object_x< Derived, IFACE, BASE > . . . 1163
L4::Server_object_t< Kobject > . . . 1159
L4::Irq_handler_object . . . 1078
L4::Server_object_t< IFACE, BASE > . . . 1159
L4::Server_object_x< Derived, IFACE, BASE > . . . 1163
L4::String . . . 1169
L4::Thread::Attr . . . 1188
L4::Thread::Modify_senders . . . 1191
L4::Type_info . . . 1197
L4::Type_info::Demand . . . 1198
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 > . . . 1201
L4::Type_info::Demand_union_t< D1, D2 > . . . 1203
L4::Type_info::Demand_t< __I::Max< Kobject_typeid< T1 >::Demand ::Caps, Kobject_demand< T2... > ::Caps >::Res, Kobject_typeid< T1 >::Demand ::Flags|Kobject_demand< T2... > ::Flags, __I::Max< Kobject_typeid< T1 >::Demand ::Mem, Kobject_demand< T2... > :↩ :Mem >::Res, __I::Max< Kobject_typeid< T1 >::Demand ::Ports, Kobject_demand< T2... > ::Ports >::Res > . . . 1201
L4::Type_info::Demand_union_t< Kobject_typeid< T1 >::Demand, Kobject_demand< T2... > > . . . 1203
L4::Type_info::Demand_t<> . . . 1201
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > . . . 1201
L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y > . . . 1207
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... > . . . 1208
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y > . . . 1209
L4::Typeid::Detail::Rpcs_end . . . 1210
L4::Typeid::Detail::_Rpcs< L4::Opcode, 0, RPCS... > . . . 1206
L4::Typeid::Rpcs< RPCS > . . . 1215
L4::Typeid::Detail::_Rpcs< l4_umword_t, 0, ARG... > . . . 1206
L4::Typeid::Rpcs_sys< ARG > . . . 1220
L4::Typeid::Detail::_Rpcs< OPCODE_TYPE, 0, RPCS... > . . . 1206

L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS > . . . 1218
L4::Typeid::Detail::_Rpcs< void, 0, OPERATION > . . . 1206
L4::Typeid::Rpc_nocode< OPERATION > . . . 1212
L4::Typeid::Detail::_Rpcs< OPCODE, O, X > . . . 1206
L4::Typeid::P_dispatch< LIST > . . . 1211
L4::Typeid::Raw_ipc< CLASS > . . . 1212
L4::Typeid::Rpcs_code< OPCODE_TYPE > . . . 1217
L4::Types::Bool< V > . . . 1222
L4::Types::Flags< BITS_ENUM, UNDERLYING > . . . 1225
l4_buf_regs_t . . . 1247
l4_exc_regs_t . . . 1248
l4_fpage_t . . . 1251
l4_icu_info_t . . . 1251
L4::Icu::Info . . . 945
l4_icu_msi_info_t . . . 1253
l4_kernel_info_mem_desc_t . . . 1254
l4_kernel_info_t . . . 1255
l4_msg_regs_t . . . 1257
l4_msgtag_t . . . 1258
l4_sched_cpu_set_t . . . 1260
l4_sched_param_t . . . 1263
l4_snd_fpage_t . . . 1264
l4_thread_regs_t . . . 1265
l4_timeout_s . . . 1266
l4_timeout_t . . . 1267
l4_tracebuffer_status_t . . . 1268
l4_tracebuffer_status_window_t . . . 1270
l4_vcon_attr_t . . . 1271
l4_vcpu_ipc_regs_t . . . 1272
l4_vcpu_regs_t . . . 1273
l4_vcpu_state_t . . . 1277
L4vcpu::Vcpu . . . 1583
l4_vhw_descriptor . . . 1279
l4_vhw_entry . . . 1282
l4_vm_svm_vmcb_control_area . . . 1285
l4_vm_svm_vmcb_state_save_area . . . 1285
l4_vm_svm_vmcb_state_save_area_seg . . . 1287
l4_vm_svm_vmcb_t . . . 1287
l4_vm_tz_state . . . 1289
L4Re::Cap_alloc . . . 1289
L4Re::Dataspace::Stats . . . 1305
L4Re::Dma_space . . . 1308
L4Re::Env . . . 1313
L4Re::Event_buffer_t< PAYLOAD > . . . 1330
L4Re::Util::Event_buffer_t< PAYLOAD > . . . 1410
L4Re::Util::Event_buffer_consumer_t< PAYLOAD > . . . 1406
L4Re::Event_buffer_t< PAYLOAD >::Event . . . 1333
L4Re::Ned::Cmd_control . . . 1358
L4Re::Smart_cap_auto< Unmap_flags > . . . 1375
L4Re::Smart_count_cap< Unmap_flags > . . . 1376
L4Re::Util::Auto_cap< T > . . . 1377
L4Re::Util::Auto_del_cap< T > . . . 1378
L4Re::Util::Cap_alloc_base . . . 1388
L4Re::Util::Counter< COUNTER > . . . 1389
L4Re::Util::Counting_cap_alloc< COUNTERTYPE > . . . 1389
L4Re::Util::Dataspace_svr . . . 1396
L4Re::Util::Event_t< PAYLOAD > . . . 1414

L4Re::Util::Item_alloc_base . . . 1418
L4Re::Util::Names::Name_space . . . 1420
L4Re::Util::Ref_cap< T > . . . 1430
L4Re::Util::Ref_del_cap< T > . . . 1431
L4Re::Util::Smart_cap_auto< Unmap_flags > . . . 1436
L4Re::Util::Smart_count_cap< Unmap_flags > . . . 1437
L4Re::Util::Vcon_svr< SVR > . . . 1438
L4Re::Util::Video::Goos_svr . . . 1439
L4Re::Vfs::Directory . . . 1450
  L4Re::Vfs::File . . . 1455
    L4Re::Vfs::Be_file . . . 1443
L4Re::Vfs::File_system . . . 1457
  L4Re::Vfs::Be_file_system . . . 1447
L4Re::Vfs::Fs . . . 1459
  L4Re::Vfs::Ops . . . 1470
L4Re::Vfs::Generic_file . . . 1464
  L4Re::Vfs::File . . . 1455
L4Re::Vfs::Mman . . . 1469
  L4Re::Vfs::Ops . . . 1470
L4Re::Vfs::Regular_file . . . 1473
  L4Re::Vfs::File . . . 1455
L4Re::Vfs::Special_file . . . 1478
  L4Re::Vfs::File . . . 1455
L4Re::Video::Color_component . . . 1481
L4Re::Video::Goos::Info . . . 1491
L4Re::Video::Pixel_info . . . 1493
L4Re::Video::View . . . 1500
L4Re::Video::View::Info . . . 1505
l4re_aux_t . . . 1508
l4re_ds_stats_t . . . 1509
l4re_elf_aux_mword_t . . . 1509
l4re_elf_aux_t . . . 1510
l4re_elf_aux_vma_t . . . 1511
l4re_env_cap_entry_t . . . 1512
l4re_env_t . . . 1514
l4re_event_t . . . 1515
l4re_video_color_component_t . . . 1516
l4re_video_goos_info_t . . . 1517
l4re_video_pixel_info_t . . . 1518
l4re_video_view_info_t . . . 1519
l4re_video_view_t . . . 1521
l4util_idt_desc_t . . . 1522
l4util_idt_header_t . . . 1523
l4util_mb_addr_range_t . . . 1524
l4util_mb_apm_t . . . 1525
l4util_mb_drive_t . . . 1525
l4util_mb_info_t . . . 1527
l4util_mb_mod_t . . . 1529
l4util_mb_vbe_ctrl_t . . . 1530
l4util_mb_vbe_mode_t . . . 1531
L4vbus::Gpio_module::Pin_slice . . . 1549
L4vbus::Pm< DEC > . . . 1571
l4vbus_device_t . . . 1579
l4vbus_resource_t . . . 1580
L4vcpu::State . . . 1581
L4virtio::Ptr< T > . . . 1601

L4virtio::Svr::Bad_descriptor . . . 1605
L4virtio::Svr::Block_request< Ds_data > . . . 1614
L4virtio::Svr::Data_buffer . . . 1617
L4virtio::Svr::Dev_config . . . 1621
L4virtio::Svr::Dev_features . . . 1632
L4virtio::Svr::Dev_status . . . 1635
L4virtio::Svr::Device_t< DATA > . . . 1642
L4virtio::Svr::Driver_mem_list_t< DATA > . . . 1646
L4virtio::Svr::Driver_mem_region_t< DATA > . . . 1653
L4virtio::Svr::Request_processor . . . 1660
L4virtio::Svr::Virtqueue::Head_desc . . . 1673
L4virtio::Virtqueue . . . 1675
  L4virtio::Driver::Virtqueue . . . 1596
  L4virtio::Svr::Virtqueue . . . 1666
L4virtio::Virtqueue::Avail . . . 1688
L4virtio::Virtqueue::Avail::Flags . . . 1689
L4virtio::Virtqueue::Desc . . . 1691
L4virtio::Virtqueue::Desc::Flags . . . 1693
L4virtio::Virtqueue::Used . . . 1697
L4virtio::Virtqueue::Used::Flags . . . 1698
L4virtio::Virtqueue::Used_elem . . . 1700
l4virtio_block_config_t . . . 1701
l4virtio_block_header_t . . . 1703
l4virtio_config_hdr_t . . . 1704
l4virtio_config_queue_t . . . 1706
L4vbus::Pm< Device > . . . 1571
  L4vbus::Device . . . 1534
    L4vbus::Gpio_module . . . 1543
    L4vbus::Gpio_pin . . . 1550
    L4vbus::Icu . . . 1558
    L4vbus::Pci_dev . . . 1560
    L4vbus::Pci_host_bridge . . . 1565
L4virtio::Ptr< void > . . . 1601
cxx::Ref_ptr< L4Re::Vfs::File > . . . 836
cxx::Ref_ptr< Mount_tree > . . . 836
L4::Ipc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_in, CLASS > . . . 1009
L4::Ipc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_out, CLASS > . . . 1009
L4::Ipc::Msg::Svr_val_ops< L4::Ipc::Snd_fpage, Dir_in, CLASS > . . . 1009
L4::Ipc::Msg::Svr_val_ops< typename _Elem< A *> ::svr_type, typename Direction< A *>::type, typename Class< typename Detail::_Plain< A *>::type >::type > . . . 1009
L4::Ipc::Msg::Svr_val_ops< typename _Elem< A > ::svr_type, typename Direction< A >::type, typename Class< typename Detail::_Plain< A >::type >::type > . . . 1009
L4::Ipc::Msg::Svr_val_ops< typename _Elem< A const *> ::svr_type, typename Direction< A const *>::type, typename Class< typename Detail::_Plain< A const *>::type >::type > . . . 1009
L4::Ipc::Msg::Svr_val_ops< typename _Elem< Array_ref< A, LEN > &> ::svr_type, typename Direction< Array_ref< A, LEN > &>::type, typename Class< typename Detail::_Plain< Array_ref< A, LEN > &>::type >::type > . . . 1009
L4::Ipc::Msg::Svr_val_ops< typename _Elem< Array_ref< A, LEN > > ::svr_type, typename Direction< Array_ref< A, LEN > >::type, typename Class< typename Detail::_Plain< Array_ref< A, LEN > >::type >::type > . . . 1009
L4::Ipc::Msg::Svr_val_ops< typename _Elem< T > ::svr_type, typename Direction< T >::type, typename Class< typename Detail::_Plain< T >::type >::type > . . . 1009
cxx::Bitmap_base::Word< Bits > . . . 759
cxx::Bitmap_base::Word< Size > . . . 759

Chapter 10

Data Structure Index

10.1 Data Structures

Here are the data structures with brief descriptions:

cxx::Auto_ptr< T >
Smart pointer with automatic deletion 701
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >
AVL tree based associative container 706
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >
AVL set for simple compareable items 710
cxx::Avl_tree< Node, Get_key, Compare >
A generic AVL tree 713
cxx::Avl_tree_node
Node of an AVL tree 718
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >
Basic slab allocator 721
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >
Merged slab allocator (allocators for objects of the same size are merged together) 724
cxx::Bitfield< T, LSB, MSB >
Definition for a member (part) of a bit field 729
cxx::Bitfield< T, LSB, MSB >::Value< TT >
Internal helper type 739
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >
Internal helper type 740
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >
Internal helper type 742
cxx::Bitmap< BITS >
A static bit map 743
cxx::Bitmap_base
Basic bitmap abstraction 747
cxx::Bitmap_base::Bit
A writeable bit in a bitmap 758
cxx::Bitmap_base::Char< BITS >
Helper abstraction for a byte contained in the bitmap 759
cxx::Bitmap_base::Word< BITS >
Helper abstraction for a word contained in the bitmap 759
cxx::Bits::Avl_map_get_key< KEY_TYPE >
Key-getter for Avl_map 761
cxx::Bits::Avl_set_get_key< KEY_TYPE >
Internal, key-getter for Avl_set nodes 761

cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >
Internal: AVL set with internally managed nodes . . . 762
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node
A smart pointer to a tree item . . . 773
cxx::Bits::Basic_list< POLICY >
Internal: Common functions for all head-based list implementations . . . 775
cxx::Bits::Bst< Node, Get_key, Compare >
Basic binary search tree (BST) . . . 777
cxx::Bits::Bst_node
Basic type of a node in a binary search tree (BST) . . . 791
cxx::Bits::Direction
The direction to go in a binary search tree . . . 793
cxx::Bits::Smart_ptr_list< ITEM >
List of smart-pointer-managed objects . . . 795
cxx::Bits::Smart_ptr_list_item< T, STORE_T >
List item for an arbitrary item in a Smart_ptr_list . . . 797
cxx::H_list< T, POLICY >
General double-linked list of unspecified cxx::H_list_item elements . . . 798
cxx::H_list_item_t< ELEM_TYPE >
Basic element type for a double-linked H_list . . . 805
cxx::H_list_t< T >
Double-linked list of typed H_list_item_t elements . . . 807
cxx::List< D, Alloc >
Doubly linked list, with internal allocation . . . 810
cxx::List< D, Alloc >::Iter
Iterator . . . 813
cxx::List_alloc
Standard list-based allocator . . . 814
cxx::List_item
Basic list item . . . 818
cxx::List_item::Iter
Iterator for a list of ListItem-s . . . 824
cxx::List_item::T_iter< T, Poly >
Iterator for derived classes from ListItem . . . 827
cxx::Lt_functor< Obj >
Generic comparator class that defaults to the less-than operator . . . 830
cxx::New_allocator< _Type >
Standard allocator based on `operator new ()` . . . 830
cxx::Nothrow
Helper type to distinguish the `oeprator new` version that does not throw exceptions . . . 831
cxx::Pair< First, Second >
Pair of two values . . . 832
cxx::Pair_first_compare< Cmp, Typ >
Comparison functor for Pair . . . 833
cxx::Ref_obj_list_item< T >
Item for list linked via cxx::Ref_ptr with default refence counting . . . 835
cxx::Ref_ptr< T, CNT >
A reference-counting pointer with automatic cleanup . . . 836
cxx::S_list< T, POLICY >
Simple single-linked list . . . 842
cxx::Slab< Type, Slab_size, Max_free, Alloc >
Slab allocator for object of type *Type* . . . 845
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >
Merged slab allocator (allocators for objects of the same size are merged together) . . . 849
cxx::static_vector< T, IDX >
Simple encapsulation for a dynamically allocated array . . . 852
cxx::String
Allocation free string class with explicit length field . . . 854

cxx::Weak_ref< T >
Typed weak reference to an object of type T . . . 861
cxx::Weak_ref_base
Generic (base) weak reference to some object . . . 863
Elf32_Dyn
ELF32 dynamic entry . . . 866
Elf32_Ehdr
ELF32 header . . . 867
Elf32_Phdr
ELF32 program header . . . 869
Elf32_Shdr
ELF32 section header - figure 1-9, page 1-9 . . . 870
Elf32_Sym
ELF32 symbol table entry . . . 871
Elf64_Dyn
ELF64 dynamic entry . . . 872
Elf64_Ehdr
ELF64 header . . . 873
Elf64_Phdr
ELF64 program header . . . 875
Elf64_Shdr
ELF64 section header . . . 876
Elf64_Sym
ELF64 symbol table entry . . . 877
gfxbitmap_offset
Offsets in pmap[] and bmap[] . . . 878
L4::Alloc_list
A simple list-based allocator . . . 879
L4::Arm_smccc
Wrapper for function calls that follow the ARM SMC/HVC calling convention . . . 880
L4::Base_exception
Base class for all exceptions, thrown by the L4Re framework . . . 881
L4::Basic_registry
This registry returns the corresponding server object based on the label of an Ipc_gate . . . 883
L4::Bounds_error
Access out of bounds . . . 885
L4::Cap< T >
C++ interface for capabilities . . . 888
L4::Cap_base
Base class for all kinds of capabilities . . . 893
L4::Com_error
Error conditions during IPC . . . 904
L4::Debugger
C++ debugger interface . . . 906
L4::Element_already_exists
Exception for duplicate element insertions . . . 913
L4::Element_not_found
Exception for a failed lookup (element not found) . . . 915
L4::Exception
Exception interface . . . 917
L4::Exception_tracer
Back-trace support for exceptions . . . 918
L4::Factory
C++ Factory interface to create kernel objects . . . 919
L4::Factory::Lstr
Special type to add a pascal string into the factory create stream . . . 931
L4::Factory::Nil
Special type to add a void argument into the factory create stream . . . 932

L4::Factory::S
Stream class for the create() argument stream . . . 933
L4::Icu
C++ Icu interface . . . 938
L4::Icu::Info
This class encapsulates information about an ICU . . . 945
L4::Invalid_capability
Indicates that an invalid object was invoked . . . 946
L4::Io_pager
Io_pager interface . . . 950
L4::Iommu
Interface for IO-MMUs used for DMA remapping . . . 952
L4::IOModifier
Modifier class for the IO stream . . . 954
L4::Ipc::Array< ELEM_TYPE, LEN_TYPE >
Array data type for dynamically sized arrays in RPCs . . . 955
L4::Ipc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >
Server-side copy in buffer for Array . . . 958
L4::Ipc::Array_ref< ELEM_TYPE, LEN_TYPE >
Array reference data type for arrays located in the message . . . 959
L4::Ipc::As_value< T >
Pass the argument as plain data value . . . 961
L4::Ipc::Buf_item
RPC warpper for a receive item . . . 962
L4::Ipc::Call
RPC attribute for a standard RPC call . . . 963
L4::Ipc::Call_t< RIGHTS >
RPC attribute for an RPC call with required rights . . . 964
L4::Ipc::Call_zero_send_timeout
RPC attribute for an RPC call, with zero send timeout . . . 966
L4::Ipc::Cap< T >
Capability type for RPC interfaces (see `L4::Cap<T>`) . . . 967
L4::Ipc::Gen_fpage< T >
Generic RPC wrapper for L4 flex-pages . . . 970
L4::Ipc::In_out< T >
Mark an argument as in-out argument . . . 974
L4::Ipc::Iostream
Input/Output stream for IPC [un]marshalling . . . 974
L4::Ipc::Istream
Input stream for IPC unmarshalling . . . 981
L4::Ipc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >
Defines client-side handling of 'MTYPE' as RPC argument . . . 991
L4::Ipc::Msg::Cls_buffer
Marker type for receive buffer values . . . 992
L4::Ipc::Msg::Cls_data
Marker type for data values . . . 994
L4::Ipc::Msg::Cls_item
Marker type for item values . . . 995
L4::Ipc::Msg::Dir_in
Marker type for input values . . . 996
L4::Ipc::Msg::Dir_out
Marker type for output values . . . 997
L4::Ipc::Msg::Do_in_data
Marker for Input data . . . 998
L4::Ipc::Msg::Do_in_items
Marker for Input items . . . 999
L4::Ipc::Msg::Do_out_data
Marker for Output data . . . 1000

L4::Ipc::Msg::Do_out_items
Marker for Output items . . . 1001
L4::Ipc::Msg::Do_rcv_buffers
Marker for receive buffers . . . 1002
L4::Ipc::Msg::Elem< Array< A, LEN > &>
Array as output argument . . . 1004
L4::Ipc::Msg::Elem< Array< A, LEN > >
Array as input arguments . . . 1005
L4::Ipc::Msg::Elem< Array_ref< A, LEN > &>
Array_ref as output argument . . . 1006
L4::Ipc::Msg::Is_valid_rpc_type< T >
Type trait defining a valid RPC parameter type . . . 1007
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 . . . 1009
L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >
Defines server-side handling for `MTYPE` server arguments . . . 1009
L4::Ipc::Msg_ptr< T >
Pointer to an element of type T in an Ipc::Istream . . . 1010
L4::Ipc::Opt< T >
Attribute for defining an optional RPC argument . . . 1011
L4::Ipc::Ostream
Output stream for IPC marshalling . . . 1013
L4::Ipc::Out< T >
Mark an argument as a output value in an RPC signature . . . 1019
L4::Ipc::Ret_array< T >
Dynamically sized output array of type T . . . 1020
L4::Ipc::Send_only
RPC attribute for a send-only RPC . . . 1021
L4::Ipc::Small_buf
A receive item for receiving a single capability . . . 1021
L4::Ipc::Snd_item
RPC wrapper for a send item . . . 1023
L4::Ipc::Str_cp_in< T >
Abstraction for extracting a zero-terminated string from an Ipc::Istream . . . 1023
L4::Ipc::Varg
Variably sized RPC argument . . . 1025
L4::Ipc::Varg_list< MAX >
Self-contained list of variable-sized RPC parameters . . . 1030
L4::Ipc::Varg_list_ref
List of variable-sized RPC parameters as received by the server . . . 1031
L4::Ipc_gate
The C++ IPC gate interface . . . 1033
L4::Ipc_svr::Br_manager_no_buffers
Empty implementation of Server_iface . . . 1036
L4::Ipc_svr::Compound_reply
Mix in for LOOP_HOOKS to always use compound reply and wait . . . 1040
L4::Ipc_svr::Default_loop_hooks
Default LOOP_HOOKS . . . 1041
L4::Ipc_svr::Default_setup_wait
Mix in for LOOP_HOOKS for setup_wait no op . . . 1042
L4::Ipc_svr::Default_timeout
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout . . . 1043
L4::Ipc_svr::Direct_dispatch< R >
Direct disptach helper, for forwarding dispatch calls a registry *R* . . . 1044
L4::Ipc_svr::Direct_dispatch< R * >
Direct disptach helper, for forwarding dispatch calls via a pointer to a registry *R* . . . 1046

L4::Ipc_svr::Exc_dispatch< R, Exc >
Dispatch helper wrapping try {} catch {} around the dispatch call 1047
L4::Ipc_svr::Ignore_errors
Mix in for LOOP_HOOKS to ignore IPC errors 1048
L4::Ipc_svr::Server_iface
Interface for server-loop related functions 1049
L4::Ipc_svr::Timed_work< HOOKS >
DEPRECATED 1058
L4::Ipc_svr::Timeout
Callback interface for Timeout_queue 1058
L4::Ipc_svr::Timeout_queue
Timeout queue to be used in l4re server loop 1061
L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >
Loop hooks mixin for integrating a timeout queue into the server loop 1065
L4::Irq
C++ Irq interface 1068
L4::Irq_eoi
Interface for sending an acknowledge message to an object 1075
L4::Irq_handler_object
Server object base class for handling IRQ messages 1078
L4::Irq_mux
IRQ multiplexer for shared IRQs 1081
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 1093
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.. 1104
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 1122
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 1132

L4::Registry_iface
Abstract interface for object registries . . . 1136
L4::Runtime_error
Exception for an abstract runtime error . . . 1141
L4::Scheduler
C++ interface of the Scheduler kernel object . . . 1144
L4::Semaphore
Kernel-provided semaphore object . . . 1149
L4::Server< LOOP_HOOKS >
Basic server loop for handling client requests . . . 1154
L4::Server_object
Abstract server object to be used with L4::Server and L4::Basic_registry . . . 1157
L4::Server_object_t< IFACE, BASE >
Base class (template) for server implementing server objects . . . 1159
L4::Server_object_x< Derived, IFACE, BASE >
Helper class to implement p_dispatch based server objects . . . 1163
L4::Smart_cap< T, SMART >
Smart capability class . . . 1165
L4::String
A null-terminated string container class . . . 1169
L4::Task
C++ interface of the Task kernel object . . . 1169
L4::Thread
C++ L4 kernel thread interface . . . 1177
L4::Thread::Attr
Thread attributes used for control_commit() . . . 1188
L4::Thread::Modify_senders
Wrapper class for modifying senders . . . 1191
L4::Triggerable
Interface that allows an object to be triggered by some source . . . 1193
L4::Type_info
Dynamic Type Information for L4Re Interfaces . . . 1197
L4::Type_info::Demand
Data type for expressing the needed receive buffers at the server-side of an interface . . . 1198
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >
Template type statically describing demand of receive buffers . . . 1201
L4::Type_info::Demand_union_t< D1, D2 >
Template type statically describing the combination of two Demand object . . . 1203
L4::Typeid::Detail::_Rpcs< OPCODE, O, X >
Empty list of RPCs . . . 1206
L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y >
Find the given RPC in the list . . . 1207
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >
Non-empty list of RPCs . . . 1208
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y >
Find the given RPC in the list . . . 1209
L4::Typeid::Detail::Rpcs_end
Internal end-of-list marker . . . 1210
L4::Typeid::P_dispatch< LIST >
Use for protocol based dispatch stage . . . 1211
L4::Typeid::Raw_ipc< CLASS >
RPCs list for passing raw incoming IPC to the server object . . . 1212
L4::Typeid::Rpc_nocode< OPERATION >
List of RPCs of an interface using a single operation without an opcode . . . 1212
L4::Typeid::Rpcs< RPCS >
Standard list of RPCs of an interface . . . 1215
L4::Typeid::Rpcs_code< OPCODE_TYPE >
List of RPCs of an interface using a special opcode type . . . 1217

L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS > . . . 1218
L4::Typeid::Rpcs_sys< ARG >
List of RPCs typically used for kernel interfaces . . . 1220
L4::Types::Bool< V >
Boolean meta type . . . 1222
L4::Types::False
False meta value . . . 1223
L4::Types::Flags< BITS_ENUM, UNDERLYING >
Template for defining typical Flags bitmaps . . . 1225
L4::Types::Same< A, B >
Compare two data types for equality . . . 1230
L4::Types::True
True meta value . . . 1232
L4::Unknown_error
Exception for an unknown condition . . . 1234
L4::Vcon
C++ L4 Vcon interface . . . 1237
L4::Vm
Virtual machine . . . 1244
l4_buf_regs_t
Encapsulation of the buffer-registers block in the UTCB . . . 1247
l4_exc_regs_t
UTCB structure for exceptions . . . 1248
l4_fpage_t
L4 flexpage type . . . 1251
l4_icu_info_t
Info structure for an ICU . . . 1251
l4_icu_msi_info_t
Info to use for a specific MSI . . . 1253
l4_kernel_info_mem_desc_t
Memory descriptor data structure . . . 1254
l4_kernel_info_t
L4 Kernel Interface Page . . . 1255
l4_msg_regs_t
Encapsulation of the message-register block in the UTCB . . . 1257
l4_msgtag_t
Message tag data structure . . . 1258
l4_sched_cpu_set_t
CPU sets . . . 1260
l4_sched_param_t
Scheduler parameter set . . . 1263
l4_snd_fpage_t
Send-flex-page types . . . 1264
l4_thread_regs_t
Encapsulation of the thread-control-register block of the UTCB . . . 1265
l4_timeout_s
Basic timeout specification . . . 1266
l4_timeout_t
Timeout pair . . . 1267
l4_tracebuffer_status_t
Trace-buffer status . . . 1268
l4_tracebuffer_status_window_t
Trace-buffer status window descriptor . . . 1270
l4_vcon_attr_t
Vcon attribute structure . . . 1271
l4_vcpu_ipc_regs_t
VCPU message registers . . . 1272

l4_vcpu_regs_t
VCPU registers . . . 1273
l4_vcpu_state_t
State of a vCPU . . . 1277
l4_vhw_descriptor
Virtual hardware devices description . . . 1279
l4_vhw_entry
Description of a device . . . 1282
l4_vm_svm_vmcb_control_area
VMCB structure for SVM VMs . . . 1285
l4_vm_svm_vmcb_state_save_area
State save area structure for SVM VMs . . . 1285
l4_vm_svm_vmcb_state_save_area_seg
State save area segment selector struct . . . 1287
l4_vm_svm_vmcb_t
Control structure for SVM VMs . . . 1287
l4_vm_tz_state
State structure for TrustZone VMs . . . 1289
L4Re::Cap_alloc
Capability allocator interface . . . 1289
L4Re::Console
Console class . . . 1294
L4Re::Dataspace
Interface for memory-like objects . . . 1296
L4Re::Dataspace::Stats
Information about the dataspace . . . 1305
L4Re::Debug_obj
Debug interface . . . 1306
L4Re::Dma_space
DMA Address Space . . . 1308
L4Re::Env
C++ interface of the initial environment that is provided to an L4 task . . . 1313
L4Re::Event
Event class . . . 1326
L4Re::Event_buffer_t< PAYLOAD >
Event buffer class . . . 1330
L4Re::Event_buffer_t< PAYLOAD >::Event
Event structure used in buffer . . . 1333
L4Re::Inhibitor
Set of inhibitor locks, which inhibit specific actions when held . . . 1334
L4Re::Log
Log interface class . . . 1339
L4Re::Mem_alloc
Memory allocation interface . . . 1342
L4Re::Mmio_space
Interface for memory-like address space accessible via IPC . . . 1347
L4Re::Namespace
Name-space interface . . . 1351
L4Re::Ned::Cmd_control
Direct control interface for Ned . . . 1358
L4Re::Parent
Parent interface . . . 1360
L4Re::Rm
Region map . . . 1362
L4Re::Smart_cap_auto< Unmap_flags >
Helper for Auto_cap and Auto_del_cap . . . 1375
L4Re::Smart_count_cap< Unmap_flags >
Helper for Ref_cap and Ref_del_cap . . . 1376

L4Re::Util::Auto_cap< T >
Automatic capability that implements automatic free and unmap of the capability selector . . . 1377
L4Re::Util::Auto_del_cap< T >
Automatic capability that implements automatic free and unmap+delete of the capability selector 1378
L4Re::Util::Br_manager
Buffer-register (BR) manager for L4::Server . . . 1379
L4Re::Util::Br_manager_hooks
Predefined server-loop hooks for a server loop using the Br_manager . . . 1384
L4Re::Util::Br_manager_timeout_hooks
Predefined server-loop hooks for a server with using the Br_manager and a timeout queue . . . 1385
L4Re::Util::Cap_alloc_base
Capability allocator . . . 1388
L4Re::Util::Counter< COUNTER >
Counter for Counting_cap_alloc with variable data width . . . 1389
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >
Internal reference-counting cap allocator . . . 1389
L4Re::Util::Dataspace_svr
Dataspace server class . . . 1396
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >
An event buffer consumer . . . 1406
L4Re::Util::Event_buffer_t< PAYLOAD >
Event_buffer utility class . . . 1410
L4Re::Util::Event_t< PAYLOAD >
Convenience wrapper for getting access to an event object . . . 1414
L4Re::Util::Item_alloc_base
Item allocator . . . 1418
L4Re::Util::Names::Name
Name class . . . 1419
L4Re::Util::Names::Name_space
Abstract server-side implementation of the L4::Namespace interface . . . 1420
L4Re::Util::Object_registry
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread . . . 1424
L4Re::Util::Ref_cap< T >
Automatic capability that implements automatic free and unmap of the capability selector . . . 1430
L4Re::Util::Ref_del_cap< T >
Automatic capability that implements automatic free and unmap+delete of the capability selector 1431
L4Re::Util::Registry_server< LOOP_HOOKS >
A server loop object which has a Object_registry included . . . 1432
L4Re::Util::Smart_cap_auto< Unmap_flags >
Helper for Auto_cap and Auto_del_cap . . . 1436
L4Re::Util::Smart_count_cap< Unmap_flags >
Helper for Ref_cap and Ref_del_cap . . . 1437
L4Re::Util::Vcon_svr< SVR >
Console server template class . . . 1438
L4Re::Util::Video::Goos_svr
Goos server class . . . 1439
L4Re::Vfs::Be_file
Boiler plate class for implementing an open file for L4Re::Vfs . . . 1443
L4Re::Vfs::Be_file_system
Boilerplate class for implementing a L4Re::Vfs::File_system . . . 1447
L4Re::Vfs::Directory
Interface for a POSIX file that is a directory . . . 1450
L4Re::Vfs::File
The basic interface for an open POSIX file . . . 1455
L4Re::Vfs::File_system
Basic interface for an L4Re::Vfs file system . . . 1457

L4Re::Vfs::Fs
POSIX File-system related functionality . . . 1459
L4Re::Vfs::Generic_file
The common interface for an open POSIX file . . . 1464
L4Re::Vfs::Mman
Interface for the POSIX memory management . . . 1469
L4Re::Vfs::Ops
Interface for the POSIX backends for an application . . . 1470
L4Re::Vfs::Regular_file
Interface for a POSIX file that provides regular file semantics . . . 1473
L4Re::Vfs::Special_file
Interface for a POSIX file that provides special file semantics . . . 1478
L4Re::Video::Color_component
A color component . . . 1481
L4Re::Video::Goos
A goos . . . 1485
L4Re::Video::Goos::Info
Information structure of a goos . . . 1491
L4Re::Video::Pixel_info
Pixel information . . . 1493
L4Re::Video::View
View . . . 1500
L4Re::Video::View::Info
Information structure of a view . . . 1505
l4re_aux_t
Auxiliary descriptor . . . 1508
l4re_ds_stats_t
Information about the data space . . . 1509
l4re_elf_aux_mword_t
Auxiliary vector element for a single unsigned data word . . . 1509
l4re_elf_aux_t
Generic header for each auxiliary vector element . . . 1510
l4re_elf_aux_vma_t
Auxiliary vector element for a reserved virtual memory area . . . 1511
l4re_env_cap_entry_t
Entry in the L4Re environment array for the named inital objects . . . 1512
l4re_env_t
Initial environment data structure . . . 1514
l4re_event_t
Event structure used in buffer . . . 1515
l4re_video_color_component_t
Color component structure . . . 1516
l4re_video_goos_info_t
Goos information structure . . . 1517
l4re_video_pixel_info_t
Pixel_info structure . . . 1518
l4re_video_view_info_t
View information structure . . . 1519
l4re_video_view_t
C representation of a goos view . . . 1521
l4util_idt_desc_t
IDT entry . . . 1522
l4util_idt_header_t
Header of an IDT table . . . 1523
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 . . . 1524

l4util_mb_apm_t
APM BIOS info . . . 1525
l4util_mb_drive_t
Drive Info structure . . . 1525
l4util_mb_info_t . . . 1527
l4util_mb_mod_t . . . 1529
l4util_mb_vbe_ctrl_t
VBE controller information . . . 1530
l4util_mb_vbe_mode_t
VBE mode information . . . 1531
L4vbus::Device
Device on a L4vbus::Vbus . . . 1534
L4vbus::Gpio_module
A Gpio_module groups multiple GPIO pins together . . . 1543
L4vbus::Gpio_module::Pin_slice
A slice of the pins provided by this module . . . 1549
L4vbus::Gpio_pin
A GPIO pin . . . 1550
L4vbus::Icu
Vbus Interrupt controller API . . . 1558
L4vbus::Pci_dev
A PCI device . . . 1560
L4vbus::Pci_host_bridge
A Pci host bridge . . . 1565
L4vbus::Pm< DEC >
Power-management API mixin . . . 1571
L4vbus::Vbus
The virtual bus (Vbus) interface . . . 1572
l4vbus_device_t
Detailed information about a vbus device . . . 1579
l4vbus_resource_t
Description of a single vbus resource . . . 1580
L4vcpu::State
C++ implementation of state word in the vCPU area . . . 1581
L4vcpu::Vcpu
C++ implementation of the vCPU save state area . . . 1583
L4virtio::Driver::Virtqueue
Driver-side implementation of a Virtqueue . . . 1596
L4virtio::Ptr< T >
Pointer used in virtio descriptors . . . 1601
L4virtio::Svr::Bad_descriptor
Exception used by Queue to indicate descriptor errors . . . 1605
L4virtio::Svr::Block_dev< Ds_data >
Base class for virtio block devices . . . 1607
L4virtio::Svr::Block_request< Ds_data >
A request to read or write data . . . 1614
L4virtio::Svr::Data_buffer
Abstract data buffer . . . 1617
L4virtio::Svr::Dev_config
Abstraction for L4-Virtio device config memory . . . 1621
L4virtio::Svr::Dev_features
Type for device feature bitmap . . . 1632
L4virtio::Svr::Dev_status
Type of the device status register . . . 1635
L4virtio::Svr::Device_t< DATA >
Server-side L4-VIRTIO device stub . . . 1642
L4virtio::Svr::Driver_mem_list_t< DATA >
List of driver memory regions assigned to a single L4-VIRTIO transport instance . . . 1646

L4virtio::Svr::Driver_mem_region_t< DATA >
Region of driver memory, that shall be managed locally . . . 1653
L4virtio::Svr::Request_processor
Encapsulate the state for processing a VIRTIO request . . . 1660
L4virtio::Svr::Virtqueue
Virtqueue implementation for the device . . . 1666
L4virtio::Svr::Virtqueue::Head_desc
VIRTIO request, essentially a descriptor from the available ring . . . 1673
L4virtio::Virtqueue
Low-level Virtqueue . . . 1675
L4virtio::Virtqueue::Avail
Type of available ring, this is read-only for the host . . . 1688
L4virtio::Virtqueue::Avail::Flags
Flags of the available ring . . . 1689
L4virtio::Virtqueue::Desc
Descriptor in the descriptor table . . . 1691
L4virtio::Virtqueue::Desc::Flags
Type for descriptor flags . . . 1693
L4virtio::Virtqueue::Used
Used ring . . . 1697
L4virtio::Virtqueue::Used::Flags
Flags for the used ring . . . 1698
L4virtio::Virtqueue::Used_elem
Type of an element of the used ring . . . 1700
l4virtio_block_config_t
Device configuration for block devices . . . 1701
l4virtio_block_header_t
Header structure of a request for a block device . . . 1703
l4virtio_config_hdr_t
L4-VIRTIO config header, provided in shared data space . . . 1704
l4virtio_config_queue_t
Queue configuration entry . . . 1706

Chapter 11

File Index

11.1 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 2066
amd64/l4/sys/consts.h
Common L4 constants, amd64 version 2080
amd64/l4/sys/**ktrace_events.h** . . . . . ??
amd64/l4/sys/l4int.h
Fixed sized integer types, amd64 version 2173
amd64/l4/sys/linkage.h
Linkage 1780
amd64/l4/sys/segment.h
Segment handling 1755
amd64/l4/sys/utcb.h
UTCB definitions for amd64 2228
amd64/l4/sys/**vm.h** . . . . . ??
amd64/l4/util/apic.h
APIC for AMD64 1709
amd64/l4/util/bitops_arch.h
Amd64 bit manipulation functions 1786
amd64/l4/util/cpu.h
CPU related functions 1794
amd64/l4/util/idt.h
IDT related functions 1720
amd64/l4/util/irq.h
Some PIC and hardware interrupt related functions 1890
amd64/l4/util/l4_macros.h
Main function 1799
amd64/l4/util/mbi_argv.h
Command line handling 1802
amd64/l4/util/perform.h
Perfomance Monitoring using P5/P6 Measurement Counters 1723
amd64/l4/util/port_io.h
Port I/O functions 1765

amd64/l4/util/rdtsc.h
Time stamp counter related functions . . . 1735
amd64/l4/util/spin.h
Spinning for amd64 . . . 1745
amd64/l4/util/stack_impl.h
Stack utilities for amd64 . . . 1806
amd64/l4/util/util.h
Utilities, amd64 version . . . 1747
amd64/l4f/l4/sys/**ipc.h** . . . **??**
amd64/l4f/l4/sys/segment.h
L4f specific fs/gs manipulation . . . 1752
amd64/l4f/l4/util/port_io.h
Port I/O functions . . . 1764
amd64/l4f/l4/util/setjmp.h
Inter-thread setjmp/longjmp . . . 1773
arm/l4/sys/**__kip-arch.h** . . . **??**
arm/l4/sys/**__vcpu-arch.h** . . . **??**
arm/l4/sys/atomic.h
Atomic memory modifications . . . 2261
arm/l4/sys/cache.h
Cache functions . . . 2063
arm/l4/sys/consts.h
Common L4 constants, arm version . . . 2079
arm/l4/sys/**ktrace_events.h** . . . **??**
arm/l4/sys/l4int.h
Fixed sized integer types, arm version . . . 2173
arm/l4/sys/linkage.h
Linkage . . . 1779
arm/l4/sys/mem_op.h
Memory access functions (ARM specific) . . . 1781
arm/l4/sys/thread.h
ARM-specific thread related definitions . . . 2339
arm/l4/sys/utcb.h
UTCB definitions for ARM . . . 2226
arm/l4/sys/vm.h
ARM virtualization interface . . . 1784
arm/l4/util/bitops_arch.h
ARM specific implementation of bitops functions . . . 1785
arm/l4/util/cpu.h
CPU related functions . . . 1793
arm/l4/util/irq.h
ARM specific implementation of irq functions . . . 1889
arm/l4/util/l4_macros.h
Main function . . . 1798
arm/l4/util/mbi_argv.h
Multiboot . . . 1801
arm/l4/util/stack_impl.h
Stack utilities, arm version . . . 1805
arm/l4f/l4/sys/ipc.h
L4 IPC System Calls, ARM . . . 2151
arm/l4f/l4/sys/syscall_defs.h
Syscall entry definitions . . . 1808
contrib/libio-io/l4/io/io.h . . . 1809
contrib/libio-io/l4/io/**types.h** . . . **??**
l4/cxx/alloc.h
Alloc list . . . 2248
l4/cxx/**arith** . . . **??**

l4/cxx/atomic.h
Atomic template . . . 2262
l4/cxx/**auto_ptr** . . . ??
l4/cxx/avl_map
AVL map . . . 1819
l4/cxx/avl_set
AVL set . . . 1822
l4/cxx/avl_tree
AVL tree . . . 1826
l4/cxx/basic_ostream
Basic IO stream . . . 1832
l4/cxx/basic_vector.h
Basic vector . . . 1836
l4/cxx/**bitfield** . . . ??
l4/cxx/**bitmap** . . . ??
l4/cxx/**dlist** . . . ??
l4/cxx/exceptions
Base exceptions . . . 1847
l4/cxx/**hlist** . . . ??
l4/cxx/iostream
IO Stream . . . 1852
l4/cxx/**iostream.h** . . . ??
l4/cxx/ipc_helper
IPC helper . . . 1853
l4/cxx/ipc_server
IPC server loop . . . 2098
l4/cxx/ipc_stream
IPC stream . . . 1855
l4/cxx/**ipc_timeout_queue** . . . ??
l4/cxx/l4iostream
L4 IO stream . . . 1876
l4/cxx/**l4iostream.h** . . . ??
l4/cxx/l4types.h
L4 Types . . . 1878
l4/cxx/**list** . . . ??
l4/cxx/**list_alloc** . . . ??
l4/cxx/main_thread
Main thread . . . 1879
l4/cxx/**minmax** . . . ??
l4/cxx/**observer** . . . ??
l4/cxx/pair
Pair implementation . . . 1881
l4/cxx/**ref_ptr** . . . ??
l4/cxx/**ref_ptr_list** . . . ??
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 . . . 1883
l4/cxx/**std_ops** . . . ??
l4/cxx/**string** . . . ??
l4/cxx/string.h
String . . . 1884
l4/cxx/thread
Thread implementation . . . 2217
l4/cxx/**type_list** . . . ??

l4/cxx/**type_traits** . . . . . ??
l4/cxx/**unique_ptr** . . . . . ??
l4/cxx/**unique_ptr_list** . . . . . ??
l4/cxx/**utils** . . . . . ??
l4/cxx/**weak_ref** . . . . . ??
l4/cxx/bits/bst.h
AVL tree 1837
l4/cxx/bits/bst_base.h
AVL tree 1841
l4/cxx/bits/bst_iter.h
AVL tree 1844
l4/cxx/bits/**list_basics.h** . . . . . ??
l4/cxx/bits/**smart_ptr_list.h** . . . . . ??
l4/cxx/bits/**type_traits.h** . . . . . ??
l4/irq/irq.h
IRQ handling routines 1886
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/**virtqueue** . . . . . ??
l4/l4virtio/server/**l4virtio** . . . . . ??
l4/l4virtio/server/**virtio** . . . . . ??
l4/l4virtio/server/**virtio-block** . . . . . ??
l4/libedid/edid.h 1900
l4/libgfxbitmap/bitmap.h
Bitmap renderer header file 1902
l4/libgfxbitmap/font.h
Bitmap font renderer header file 1905
l4/libgfxbitmap/support
Terminal support functionality 1907
l4/re/cap_alloc
Abstract capability-allocator interface 1944
l4/re/**console** . . . . . ??
l4/re/consts
Constants 1951
l4/re/consts.h
Constants 2082
l4/re/dataspace
Dataspace interface 1953
l4/re/dataspace-sys.h
Dataspace protocol defintion 1955
l4/re/debug
Debug interface 1956
l4/re/dma_space 1958
l4/re/elf_aux.h
Auxiliary information for binaries 1961

l4/re/env
Environment interface . . . 1963
l4/re/env.h
Environment interface . . . 1965
l4/re/error_helper
Error helper . . . 1969
l4/re/**event** . . . **??**
l4/re/**event-sys.h** . . . **??**
l4/re/event.h
Events . . . 1917
l4/re/**event_enums.h** . . . **??**
l4/re/**inhibitor** . . . **??**
l4/re/**inhibitor-sys.h** . . . **??**
l4/re/l4aux.h
Auxiliary definitions . . . 1984
l4/re/log
Log interface . . . 1985
l4/re/log-sys.h
Log protocol definition . . . 1987
l4/re/mem_alloc
Memory allocator interface . . . 1988
l4/re/mem_alloc-sys.h
Memory allocator protocol definitions . . . 1990
l4/re/**mmio_space** . . . **??**
l4/re/namespace
Namespace interface . . . 1992
l4/re/namespace-sys.h
Namespace protocol definitions . . . 1994
l4/re/parent
Parent interface . . . 1995
l4/re/parent-sys.h
Parent protocol definition . . . 1997
l4/re/protocols.h
L4Re Protocol Constants (C version) . . . 1998
l4/re/rm
Region mapper interface . . . 2000
l4/re/rm-sys.h
Region mapper protocol definitions . . . 2005
l4/re/shared_cap
Shared_cap / Shared_del_cap . . . 2006
l4/re/unique_cap
Unique_cap / Unique_del_cap . . . 2011
l4/re/c/dataspace.h
Data space C interface . . . 1908
l4/re/c/debug.h
Debug C interface . . . 1911
l4/re/c/dma_space.h
DMA space C interface . . . 1912
l4/re/c/event.h
Event C interface . . . 1915
l4/re/c/**event_buffer.h** . . . **??**
l4/re/c/**inhibitor.h** . . . **??**
l4/re/c/log.h
Log C interface . . . 1919
l4/re/c/mem_alloc.h
Memory allocator C interface . . . 1921
l4/re/c/namespace.h
Namespace functions, C interface . . . 1924

l4/re/c/rm.h
Region map interface, C interface . . . 1926
l4/re/c/util/cap_alloc.h
Capability allocator C interface . . . 1930
l4/re/c/util/kumem_alloc.h
Kumem allocator utility C interface . . . 1931
l4/re/c/util/video/goos_fb.h
Framebuffer utility functionality . . . 1933
l4/re/c/video/colors.h . . . 1935
l4/re/c/video/goos.h . . . 1937
l4/re/c/video/view.h . . . 1940
l4/re/impl/dataspace_impl.h
Dataspace client stub implementation . . . 1975
l4/re/impl/mem_alloc_impl.h
Memory allocator client stub implementation . . . 1977
l4/re/impl/namespace_impl.h
Namespace client stub implementation . . . 1979
l4/re/impl/rm_impl.h
Region map client stub implementation . . . 1981
l4/re/util/bitmap_cap_alloc
Bitmap capability allocator . . . 2016
l4/re/util/**br_manager** . . . **??**
l4/re/util/cap
Capability utility functions . . . 2018
l4/re/util/cap_alloc
Capability allocator . . . 1947
l4/re/util/cap_alloc_impl.h
Capability allocator implementation . . . 2020
l4/re/util/counting_cap_alloc
Reference-counting capability allocator . . . 2022
l4/re/util/**dataspace_svr** . . . **??**
l4/re/util/**debug** . . . **??**
l4/re/util/**env_ns** . . . **??**
l4/re/util/event . . . 1972
l4/re/util/**event_buffer** . . . **??**
l4/re/util/**event_svr** . . . **??**
l4/re/util/**icu_svr** . . . **??**
l4/re/util/item_alloc
Item allocator . . . 2025
l4/re/util/kumem_alloc
Kumem allocator helper . . . 2027
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 . . . 2029
l4/re/util/**region_mapping_svr** . . . **??**
l4/re/util/**region_mapping_svr_2** . . . **??**
l4/re/util/shared_cap
Shared_cap / Shared_del_cap . . . 2009
l4/re/util/unique_cap
Unique_cap / Unique_del_cap . . . 2013
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 . . . 2035
l4/re/video/**view** . . . **??**
l4/shmc/shmc.h
Shared memory library header file . . . 2036
l4/sigma0/sigma0.h
Sigma0 interface . . . 2041
l4/sys/__kernel_object_impl.h
Low-level kernel debugger functions . . . 2044
l4/sys/**__kip-32bit.h** . . . **??**
l4/sys/**__kip-64bit.h** . . . **??**
l4/sys/__ktrace-impl.h
L4 kernel event tracing . . . 2046
l4/sys/**__l4_fpage.h** . . . **??**
l4/sys/**__timeout.h** . . . **??**
l4/sys/__typeinfo.h
Type information handling . . . 2048
l4/sys/**__vcpu-arm.h** . . . **??**
l4/sys/**__vm-svm.h** . . . **??**
l4/sys/**__vm-vmx.h** . . . **??**
l4/sys/**arm_smccc** . . . **??**
l4/sys/**arm_smccc.h** . . . **??**
l4/sys/assert.h
Low-level assert implementation . . . 2251
l4/sys/cache.h
Cache-consistency functions . . . 2061
l4/sys/capability
L4::Cap related definitions . . . 2068
l4/sys/compiler.h
L4 compiler related defines . . . 2072
l4/sys/consts.h
Common constants . . . 2075
l4/sys/debugger
The debugger interface specifies common debugging related definitions . . . 2115
l4/sys/debugger.h
Debugger related definitions . . . 2117
l4/sys/err.h
Error codes . . . 2124
l4/sys/exception
Exception C++ interface . . . 2126
l4/sys/factory
Common factory related definitions . . . 2128
l4/sys/factory.h
Common factory related definitions . . . 2132
l4/sys/icu
Interrupt controller . . . 2138
l4/sys/icu.h
Interrupt controller . . . 2139
l4/sys/**iommu** . . . **??**
l4/sys/ipc.h
Common IPC interface . . . 2145
l4/sys/ipc_gate
The C++ IPC gate interface . . . 2154
l4/sys/ipc_gate.h
The C IPC gate interface . . . 2156
l4/sys/irq
C++ Irq interface . . . 2159

l4/sys/irq.h
C Irq interface . . . 1895
l4/sys/**kdebug.h** . . . **??**
l4/sys/kernel_object.h
Kernel object system calls . . . 2163
l4/sys/kip . . . 2165
l4/sys/kip.h
Kernel Info Page access functions . . . 2299
l4/sys/**kobject** . . . **??**
l4/sys/ktrace.h
L4 kernel event tracing . . . 2168
l4/sys/l4int.h
Fixed sized integer types, generic version . . . 2171
l4/sys/memdesc.h
Memory description functions . . . 2175
l4/sys/meta
Meta interface for getting dynamic type information about objects behind capabilities . . . 2179
l4/sys/pager
Pager and Io_pager C++ interface . . . 2181
l4/sys/platform_control
Platform control object . . . 2183
l4/sys/platform_control.h
Platform control object . . . 2185
l4/sys/rcv_endpoint
The C++ Receive endpoint interface . . . 2189
l4/sys/rcv_endpoint.h
Receive endpoint C interface . . . 2191
l4/sys/scheduler
Scheduler object functions . . . 2193
l4/sys/scheduler.h
Scheduler object functions . . . 2195
l4/sys/semaphore
Semaphore class definition . . . 2200
l4/sys/**semaphore.h** . . . **??**
l4/sys/smart_capability
L4::Capability class . . . 2202
l4/sys/task
Common task related definitions . . . 2206
l4/sys/task.h
Common task related definitions . . . 2209
l4/sys/thread
Common thread related definitions . . . 2214
l4/sys/thread.h
Common thread related definitions . . . 2329
l4/sys/typeinfo_svr
Type information server template . . . 2219
l4/sys/types.h
Common L4 ABI Data Types . . . 1813
l4/sys/utcb.h
UTCB definitions . . . 2221
l4/sys/vcon
Virtual console interface . . . 2234
l4/sys/vcon.h
Virtual console interface . . . 2236
l4/sys/**vcpu.h** . . . **??**
l4/sys/vhw.h
Descriptors for virtual hardware (under UX) . . . 2242

l4/sys/vm
Virtualization interface . . . 2245
l4/sys/cxx/**capability.h** . . . **??**
l4/sys/cxx/**ipc_array** . . . **??**
l4/sys/cxx/**ipc_basics** . . . **??**
l4/sys/cxx/ipc_client . . . 2083
l4/sys/cxx/**ipc_epiface** . . . **??**
l4/sys/cxx/ipc_iface
Interface Definition Language . . . 2086
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 . . . 2100
l4/sys/cxx/**ipc_varg** . . . **??**
l4/sys/cxx/smart_capability_1x . . . 2109
l4/sys/cxx/types . . . 2113
l4/util/alloc.h
Allocator using a bit-array . . . 2246
l4/util/assert.h
Some useful assert-style macros . . . 2249
l4/util/atomic.h
Atomic operations header and generic implementations . . . 2254
l4/util/backtrace.h
Backtrace . . . 2264
l4/util/base64.h
Base 64 encoding and decoding functions adapted from Bob Trower 08/04/01 . . . 2265
l4/util/bitops.h
Bit manipulation functions . . . 2267
l4/util/elf.h
ELF definition . . . 2272
l4/util/getopt.h
Getopt . . . 2295
l4/util/keymap.h
Event to ASCII key mapping . . . 2296
l4/util/kip.h . . . 2297
l4/util/kprintf.h
Printf using the kernel debugger . . . 2302
l4/util/l4_macros.h
Some useful generic macros, L4f version . . . 1799
l4/util/list_alloc.h
Simple list-based allocator . . . 2303
l4/util/**llulc.h** . . . **??**
l4/util/lock.h
Simple lock implementation . . . 2306
l4/util/mb_info.h
Multiboot info structure as defined by GRUB . . . 2308
l4/util/parse_cmd.h
Comfortable command-line parsing . . . 2315
l4/util/rand.h
Simple Pseudo-Random Number Generator . . . 2316
l4/util/reboot.h
Machine restarting functions . . . 2318
l4/util/sll.h
List implemenation . . . 2319
l4/util/splitlog2.h
Split a range in log2 aligned and size-aligned chunks . . . 2323

l4/util/stack.h
Some helper functions for stack manipulation . . . 2324
l4/util/thread.h
Low-level Thread Functions . . . 2326
l4/util/**util.h** . . . ??
l4/vbus/**vbus** . . . ??
l4/vbus/vbus.h
Description of the vbus C API . . . 2340
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** . . . ??
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 . . . 2343
l4/vbus/**vdevice-ops.h** . . . ??
l4/vcpu/**vcpu** . . . ??
l4/vcpu/**vcpu.h** . . . ??
l4/vcpu/vmx/**vmcs.h** . . . ??
x86/l4/sys/**__kip-arch.h** . . . ??
x86/l4/sys/**__vcpu-arch.h** . . . ??
x86/l4/sys/cache.h
Cache functions . . . 2067
x86/l4/sys/consts.h
Common L4 constants, x86 version . . . 2081
x86/l4/sys/**ipc-invoke.h** . . . ??
x86/l4/sys/**ktrace_events.h** . . . ??
x86/l4/sys/l4int.h
Fixed sized integer types, x86 version . . . 2174
x86/l4/sys/linkage.h
Linkage . . . 1780
x86/l4/sys/segment.h
Segment handling . . . 1762
x86/l4/sys/**syscall-invoke.h** . . . ??
x86/l4/sys/utcb.h
UTCB definitions for X86 . . . 2231
x86/l4/sys/**vm.h** . . . ??
x86/l4/util/apic.h
APIC for X86 . . . 1714
x86/l4/util/bitops_arch.h
X86 bit manipulation functions . . . 1789
x86/l4/util/cpu.h
CPU related functions . . . 1796
x86/l4/util/idt.h
IDT related functions . . . 1722
x86/l4/util/irq.h
Some PIC and hardware interrupt related functions . . . 1893
x86/l4/util/l4_macros.h
Main function . . . 1800

x86/l4/util/mbi_argv.h
Command line handling 1803
x86/l4/util/perform.h
Perfomance Monitoring using P5/P6 Measurement Counters 1729
x86/l4/util/port_io.h
X86 port I/O 1769
x86/l4/util/rdtsc.h
Time stamp counter related functions 1740
x86/l4/util/spin.h
Spinning for x86 1746
x86/l4/util/stack_impl.h
Stack utilities for x86 1807
x86/l4/util/util.h
Utilities for x86 1750
x86/l4f/l4/sys/**ipc-l42-gcc3-nopic.h** . . . . . **??**
x86/l4f/l4/sys/ipc.h
L4 IPC System Calls, x86 2153
x86/l4f/l4/sys/segment.h
L4f specific segment manipulation 1760
x86/l4f/l4/util/port_io.h
Port I/O functions 1766
x86/l4f/l4/util/setjmp.h
Inter-thread setjmp/longjmp 1776

Chapter 12

Module Documentation

12.1 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.

12.1.1 Detailed Description

12.2 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)

12.2.1 Detailed Description

12.2.2 Function Documentation

12.2.2.1 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 424 of file atomic.h.

12.2.2.2 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 476 of file atomic.h.

12.2.2.3 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 436 of file atomic.h.

12.2.2.4 l4util_atomic_inc()

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

Atomic increment.

Parameters

dest	destination operand

Definition at line 379 of file atomic.h.

12.2.2.5 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 335 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:

12.2.2.6 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 319 of file atomic.h.

12.2.2.7 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 311 of file atomic.h.

12.2.2.8 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 303 of file atomic.h.

12.2.2.9 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 327 of file atomic.h.

12.2.2.10 l4util_inc8()

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

Parameters

dest	destination operand

Definition at line 367 of file atomic.h.

12.2.2.11 l4util_inc8_res()

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

Parameters

dest	destination operand

Returns

res

Definition at line 396 of file atomic.h.

12.2.2.12 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 361 of file atomic.h.

12.2.2.13 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 349 of file atomic.h.

12.2.2.14 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 343 of file atomic.h.

12.2.2.15 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 355 of file atomic.h.

12.3 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.

12.3.1 Detailed Description

12.4 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 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.

12.4.1 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.

12.5 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

- void l4_barrier (void)

  Memory barrier.
- void l4_mb (void)

  Memory barrier.
- void l4_wmb (void)

  Write memory barrier.

12.5.1 Detailed Description

L4 standard macros for header files, function definitions, and public APIs etc.

Include File

```
#include <l4/sys/compiler.h>
```

12.5.2 Macro Definition Documentation

12.5.2.1 L4_ALWAYS_INLINE

```
#define L4_ALWAYS_INLINE
```

L4 Inline function attribute.

Always inline a function

Definition at line 67 of file compiler.h.

12.5.2.2 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 212 of file compiler.h.

12.5.2.3 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 marke with L4_NOTHROW if they never do. In C++ this is equvalent to `throw()`.

```
int foo() L4_NOTHROW;
...
int foo() L4_NOTHROW
{
  ...
  return result;
}
```

Definition at line 185 of file compiler.h.

Referenced by l4_msgtag_t::has_error(), l4_round_page(), l4_sleep_forever(), l4_trunc_page(), l4_trunc_size(), l4_vm_vmx_field_len(), l4vcpu_irq_disable_save(), and l4virtio_get_feature().

12.6 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 (const unsigned long val)

  Find the next power of 2 for a given number.

12.6.1 Detailed Description

12.6.2 Function Documentation

12.6.2.1 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:

12.6.2.2 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().

Here is the caller graph for this function:

12.6.2.3 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.

12.6.2.4 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:

12.6.2.5 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:

12.6.2.6 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:

12.6.2.7 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.

12.6.2.8 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.

12.6.2.9 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.

12.6.2.10 l4util_next_power2()

```
int l4util_next_power2 (
            const 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.

12.6.2.11 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:

12.6.2.12 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.

12.7 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

12.7.1 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.

12.8 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 }

  Constants for buffer descriptors.

Functions

- void l4_utcb_inherit_fpu (int switch_on) L4_NOTHROW

  Enable or disable inheritance of FPU state to receiver.

12.8.1 Detailed Description

12.8.2 Enumeration Type Documentation

12.8.2.1 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 219 of file consts.h.

12.9 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.

12.9.1 Detailed Description

12.10 C++ IPC Interface Definition.

APIs for defining IPC interfaces using C++ as language.

Collaboration diagram for C++ IPC Interface Definition.:

Modules

- Internal Helpers

Namespaces

- L4::Typeid

  Definition of interface data-type helpers.

12.10.1 Detailed Description

APIs for defining IPC interfaces using C++ as language.

12.11 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.

12.11.1 Detailed Description

12.11.2 Function Documentation

12.11.2.1 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 EXTERN_C_END, l4util_cpu_capabilities_nocheck(), and l4util_cpu_has_cpuid().

Here is the call graph for this function:

12.11.2.2 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:

12.11.2.3 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:

12.12 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.

12.12.1 Detailed Description

Various functions for cache consistency.

Include File

```
#include <l4/sys/cache.h>
```

12.12.2 Function Documentation

12.12.2.1 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 84 of file cache.h.

12.12.2.2 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 108 of file cache.h.

12.12.2.3 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 116 of file cache.h.

12.12.2.4 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 92 of file cache.h.

12.12.2.5 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 100 of file cache.h.

12.13 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_MASK, L4_INVALID_CAP }

  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 = L4_BASE_CA↩PS_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.

12.13.1 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.

12.13.2 Enumeration Type Documentation

12.13.2.1 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.

12.13.2.2 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

This 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.
L4_BASE_LOG_CAP	Capability selector for the log object.
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.
L4_BASE_DEBUGGER_CAP	Capability selector for the debugger cap.
L4_BASE_ARM_SMCCC_CAP	Capability selector for the ARM SMCCC cap.
L4_BASE_CAPS_LAST	Last capability index used for base capabilities.

Definition at line 240 of file consts.h.

12.13.3 Function Documentation

12.13.3.1 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 400 of file types.h.

References L4_CAP_SHIFT.

12.13.3.2 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 392 of file types.h.

12.13.3.3 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 396 of file types.h.

12.14 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.

12.14.1 Detailed Description

Capability allocator C interface.

12.14.2 Function Documentation

12.14.2.1 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

12.15 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.

12.15.1 Detailed Description

12.15.2 Function Documentation

12.15.2.1 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.

12.15.2.2 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.

12.15.2.3 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.

12.15.2.4 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.

12.15.2.5 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.

12.15.2.6 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.

12.15.2.7 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.

12.16 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.

12.16.1 Detailed Description

12.16.2 Function Documentation

12.16.2.1 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.

12.17 Console API

Console interface.

Collaboration diagram for Console API:

Data Structures

- class L4Re::Console

  Console class.

12.17.1 Detailed Description

Console interface.

12.18 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.

12.18.1 Detailed Description

12.18.2 Function Documentation

12.18.2.1 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.

12.18.2.2 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.

12.18.2.3 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.

12.18.2.4 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.

12.18.2.5 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.

12.18.2.6 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.

12.18.2.7 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.

12.19 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.

12.19.1 Detailed Description

12.19.2 Function Documentation

12.19.2.1 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.

12.19.2.2 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.

12.19.2.3 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.

12.19.2.4 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.

12.19.2.5 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.

12.19.2.6 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.

12.19.2.7 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.

12.20 DMA API for L4Re

12.21 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, l4_addr_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.

12.21.1 Detailed Description

DMA Space C interface.

12.21.2 Typedef Documentation

12.21.2.1 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.

12.21.3 Function Documentation

12.21.3.1 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 #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 #Space_attrib.

12.21.3.2 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}

12.21.3.3 l4re_dma_space_map()

```
long l4re_dma_space_map (
            l4re_dma_space_t dma,
            l4re_ds_t src,
            l4_addr_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 of the region to be mapped to for DMA, after successful mapping the size returned is the size mapped for DMA as 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.

12.21.3.4 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.

12.22 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 { , L4RE_DS_MAP_NORMAL = 0x00, L4RE_DS_MAP_CACHEABLE = L4RE_DS_MAP_NORMAL, L4RE_↩ DS_MAP_BUFFERABLE = 0x10, L4RE_DS_MAP_UNCACHEABLE = 0x20, L4RE_DS_MAP_CACHING_MASK = 0x30, L4RE_DS_MAP_CACHING_SHIFT = 4 }

  Flags to specify the memory mapping type of a request.

Functions

- long l4re_ds_clear (const l4re_ds_t ds, l4_addr_t offset, unsigned long size) L4_NOTHROW
- long l4re_ds_allocate (const l4re_ds_t ds, l4_addr_t offset, l4_size_t size) L4_NOTHROW
- int l4re_ds_copy_in (const l4re_ds_t ds, l4_addr_t dst_offs, const l4re_ds_t src, l4_addr_t src_offs, unsigned long size) L4_NOTHROW
- unsigned long l4re_ds_size (const l4re_ds_t ds) L4_NOTHROW
- long l4re_ds_flags (const l4re_ds_t ds) L4_NOTHROW
- int l4re_ds_info (const l4re_ds_t ds, l4re_ds_stats_t *stats) L4_NOTHROW
- int l4re_ds_phys (const l4re_ds_t ds, l4_addr_t offset, l4_addr_t *phys_addr, l4_size_t *phys_size) L4_N↩ OTHROW

  Return physical address.

12.22.1 Detailed Description

Dataspace C interface.

12.22.2 Enumeration Type Documentation

12.22.2.1 l4re_ds_map_flags

```
enum l4re_ds_map_flags
```

Flags to specify the memory mapping type of a request.

Enumerator

L4RE_DS_MAP_NORMAL	request normal memory mapping
L4RE_DS_MAP_CACHEABLE	request normal memory mapping
L4RE_DS_MAP_BUFFERABLE	request bufferable (write buffered) mappings
L4RE_DS_MAP_UNCACHEABLE	request uncacheable memory mappings
L4RE_DS_MAP_CACHING_MASK	mask for caching flags
L4RE_DS_MAP_CACHING_SHIFT	shift value for caching flags

Definition at line 54 of file dataspace.h.

12.22.3 Function Documentation

12.22.3.1 l4re_ds_allocate()

```
long l4re_ds_allocate (
            const l4re_ds_t ds,
            l4_addr_t offset,
            l4_size_t size )
```

Returns

0 on success, <0 on errors

See also

L4Re::Dataspace::allocate

12.22.3.2 l4re_ds_clear()

```
long l4re_ds_clear (
            const l4re_ds_t ds,
            l4_addr_t offset,
            unsigned long size )
```

Returns

0 on success, <0 on errors

See also

L4Re::Dataspace::clear

12.22.3.3 l4re_ds_copy_in()

```
int l4re_ds_copy_in (
            const l4re_ds_t ds,
            l4_addr_t dst_offs,
            const l4re_ds_t src,
            l4_addr_t src_offs,
            unsigned long size )
```

Returns

0 on success, <0 on errors

See also

L4Re::Dataspace::copy_in

12.22.3.4 l4re_ds_flags()

```
long l4re_ds_flags (
            const l4re_ds_t ds )
```

See also

L4Re::Dataspace::flags

12.22.3.5 l4re_ds_info()

```
int l4re_ds_info (
            const l4re_ds_t ds,
            l4re_ds_stats_t * stats )
```

See also

L4Re::Dataspace::info

12.22.3.6 l4re_ds_phys()

```
int l4re_ds_phys (
            const l4re_ds_t ds,
            l4_addr_t offset,
            l4_addr_t * phys_addr,
            l4_size_t * phys_size )
```

Return physical address.

Parameters

ds	Dataspace
offset	Offset in bytes in dataspace

Return values

phys_addr	Physical address
phys_size	Size of physically contiguous region starting from *phys_addr* (in bytes).

Returns

0 for success, <0 on error

The function returns the physical address of an offset in a dataspace. Use multiple calls of the function to get all physical regions in case of physically non-contiguous dataspaces.

See also

L4Re::Dataspace::phys

12.22.3.7 l4re_ds_size()

```
unsigned long l4re_ds_size (
            const l4re_ds_t ds )
```

Returns

Size of the dataspace in bytes.

See also

L4Re::Dataspace::size

12.23 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.

12.23.1 Detailed Description

12.23.2 Function Documentation

12.23.2.1 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

12.24 Debugging API

Debugging Interface.

Collaboration diagram for Debugging API:

Data Structures

- class L4Re::Debug_obj

  Debug interface.

12.24.1 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.

12.25 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.

12.25.1 Detailed Description

12.25.2 Enumeration Type Documentation

12.25.2.1 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.

12.25.3 Function Documentation

12.25.3.1 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

12.25.3.2 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

12.25.3.3 libedid_dump()

```
void libedid_dump (
            const unsigned char * edid )
```

Dump raw EDID data to stdout.

Parameters

edid	Pointer to a 128byte EDID block

12.25.3.4 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

12.25.3.5 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

12.25.3.6 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.

12.25.3.7 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.

12.25.3.8 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

12.25.3.9 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

12.26 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 - figure 1-9, page 1-9.
- 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_Sym

  ELF32 symbol table entry.
- struct Elf64_Sym

  ELF64 symbol table entry.

Macros

- #define EI_NIDENT 16

  number of characters
- #define EI_CLASS 4

  ELF class byte index.
- #define EI_CLASS 4

  ELF class byte index.
- #define ELFCLASSNONE 0

  Invalid ELF class.
- #define ELFCLASSNONE 0

  Invalid ELF class.
- #define ELFCLASS32 1

  32-bit objects
- #define ELFCLASS64 2

  64-bit objects
- #define ELFCLASSNUM 3

  Mask for 32-bit or 64-bit class.
- #define EI_DATA 5

  Data encoding byte index.
- #define EI_DATA 5

  Data encoding byte index.
- #define ELFDATANONE 0

  Invalid data encoding.
- #define ELFDATANONE 0

  Invalid data encoding.
- #define ELFDATA2LSB 1

  2's complement, little endian
- #define ELFDATA2LSB 1

  2's complement, little endian
- #define ELFDATA2MSB 2

  2's complement, big endian
- #define ELFDATA2MSB 2

  2's complement, big endian
- #define EI_VERSION 6

  File version byte index.
- #define EI_VERSION 6

  File version byte index.
- #define EI_OSABI 7

  OS ABI identification.
- #define EI_OSABI 7

  OS ABI identification.
- #define ELFOSABI_NONE 0

  UNIX System V ABI.
- #define ELFOSABI_SYSV 0

  Alias.
- #define ELFOSABI_SYSV 0

  Alias.
- #define ELFOSABI_HPUX 1

  HP-UX.
- #define ELFOSABI_HPUX 1

HP-UX.

- #define ELFOSABI_NETBSD 2

  NetBSD.
- #define ELFOSABI_LINUX 3

  Linux.
- #define ELFOSABI_SOLARIS 6

  Sun Solaris.
- #define ELFOSABI_AIX 7

  IBM AIX.
- #define ELFOSABI_IRIX 8

  SGI Irix.
- #define ELFOSABI_FREEBSD 9

  FreeBSD.
- #define ELFOSABI_TRU64 10

  Compaq TRU64 UNIX.
- #define ELFOSABI_MODESTO 11

  Novell Modesto.
- #define ELFOSABI_OPENBSD 12

  OpenBSD.
- #define ELFOSABI_ARM 97

  ARM.
- #define ELFOSABI_STANDALONE 255

  Standalone (embedded) application.
- #define ELFOSABI_STANDALONE 255

  Standalone (embedded) application.
- #define EI_ABIVERSION 8

  ABI version.
- #define EI_ABIVERSION 8

  ABI version.
- #define EI_PAD 9

  Byte index of padding bytes.
- #define EI_PAD 9

  Byte index of padding bytes.
- #define ET_NONE 0

  no file type
- #define ET_REL 1

  relocatable file
- #define ET_EXEC 2

  executable file
- #define ET_DYN 3

  shared object file
- #define ET_CORE 4

  core file
- #define ET_LOPROC 0xff00

  processor-specific
- #define ET_HIPROC 0xffff

  processor-specific
- #define EM_NONE 0

  no machine
- #define EM_M32 1

  AT&T WE 32100.

- #define EM_SPARC 2

  SPARC.
- #define EM_386 3

  Intel 80386.
- #define EM_68K 4

  Motorola 68000.
- #define EM_88K 5

  Motorola 88000.
- #define EM_860 7

  Intel 80860.
- #define EM_MIPS 8

  MIPS RS3000 big-endian.
- #define EM_MIPS_RS4_BE 10

  MIPS RS4000 big-endian.
- #define EM_SPARC64 11

  SPARC 64-bit.
- #define EM_PARISC 15

  HP PA-RISC.
- #define EM_VPP500 17

  Fujitsu VPP500.
- #define EM_SPARC32PLUS 18

  Sun's V8plus.
- #define EM_960 19

  Intel 80960.
- #define EM_PPC 20

  PowerPC.
- #define EM_V800 36

  NEC V800.
- #define EM_FR20 37

  Fujitsu FR20.
- #define EM_RH32 38

  TRW RH-32.
- #define EM_RCE 39

  Motorola RCE.
- #define EM_ARM 40

  Advanced RISC Machines ARM.
- #define EM_ALPHA 41

  Digital Alpha.
- #define EM_SH 42

  Hitachi SuperH.
- #define EM_SPARCV9 43

  SPARC v9 64-bit.
- #define EM_TRICORE 44

  Siemens Tricore embedded processor.
- #define EM_ARC 45

  Argonaut RISC Core, Argonaut Techn Inc.
- #define EM_H8_300 46

  Hitachi H8/300.
- #define EM_H8_300H 47

  Hitachi H8/300H.
- #define EM_H8S 48

Hitachi H8/S.

- #define EM_H8_500 49

  Hitachi H8/500.
- #define EM_IA_64 50

  HP/Intel IA-64.
- #define EM_MIPS_X 51

  Stanford MIPS-X.
- #define EM_COLDFIRE 52

  Motorola Coldfire.
- #define EM_68HC12 53

  Motorola M68HC12.
- #define EM_X86_64 62

  Advanced Micro Devices x86-64.
- #define EM_PDSP 63

  Sony DSP Processor.
- #define EM_FX66 66

  Siemens FX66 microcontroller.
- #define EM_ST9PLUS 67

  STMicroelectronics ST9+ 8/16 mc.
- #define EM_ST7 68

  STmicroelectronics ST7 8 bit mc.
- #define EM_68HC16 69

  Motorola MC68HC16 microcontroller.
- #define EM_68HC11 70

  Motorola MC68HC11 microcontroller.
- #define EM_68HC08 71

  Motorola MC68HC08 microcontroller.
- #define EM_68HC05 72

  Motorola MC68HC05 microcontroller.
- #define EM_SVX 73

  Silicon Graphics SVx.
- #define EM_ST19 74

  STMicroelectronics ST19 8 bit mc.
- #define EM_VAX 75

  Digital VAX.
- #define EM_CRIS 76

  Axis Communications 32-bit embedded processor.
- #define EM_JAVELIN 77

  Infineon Technologies 32-bit embedded processor.
- #define EM_FIREPATH 78

  Element 14 64-bit DSP Processor.
- #define EM_ZSP 79

  LSI Logic 16-bit DSP Processor.
- #define EM_MMIX 80

  Donald Knuth's educational 64-bit processor.
- #define EM_HUANY 81

  Harvard University machine-independent object files.
- #define EM_PRISM 82

  SiTera Prism.
- #define EM_AVR 83

  Atmel AVR 8-bit microcontroller.

- #define EM_FR30 84

  Fujitsu FR30.
- #define EM_D10V 85

  Mitsubishi D10V.
- #define EM_D30V 86

  Mitsubishi D30V.
- #define EM_V850 87

  NEC v850.
- #define EM_M32R 88

  Mitsubishi M32R.
- #define EM_MN10300 89

  Matsushita MN10300.
- #define EM_MN10200 90

  Matsushita MN10200.
- #define EM_PJ 91

  picoJava
- #define EM_OPENRISC 92

  OpenRISC 32-bit embedded processor.
- #define EM_ARC_A5 93

  ARC Cores Tangent-A5.
- #define EM_XTENSA 94

  Tensilica Xtensa Architecture.
- #define EM_ALTERA_NIOS2 113

  Altera Nios II.
- #define EM_AARCH64 183

  ARM AARCH64.
- #define EM_TILEPRO 188

  Tilera TILEPro.
- #define EM_MICROBLAZE 189

  Xilinx MicroBlaze.
- #define EM_TILEGX 191

  Tilera TILE-Gx.
- #define EV_NONE 0

  Invalid version.
- #define EV_CURRENT 1

  Current version.
- #define EI_MAG0 0

  file id
- #define EI_MAG1 1

  file id
- #define EI_MAG2 2

  file id
- #define EI_MAG3 3

  file id
- #define ELFMAG0 0x7f

  e_ident[EI_MAG0]
- #define ELFMAG1 'E'

  e_ident[EI_MAG1]
- #define ELFMAG2 'L'

  e_ident[EI_MAG2]
- #define ELFMAG3 'F'

e_ident[EI_MAG3]

- #define ELFCLASSS32 1

  32-bit object
- #define ELFCLASSS64 2

  64-bit object
- #define SHN_UNDEF 0

  undefined section header entry
- #define SHN_LORESERVE 0xff00

  lower bound of reserved indexes
- #define SHN_LOPROC 0xff00

  lower bound of proc spec entr
- #define SHN_HIPROC 0xff1f

  upper bound of proc spec entr
- #define SHN_ABS 0xfff1

  absolute values for ref
- #define SHN_COMMON 0xfff2

  common symbols
- #define SHN_HIRESERVE 0xffff

  upper bound of reserved indexes
- #define SHT_INIT_ARRAY 14

  Array of constructors.
- #define SHT_FINI_ARRAY 15

  Array of destructors.
- #define SHT_PREINIT_ARRAY 16

  Array of pre-constructors.
- #define SHT_GROUP 17

  Section group.
- #define SHT_SYMTAB_SHNDX 18

  Extended section indeces.
- #define SHT_NUM 19

  Number of defined types.
- #define SHF_WRITE 0x1

  writeable during execution
- #define SHF_ALLOC 0x2

  section occupies virt memory
- #define SHF_EXECINSTR 0x4

  code section
- #define SHF_MERGE 0x10

  Might be merged.
- #define SHF_STRINGS 0x20

  Contains nul-terminated strings.
- #define SHF_INFO_LINK 0x40

  'sh_info' contains SHT index
- #define SHF_LINK_ORDER 0x80

  Preserve order after combining.
- #define SHF_OS_NONCONFORMING 0x100

  Non-standard OS specific handling required.
- #define SHF_GROUP 0x200

  Section is member of a group.
- #define SHF_TLS 0x400

  Section hold thread-local data.

- #define SHF_MASKOS 0x0ff00000

  OS-specific.
- #define SHF_MASKPROC 0xf0000000

  proc spec mask
- #define PT_NULL 0

  array is unused
- #define PT_LOAD 1

  loadable
- #define PT_DYNAMIC 2

  dynamic linking information
- #define PT_INTERP 3

  path to interpreter
- #define PT_NOTE 4

  auxiliary information
- #define PT_SHLIB 5

  reserved
- #define PT_PHDR 6

  location of the pht itself
- #define PT_TLS 7

  Thread-local storage segment.
- #define PT_NUM 8

  Number of defined types.
- #define PT_LOOS 0x60000000

  os spec.
- #define PT_HIOS 0x6fffffff

  os spec.
- #define PT_LOPROC 0x70000000

  processor spec.
- #define PT_HIPROC 0x7fffffff

  processor spec.
- #define PT_GNU_EH_FRAME (PT_LOOS + 0x474e550)

  EH frame information.
- #define PT_GNU_STACK (PT_LOOS + 0x474e551)

  Flags for stack.
- #define PT_GNU_RELRO (PT_LOOS + 0x474e552)

  Read only after reloc.
- #define PT_L4_STACK (PT_LOOS + 0x12)

  Address of the stack.
- #define PT_L4_KIP (PT_LOOS + 0x13)

  Address of the KIP.
- #define PT_L4_AUX (PT_LOOS + 0x14)

  Address of the AUX strcutures.
- #define NT_PRSTATUS 1

  Contains copy of prstatus struct.
- #define NT_FPREGSET 2

  Contains copy of fpregset struct.
- #define NT_PRPSINFO 3

  Contains copy of prpsinfo struct.
- #define NT_PRXREG 4

  Contains copy of prxregset struct.
- #define NT_TASKSTRUCT 4

Contains copy of task structure.

- #define NT_PLATFORM 5

  String from sysinfo(SI_PLATFORM)
- #define NT_AUXV 6

  Contains copy of auxv array.
- #define NT_GWINDOWS 7

  Contains copy of gwindows struct.
- #define NT_ASRS 8

  Contains copy of asrset struct.
- #define NT_PSTATUS 10

  Contains copy of pstatus struct.
- #define NT_PSINFO 13

  Contains copy of psinfo struct.
- #define NT_PRCRED 14

  Contains copy of prcred struct.
- #define NT_UTSNAME 15

  Contains copy of utsname struct.
- #define NT_LWPSTATUS 16

  Contains copy of lwpstatus struct.
- #define NT_LWPSINFO 17

  Contains copy of lwpinfo struct.
- #define NT_PRFPXREG 20

  Contains copy of fprxregset struct.
- #define NT_VERSION 1

  Contains a version string.
- #define DT_NULL 0

  Dynamic Array Tags, d_tag - figure 2-10, page 2-12.
- #define DT_NEEDED 1

  name of a needed library
- #define DT_PLTRELSZ 2

  total size of relocation entry
- #define DT_PLTGOT 3

  address assoc with prog link table
- #define DT_HASH 4

  address of symbol hash table
- #define DT_STRTAB 5

  address of string table
- #define DT_SYMTAB 6

  address of symbol table
- #define DT_RELA 7

  address of relocation table
- #define DT_RELASZ 8

  total size of relocation table
- #define DT_RELAENT 9

  size of DT_RELA relocation entry
- #define DT_STRSZ 10

  size of the string table
- #define DT_SYMENT 11

  size of a symbol table entry
- #define DT_INIT 12

  address of initialization function

- #define DT_FINI 13

  address of termination function
- #define DT_SONAME 14

  name of the shared object
- #define DT_RPATH 15

  search library path
- #define DT_SYMBOLIC 16

  alter symbol resolution algorithm
- #define DT_REL 17

  address of relocation table
- #define DT_RELSZ 18

  total size of DT_REL relocation table
- #define DT_RELENT 19

  size of the DT_REL relocation entry
- #define DT_PTRREL 20

  type of relocation entry
- #define DT_DEBUG 21

  for debugging purposes
- #define DT_TEXTREL 22

  at least on entry changes r/o section
- #define DT_JMPREL 23

  address of relocation entries
- #define DT_BIND_NOW 24

  Process relocations of object.
- #define DT_INIT_ARRAY 25

  Array with addresses of init fct.
- #define DT_FINI_ARRAY 26

  Array with addresses of fini fct.
- #define DT_INIT_ARRAYSZ 27

  Size in bytes of DT_INIT_ARRAY.
- #define DT_FINI_ARRAYSZ 28

  Size in bytes of DT_FINI_ARRAY.
- #define DT_RUNPATH 29

  Library search path.
- #define DT_FLAGS 30

  Flags for the object being loaded.
- #define DT_ENCODING 32

  Start of encoded range.
- #define DT_PREINIT_ARRAY 32

  Array with addresses of preinit fct.
- #define DT_PREINIT_ARRAYSZ 33

  size in bytes of DT_PREINIT_ARRAY
- #define DT_NUM 34

  Number used.
- #define DT_LOOS 0x6000000d

  Start of OS-specific.
- #define DT_HIOS 0x6ffff000

  End of OS-specific.
- #define DT_LOPROC 0x70000000

  processor spec.
- #define DT_HIPROC 0x7fffffff

processor spec.

- #define DF_ORIGIN 0x00000001

  Object may use DF_ORIGIN.
- #define DF_SYMBOLIC 0x00000002

  Symbol resolutions starts here.
- #define DF_TEXTREL 0x00000004

  Object contains text relocations.
- #define DF_BIND_NOW 0x00000008

  No lazy binding for this object.
- #define DF_STATIC_TLS 0x00000010

  Module uses the static TLS model.
- #define DF_1_NOW 0x00000001

  Set RTLD_NOW for this object.
- #define DF_1_GLOBAL 0x00000002

  Set RTLD_GLOBAL for this object.
- #define DF_1_GROUP 0x00000004

  Set RTLD_GROUP for this object.
- #define DF_1_NODELETE 0x00000008

  Set RTLD_NODELETE for this object.
- #define DF_1_LOADFLTR 0x00000010

  Trigger filtee loading at runtime.
- #define DF_1_INITFIRST 0x00000020

  Set RTLD_INITFIRST for this object.
- #define DF_1_NOOPEN 0x00000040

  Set RTLD_NOOPEN for this object.
- #define DF_1_ORIGIN 0x00000080

  $ORIGIN must be handled.
- #define DF_1_DIRECT 0x00000100

  Direct binding enabled.
- #define DF_1_INTERPOSE 0x00000400

  Object is used to interpose.
- #define DF_1_NODEFLIB 0x00000800

  Ignore default lib search path.
- #define DF_1_NODUMP 0x00001000

  Object can't be dldump'ed.
- #define DF_1_CONFALT 0x00002000

  Configuration alternative created.
- #define DF_1_ENDFILTEE 0x00004000

  Filtee terminates filters search.
- #define DF_1_DISPRELDNE 0x00008000

  Disp reloc applied at build time.
- #define DF_1_DISPRELPND 0x00010000

  Disp reloc applied at run-time.
- #define DF_P1_LAZYLOAD 0x00000001

  Lazyload following object.
- #define DF_P1_GROUPPERM 0x00000002

  Symbols from next object are not generally available.
- #define R_386_NONE 0

  none
- #define R_386_32 1

  S + A.

- #define R_386_PC32 2

  S + A - P.
- #define R_386_GOT32 3

  G + A - P.
- #define R_386_PLT32 4

  L + A - P.
- #define R_386_COPY 5

  none
- #define R_386_GLOB_DAT 6

  S.
- #define R_386_JMP_SLOT 7

  S.
- #define R_386_RELATIVE 8

  B + A.
- #define R_386_GOTOFF 9

  S + A - GOT.
- #define R_386_GOTPC 10

  GOT + A - P.
- #define STB_LOCAL 0

  not visible outside object file
- #define STB_GLOBAL 1

  visible to all objects beeing combined
- #define STB_WEAK 2

  resemble global symbols
- #define STB_LOOS 10

  os specific
- #define STB_HIOS 12

  os specific
- #define STB_LOPROC 13

  proc specific
- #define STB_HIPROC 15

  proc specific
- #define STT_NOTYPE 0

  symbol's type not specified
- #define STT_OBJECT 1

  associated with a data object
- #define STT_FUNC 2

  associated with a function or other code
- #define STT_SECTION 3

  associated with a section
- #define STT_FILE 4

  source file name associated with object
- #define STT_LOOS 10

  os specific
- #define STT_HIOS 12

  os specific
- #define STT_LOPROC 13

  proc specific
- #define STT_HIPROC 15

  proc specific

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

12.26.1 Detailed Description

Functions and types related to ELF binaries.

12.26.2 Macro Definition Documentation

12.26.2.1 DF_1_CONFALT

```
#define DF_1_CONFALT 0x00002000
```

Configuration alternative created.

Definition at line 581 of file elf.h.

12.26.2.2 DF_1_DIRECT

```
#define DF_1_DIRECT 0x00000100
```

Direct binding enabled.

Definition at line 576 of file elf.h.

12.26.2.3 DF_1_DISPRELDNE

```
#define DF_1_DISPRELDNE 0x00008000
```

Disp reloc applied at build time.

Definition at line 583 of file elf.h.

12.26.2.4 DF_1_DISPRELPND

```
#define DF_1_DISPRELPND 0x00010000
```

Disp reloc applied at run-time.

Definition at line 584 of file elf.h.

12.26.2.5 DF_1_ENDFILTEE

```
#define DF_1_ENDFILTEE 0x00004000
```

Filtee terminates filters search.

Definition at line 582 of file elf.h.

12.26.2.6 DF_1_GLOBAL

```
#define DF_1_GLOBAL 0x00000002
```

Set RTLD_GLOBAL for this object.

Definition at line 569 of file elf.h.

12.26.2.7 DF_1_GROUP

```
#define DF_1_GROUP 0x00000004
```

Set RTLD_GROUP for this object.

Definition at line 570 of file elf.h.

12.26.2.8 DF_1_INTERPOSE

```
#define DF_1_INTERPOSE 0x00000400
```

Object is used to interpose.

Definition at line 578 of file elf.h.

12.26.2.9 DF_1_LOADFLTR

```
#define DF_1_LOADFLTR 0x00000010
```

Trigger filtee loading at runtime.

Definition at line 572 of file elf.h.

12.26.2.10 DF_1_NODEFLIB

```
#define DF_1_NODEFLIB 0x00000800
```

Ignore default lib search path.

Definition at line 579 of file elf.h.

12.26.2.11 DF_1_NODELETE

```
#define DF_1_NODELETE 0x00000008
```

Set RTLD_NODELETE for this object.

Definition at line 571 of file elf.h.

12.26.2.12 DF_1_NODUMP

```
#define DF_1_NODUMP 0x00001000
```

Object can't be dldump'ed.

Definition at line 580 of file elf.h.

12.26.2.13 DF_1_NOOPEN

```
#define DF_1_NOOPEN 0x00000040
```

Set RTLD_NOOPEN for this object.

Definition at line 574 of file elf.h.

12.26.2.14 DF_1_NOW

```
#define DF_1_NOW 0x00000001
```

Set RTLD_NOW for this object.

Definition at line 568 of file elf.h.

12.26.2.15 DF_1_ORIGIN

```
#define DF_1_ORIGIN 0x00000080
```

$ORIGIN must be handled.

Definition at line 575 of file elf.h.

12.26.2.16 DF_P1_GROUPPERM

```
#define DF_P1_GROUPPERM 0x00000002
```

Symbols from next object are not generally available.

Definition at line 592 of file elf.h.

12.26.2.17 DF_P1_LAZYLOAD

```
#define DF_P1_LAZYLOAD 0x00000001
```

Lazyload following object.

Definition at line 591 of file elf.h.

12.26.2.18 DT_HIPROC

```
#define DT_HIPROC 0x7fffffff
```

processor spec.

Definition at line 557 of file elf.h.

12.26.2.19 DT_LOPROC

```
#define DT_LOPROC 0x70000000
```

processor spec.

Definition at line 556 of file elf.h.

12.26.2.20 DT_NULL

```
#define DT_NULL 0
```

Dynamic Array Tags, d_tag - figure 2-10, page 2-12.

end of _DYNAMIC array

Definition at line 519 of file elf.h.

12.26.2.21 EI_CLASS [1/2]

```
#define EI_CLASS 4
```

ELF class byte index.

file class

Definition at line 294 of file elf.h.

12.26.2.22 EI_CLASS [2/2]

```
#define EI_CLASS 4
```

ELF class byte index.

file class

Definition at line 294 of file elf.h.

12.26.2.23 EI_DATA [1/2]

```
#define EI_DATA 5
```

Data encoding byte index.

data encoding

Definition at line 295 of file elf.h.

12.26.2.24 EI_DATA [2/2]

```
#define EI_DATA 5
```

Data encoding byte index.

data encoding

Definition at line 295 of file elf.h.

12.26.2.25 EI_OSABI [1/2]

```
#define EI_OSABI 7
```

OS ABI identification.

Operating system / ABI identification.

Definition at line 297 of file elf.h.

12.26.2.26 EI_OSABI [2/2]

```
#define EI_OSABI 7
```

OS ABI identification.

Operating system / ABI identification.

Definition at line 297 of file elf.h.

12.26.2.27 EI_PAD [1/2]

```
#define EI_PAD 9
```

Byte index of padding bytes.

start of padding bytes

Definition at line 299 of file elf.h.

12.26.2.28 EI_PAD [2/2]

```
#define EI_PAD 9
```

Byte index of padding bytes.

start of padding bytes

Definition at line 299 of file elf.h.

12.26.2.29 EI_VERSION [1/2]

```
#define EI_VERSION 6
```

File version byte index.

file version

Value must be EV_CURRENT

Definition at line 296 of file elf.h.

12.26.2.30 EI_VERSION [2/2]

```
#define EI_VERSION 6
```

File version byte index.

file version

Value must be EV_CURRENT

Definition at line 296 of file elf.h.

12.26.2.31 ELFCLASSNONE [1/2]

```
#define ELFCLASSNONE 0
```

Invalid ELF class.

Invalid class.

Definition at line 310 of file elf.h.

12.26.2.32 ELFCLASSNONE [2/2]

```
#define ELFCLASSNONE 0
```

Invalid ELF class.

Invalid class.

Definition at line 310 of file elf.h.

12.26.2.33 ELFDATA2LSB [1/2]

```
#define ELFDATA2LSB 1
```

2's complement, little endian

0x01020304 => [0x04|0x03|0x02|0x01]

Definition at line 317 of file elf.h.

12.26.2.34 ELFDATA2LSB [2/2]

```
#define ELFDATA2LSB 1
```

2's complement, little endian

0x01020304 => [0x04|0x03|0x02|0x01]

Definition at line 317 of file elf.h.

12.26.2.35 ELFDATA2MSB [1/2]

```
#define ELFDATA2MSB 2
```

2's complement, big endian

0x01020304 => [0x01|0x02|0x03|0x04]

Definition at line 318 of file elf.h.

12.26.2.36 ELFDATA2MSB [2/2]

```
#define ELFDATA2MSB 2
```

2's complement, big endian

0x01020304 => [0x01|0x02|0x03|0x04]

Definition at line 318 of file elf.h.

12.26.2.37 ELFDATANONE [1/2]

```
#define ELFDATANONE 0
```

Invalid data encoding.

invalid data encoding

Definition at line 316 of file elf.h.

12.26.2.38 ELFDATANONE [2/2]

```
#define ELFDATANONE 0
```

Invalid data encoding.

invalid data encoding

Definition at line 316 of file elf.h.

12.26.2.39 ELFOSABI_AIX

```
#define ELFOSABI_AIX 7
```

IBM AIX.

Definition at line 181 of file elf.h.

12.26.2.40 ELFOSABI_FREEBSD

```
#define ELFOSABI_FREEBSD 9
```

FreeBSD.

Definition at line 183 of file elf.h.

12.26.2.41 ELFOSABI_HPUX [1/2]

```
#define ELFOSABI_HPUX 1
```

HP-UX.

HP-UX operating system.

Definition at line 323 of file elf.h.

12.26.2.42 ELFOSABI_HPUX [2/2]

```
#define ELFOSABI_HPUX 1
```

HP-UX.

HP-UX operating system.

Definition at line 323 of file elf.h.

12.26.2.43 ELFOSABI_IRIX

```
#define ELFOSABI_IRIX 8
```

SGI Irix.

Definition at line 182 of file elf.h.

12.26.2.44 ELFOSABI_LINUX

```
#define ELFOSABI_LINUX 3
```

Linux.

Definition at line 179 of file elf.h.

12.26.2.45 ELFOSABI_MODESTO

```
#define ELFOSABI_MODESTO 11
```

Novell Modesto.

Definition at line 185 of file elf.h.

12.26.2.46 ELFOSABI_NETBSD

```
#define ELFOSABI_NETBSD 2
```

NetBSD.

Definition at line 178 of file elf.h.

12.26.2.47 ELFOSABI_OPENBSD

```
#define ELFOSABI_OPENBSD 12
```

OpenBSD.

Definition at line 186 of file elf.h.

12.26.2.48 ELFOSABI_SOLARIS

```
#define ELFOSABI_SOLARIS 6
```

Sun Solaris.

Definition at line 180 of file elf.h.

12.26.2.49 ELFOSABI_SYSV [1/2]

```
#define ELFOSABI_SYSV 0
```

Alias.

UNIX System V ABI (this specification)

Definition at line 322 of file elf.h.

12.26.2.50 ELFOSABI_SYSV [2/2]

```
#define ELFOSABI_SYSV 0
```

Alias.

UNIX System V ABI (this specification)

Definition at line 322 of file elf.h.

12.26.2.51 ELFOSABI_TRU64

```
#define ELFOSABI_TRU64 10
```

Compaq TRU64 UNIX.

Definition at line 184 of file elf.h.

12.26.2.52 EM_ARC

```
#define EM_ARC 45
```

Argonaut RISC Core, Argonaut Techn Inc.

Definition at line 230 of file elf.h.

12.26.2.53 NT_VERSION

```
#define NT_VERSION 1
```

Contains a version string.

Definition at line 495 of file elf.h.

12.26.2.54 PT_GNU_EH_FRAME

```
#define PT_GNU_EH_FRAME (PT_LOOS + 0x474e550)
```

EH frame information.

Definition at line 458 of file elf.h.

12.26.2.55 PT_GNU_RELRO

```
#define PT_GNU_RELRO (PT_LOOS + 0x474e552)
```

Read only after reloc.

Definition at line 460 of file elf.h.

12.26.2.56 PT_GNU_STACK

```
#define PT_GNU_STACK (PT_LOOS + 0x474e551)
```

Flags for stack.

Definition at line 459 of file elf.h.

12.26.2.57 PT_HIOS

```
#define PT_HIOS 0x6fffffff
```

os spec.

Definition at line 454 of file elf.h.

12.26.2.58 PT_HIPROC

```
#define PT_HIPROC 0x7fffffff
```

processor spec.

Definition at line 456 of file elf.h.

12.26.2.59 PT_L4_AUX

```
#define PT_L4_AUX (PT_LOOS + 0x14)
```

Address of the AUX strcutures.

Definition at line 464 of file elf.h.

12.26.2.60 PT_L4_KIP

```
#define PT_L4_KIP (PT_LOOS + 0x13)
```

Address of the KIP.

Definition at line 463 of file elf.h.

12.26.2.61 PT_L4_STACK

```
#define PT_L4_STACK (PT_LOOS + 0x12)
```

Address of the stack.

Definition at line 462 of file elf.h.

12.26.2.62 PT_LOOS

```
#define PT_LOOS 0x60000000
```

os spec.

Definition at line 453 of file elf.h.

12.26.2.63 PT_LOPROC

```
#define PT_LOPROC 0x70000000
```

processor spec.

Definition at line 455 of file elf.h.

12.26.2.64 SHF_GROUP

```
#define SHF_GROUP 0x200
```

Section is member of a group.

Definition at line 408 of file elf.h.

12.26.2.65 SHF_MASKOS

```
#define SHF_MASKOS 0x0ff00000
```

OS-specific.

Definition at line 410 of file elf.h.

12.26.2.66 SHF_TLS

```
#define SHF_TLS 0x400
```

Section hold thread-local data.

Definition at line 409 of file elf.h.

12.26.2.67 SHT_NUM

```
#define SHT_NUM 19
```

Number of defined types.

Definition at line 388 of file elf.h.

12.27 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_RE↩MSGCUT = 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.
- 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.

12.27.1 Detailed Description

Error handling for L4 object invocation.

Include File

```
#include <l4/sys/ipc.h>
```

12.27.2 Enumeration Type Documentation

12.27.2.1 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.
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.

12.27.3 Function Documentation

12.27.3.1 l4_error()

```
long l4_error (
            l4_msgtag_t tag )  [inline]
```

Return error code of a system call return message tag.

Parameters

tag	System call return message type

Returns

0 for no error, error number in case of error

Examples:

examples/libs/l4re/streammap/client.cc, examples/sys/aliens/main.c, examples/sys/isr/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 517 of file ipc.h.

Referenced by L4Re::Mem_alloc::alloc(), L4::cap_dynamic_cast(), 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:

12.27.3.2 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 500 of file ipc.h.

References l4_msgtag_has_error().

Referenced by l4util_ioport_map(), and L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process().

Here is the call graph for this function:

Here is the caller graph for this function:

12.27.3.3 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 529 of file ipc.h.

12.27.3.4 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 526 of file ipc.h.

12.27.3.5 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 523 of file ipc.h.

12.28 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.

12.28.1 Detailed Description

Common error codes.

Include File

```
#include <l4/sys/err.h>
```

12.28.2 Enumeration Type Documentation

12.28.2.1 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.

12.29 Event API

Event interface.

Collaboration diagram for Event API:

Data Structures

- class L4Re::Event

  Event class.
- class L4Re::Event_buffer_t< PAYLOAD >

  Event buffer class.

12.29.1 Detailed Description

Event interface.

12.30 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_N↩ OTHROW

  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.

12.30.1 Detailed Description

Event C interface.

12.30.2 Function Documentation

12.30.2.1 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

12.30.2.2 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

12.30.2.3 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

12.30.2.4 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

12.30.2.5 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

12.31 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().

12.31.1 Detailed Description

Overly definition of the MRs for exception messages.

12.31.2 Function Documentation

12.31.2.1 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.

12.31.2.2 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 115 of file utcb.h.

References l4_utcb_exc_typeval().

Here is the call graph for this function:

12.31.2.3 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 105 of file utcb.h.

References l4_exc_regs_t::trapno.

12.31.2.4 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 90 of file utcb.h.

12.31.2.5 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 95 of file utcb.h.

References l4_exc_regs_t::pc.

12.32 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.

12.32.1 Detailed Description

extended vCPU handling functionality.

12.32.2 Function Documentation

12.32.2.1 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.

Parameters

regmgr	Region manager to use for allocation.

Returns

0 for success, error code otherwise

12.33 Factory

C factory interface to create kernel objects.

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 const utcb_area) L4_NOTHROW

  Create a new task.
- l4_msgtag_t l4_factory_create_task_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_fpage_t const utcb_area, l4_utcb_t *utcb) 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_thread_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) 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_factory_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit, l4_utcb_t *utcb) 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_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

  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_irq_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW

  Create a new IRQ.
- 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_vm_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW

  Create a new virtual machine.

12.33.1 Detailed Description

C factory interface to create kernel objects.

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.

12.33.2 Function Documentation

12.33.2.1 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 373 of file factory.h.

References l4_factory_create_factory_u(), and l4_utcb().

Here is the call graph for this function:

12.33.2.2 l4_factory_create_factory_u()

```
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 )  [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.
	utcb	The UTCB to use for the operation.

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 307 of file factory.h.

Referenced by L4::Factory::create_factory(), and l4_factory_create_factory().

Here is the caller graph for this function:

12.33.2.3 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 (is used if `thread_cap` is valid).

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 write 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 381 of file factory.h.

References l4_factory_create_gate_u(), and l4_utcb().

Here is the call graph for this function:

12.33.2.4 l4_factory_create_gate_u()

```
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 )  [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 (is used if `thread_cap` is valid).
	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.
-L4_ENOENT	`thread_cap` is void or points to something that is not a thread.
-L4_EPERM	No write rights on `thread_cap`.

An unbound IPC gate can be bound to a thread using L4::lpc_gate::bind_thread().

See also

L4::lpc_gate

Definition at line 318 of file factory.h.

Referenced by L4::Factory::create_gate(), and l4_factory_create_gate().

Here is the caller graph for this function:

12.33.2.5 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 389 of file factory.h.

References l4_factory_create_irq_u(), and l4_utcb().

Here is the call graph for this function:

12.33.2.6 l4_factory_create_irq_u()

```
l4_msgtag_t l4_factory_create_irq_u (
            l4_cap_idx_t factory,
            l4_cap_idx_t target_cap,
            l4_utcb_t * utcb )  [inline]
```

Create a new IRQ.

Parameters

	factory	Factory to use for creation.
out	*target_cap*	The kernel stores the new IRQ's capability into this slot.
	utcb	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Irq>` as argument instead.

See also

L4::Irq

Definition at line 340 of file factory.h.

Referenced by L4::Factory::create_irq(), and l4_factory_create_irq().

Here is the caller graph for this function:

12.33.2.7 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 const 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 359 of file factory.h.

References l4_factory_create_task_u(), and l4_utcb().

Here is the call graph for this function:

12.33.2.8 l4_factory_create_task_u()

```
l4_msgtag_t l4_factory_create_task_u (
            l4_cap_idx_t factory,
            l4_cap_idx_t target_cap,
            l4_fpage_t const utcb_area,
            l4_utcb_t * utcb )  [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 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.

See also

L4::Task

Definition at line 289 of file factory.h.

Referenced by L4::Factory::create_task(), and l4_factory_create_task().

Here is the caller graph for this function:

12.33.2.9 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 366 of file factory.h.

References l4_factory_create_thread_u(), and l4_utcb().

Here is the call graph for this function:

12.33.2.10 l4_factory_create_thread_u()

```
l4_msgtag_t l4_factory_create_thread_u (
            l4_cap_idx_t factory,
            l4_cap_idx_t target_cap,
            l4_utcb_t * utcb )  [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.
	utcb	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Thread>` as argument instead.

See also

L4::Thread

Definition at line 300 of file factory.h.

Referenced by L4::Factory::create_thread(), and l4_factory_create_thread().

Here is the caller graph for this function:

12.33.2.11 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 396 of file factory.h.

References l4_factory_create_vm_u(), and l4_utcb().

Here is the call graph for this function:

12.33.2.12 l4_factory_create_vm_u()

```
l4_msgtag_t l4_factory_create_vm_u (
            l4_cap_idx_t factory,
            l4_cap_idx_t target_cap,
            l4_utcb_t * utcb )  [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.
	utcb	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Vm>` as argument instead.

See also

L4::Vm

Definition at line 347 of file factory.h.

Referenced by L4::Factory::create_vm(), and l4_factory_create_vm().

Here is the caller graph for this function:

12.34 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

Data Structures

- struct l4_tracebuffer_status_window_t

  Trace-buffer status window descriptor.
- struct l4_tracebuffer_status_t

  Trace-buffer status.

Enumerations

- enum {
  LOG_EVENT_CONTEXT_SWITCH = 0, LOG_EVENT_IPC_SHORTCUT = 1, LOG_EVENT_IRQ_RAISED = 2, LOG_EVENT_TIMER_IRQ = 3,
  LOG_EVENT_THREAD_EX_REGS = 4, LOG_EVENT_MAX_EVENTS = 16 }

  Log event types.

Functions

- l4_tracebuffer_status_t * fiasco_tbuf_get_status (void)

  Return trace-buffer status.
- l4_addr_t fiasco_tbuf_get_status_phys (void)

  Return the physical address of the trace-buffer status struct.
- 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.

12.34.1 Detailed Description

Kernel debugger extensions of the Fiasco L4 implementation.

12.34.2 Enumeration Type Documentation

12.34.2.1 anonymous enum

```
anonymous enum
```

Log event types.

Enumerator

LOG_EVENT_CONTEXT_SWITCH	Event: context switch.
LOG_EVENT_IPC_SHORTCUT	Event: IPC shortcut.
LOG_EVENT_IRQ_RAISED	Event: IRQ occurred.
LOG_EVENT_TIMER_IRQ	Event: Timer IRQ occurred.
LOG_EVENT_THREAD_EX_REGS	Event: thread_ex_regs.
LOG_EVENT_MAX_EVENTS	Maximum number of events.

Definition at line 37 of file ktrace.h.

12.34.3 Function Documentation

12.34.3.1 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 149 of file segment.h.

12.34.3.2 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.

Returns

System call error

Definition at line 52 of file segment.h.

12.34.3.3 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.

Definition at line 136 of file segment.h.

References L4_EINVAL, and L4_TASK_LDT_X86_MAX_ENTRIES.

12.34.3.4 fiasco_tbuf_get_status()

```
l4_tracebuffer_status_t* fiasco_tbuf_get_status (
            void )  [inline]
```

Return trace-buffer status.

Returns

Pointer to trace-buffer status struct.

Definition at line 35 of file __ktrace-impl.h.

12.34.3.5 fiasco_tbuf_get_status_phys()

```
l4_addr_t fiasco_tbuf_get_status_phys (
            void )  [inline]
```

Return the physical address of the trace-buffer status struct.

Returns

physical address of status struct.

Definition at line 42 of file __ktrace-impl.h.

12.34.3.6 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 48 of file __ktrace-impl.h.

References l4_error().

Here is the call graph for this function:

12.34.3.7 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 55 of file __ktrace-impl.h.

References l4_error().

Here is the call graph for this function:

12.34.3.8 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 78 of file __ktrace-impl.h.

References l4_error().

Here is the call graph for this function:

12.35 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.

12.36 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.

12.36.1 Detailed Description

Virtual hardware devices, provided by Fiasco-UX.

Include File

```
#include <l4/sys/vhw.h>
```

12.36.2 Enumeration Type Documentation

12.36.2.1 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.

12.37 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 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 }

  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

  Flex-page type.
- enum L4_fpage_control

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.

12.37.1 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.

12.37.2 Enumeration Type Documentation

12.37.2.1 anonymous enum

`anonymous enum`

Constants for flexpages.

Enumerator

L4_WHOLE_ADDRESS_SPACE	Whole address space size.

Definition at line 89 of file __l4_fpage.h.

12.37.2.2 anonymous enum

`anonymous enum`

Special constants for IO flex pages.

Enumerator

L4_WHOLE_IOADDRESS_SPACE	Whole I/O address space size.
L4_IOPORT_MAX	Maximum I/O port address.

Definition at line 281 of file __l4_fpage.h.

12.37.2.3 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
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

Enumerator

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 134 of file __l4_fpage.h.

12.37.2.4 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 262 of file __l4_fpage.h.

12.37.2.5 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)

Enumerator

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 55 of file __l4_fpage.h.

12.37.2.6 L4_fpage_rights

`enum L4_fpage_rights`

Flex-page rights.

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 107 of file __l4_fpage.h.

12.37.2.7 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 240 of file __l4_fpage.h.

12.37.3 Function Documentation

12.37.3.1 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)
rights	Access rights, see L4_fpage_rights

Returns

Memory flex page

Definition at line 633 of file __l4_fpage.h.

Referenced by L4Re::Rm::attach().

Here is the caller graph for this function:

12.37.3.2 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.

Definition at line 653 of file __l4_fpage.h.

References L4_WHOLE_ADDRESS_SPACE.

12.37.3.3 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 685 of file __l4_fpage.h.

References l4_fpage_memaddr().

Here is the call graph for this function:

12.37.3.4 l4_fpage_invalid()

```
l4_fpage_t l4_fpage_invalid (
            void  )  [inline]
```

Get an invalid flex page.

Returns

Special *invalid* flex page.

Definition at line 659 of file __l4_fpage.h.

12.37.3.5 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 589 of file __l4_fpage.h.

12.37.3.6 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 693 of file __l4_fpage.h.

References l4_trunc_size().

Referenced by L4Re::Dataspace::map(), and L4Re::Dataspace::map_region().

Here is the call graph for this function:

Here is the caller graph for this function:

12.37.3.7 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 595 of file __l4_fpage.h.

Referenced by l4_fpage_contains().

Here is the caller graph for this function:

12.37.3.8 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 601 of file __l4_fpage.h.

12.37.3.9 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 583 of file __l4_fpage.h.

12.37.3.10 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 565 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:

12.37.3.11 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 624 of file __l4_fpage.h.

References l4_fpage_t::raw.

12.37.3.12 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 577 of file __l4_fpage.h.

12.37.3.13 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 571 of file __l4_fpage.h.

12.37.3.14 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 639 of file __l4_fpage.h.

References L4_FPAGE_ADDR_SHIFT.

Referenced by l4util_ioport_map().

Here is the caller graph for this function:

12.37.3.15 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 666 of file __l4_fpage.h.

References l4_fpage_rights(), and L4_FPAGE_W.

Here is the call graph for this function:

12.37.3.16 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.

Definition at line 645 of file __l4_fpage.h.

Referenced by L4::Cap_base::fpage().

Here is the caller graph for this function:

12.38 Functions for rendering bitmap data in frame buffers

Collaboration diagram for Functions for rendering bitmap data in frame buffers:

Data Structures

- struct gfxbitmap_offset

  offsets in pmap[] and bmap[]

Typedefs

- typedef unsigned int gfxbitmap_color_t

  Standard color type.
- typedef unsigned int gfxbitmap_color_pix_t

  Specific color type.

Functions

- 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.

12.38.1 Detailed Description

12.38.2 Typedef Documentation

12.38.2.1 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 66 of file bitmap.h.

12.38.2.2 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 57 of file bitmap.h.

12.38.3 Function Documentation

12.38.3.1 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.

12.38.3.2 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.

12.38.3.3 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.

12.38.3.4 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.

12.38.3.5 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.

12.39 Functions for rendering bitmap fonts to frame buffers

Collaboration diagram for Functions for rendering bitmap fonts to frame buffers:

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.

12.39.1 Detailed Description

12.39.2 Enumeration Type Documentation

12.39.2.1 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.

12.39.3 Function Documentation

12.39.3.1 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.

12.39.3.2 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.

12.39.3.3 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.

12.39.3.4 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

12.39.3.5 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.

12.39.3.6 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.

12.39.3.7 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.

12.40 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.

12.40.1 Detailed Description

12.41 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

12.41.1 Detailed Description

12.41.2 Function Documentation

12.41.2.1 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.

12.41.2.2 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.

12.41.2.3 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:

12.41.2.4 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.

12.41.2.5 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.

12.41.2.6 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.

12.41.2.7 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.

12.41.2.8 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.

12.41.2.9 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:

12.41.2.10 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.

12.41.2.11 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.

12.41.2.12 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.

12.42 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_U↩
  SE_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_RE↩
  SOURCE_IRQ, L4IO_RESOURCE_MEM = L4VBUS_RESOURCE_MEM, L4IO_RESOURCE_PORT = L4↩
  VBUS_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_search_iomem_region (l4_addr_t phys, l4_addr_t size, l4_addr_t *rstart, l4_addr_t *rsize)

  Search for a 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.

12.42.1 Detailed Description

12.42.2 Typedef Documentation

12.42.2.1 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.

12.42.3 Enumeration Type Documentation

12.42.3.1 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.

12.42.3.2 l4io_iomem_flags_t

```
enum l4io_iomem_flags_t
```

Flags for IO memory.

Enumerator

L4IO_MEM_NONCACHED	Non-cache memory.

Enumerator

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.

12.42.3.3 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.

12.42.4 Function Documentation

12.42.4.1 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.

12.42.4.2 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

12.42.4.3 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

12.42.4.4 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

12.42.4.5 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

12.42.4.6 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_EADDRNOTAV↩AIL	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.

12.42.4.7 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_EADDRNOTAV↩AIL	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.

12.42.4.8 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

12.42.4.9 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.

12.42.4.10 l4io_search_iomem_region()

```
long l4io_search_iomem_region (
            l4_addr_t phys,
            l4_addr_t size,
            l4_addr_t * rstart,
            l4_addr_t * rsize )
```

Search for a IO memory region.

Parameters

phys	Physical address to look for
size	Size of requested memory area

Return values

rstart	Start address for region
rsize	Size of region in bytes

Returns

0 if an IO region was found, <0 if not

12.43 IPC-Gate API

Secure communication object.

Collaboration diagram for IPC-Gate API:

Functions

- l4_msgtag_t l4_ipc_gate_bind_thread (l4_cap_idx_t gate, l4_cap_idx_t thread, l4_umword_t label)

  Bind the IPC gate to a thread.
- 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 gate to a thread.

12.43.1 Detailed Description

Secure communication object.

IPC-Gate objects provide a means to establish secure communication channels to L4 Threads (Thread). An IPC-Gate object can be created using a Factory (l4_factory_create_gate()) and get assigned a specific L4 thread and a *label* as protected payload. The *label* has the size of one machine word and can only be seen by the Task running the thread that is assigned of the IPC-gate. The *label* is received as part of the IPC message. The *label* can thus be used to securely identify the IPC-gate that was used to send a message.

An IPC-gate is usually used to represent an user-level object and may be the address of the data structure for the object in the server task.

With client privileges an IPC-gate does not provide any direct API and thus an IPC-gate kernel object cannot be modified by invocations. Each invocation of an IPC-gate kernel object is translated into an IPC message to the assigned thread.

Include File

```
#include <l4/sys/ipc_gate.h>
```

For the C++ interface refer to the L4::Ipc_gate documentation.

See also

Object Invocation

12.43.2 Function Documentation

12.43.2.1 l4_ipc_gate_bind_thread()

```
l4_msgtag_t l4_ipc_gate_bind_thread (
            l4_cap_idx_t gate,
            l4_cap_idx_t thread,
            l4_umword_t label )  [inline]
```

Bind the IPC gate to a thread.

Parameters

gate	The IPC gate object.
thread	The thread object that shall be bound to `gate`.
label	Label to assign to `gate`. 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	`thread` is missing L4_CAP_FPAGE_S right.

Definition at line 130 of file ipc_gate.h.

12.43.2.2 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 137 of file ipc_gate.h.

12.43.2.3 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 gate 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	`thread` is missing L4_CAP_FPAGE_S right.

Examples:

examples/sys/isr/main.c.

Definition at line 80 of file rcv_endpoint.h.

12.44 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.

12.44.1 Detailed Description

12.45 IRQs

C IRQ 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_attach (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread) L4_NOTHROW

  Attach a thread to an interrupt source.
- l4_msgtag_t l4_irq_attach_u (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW

  Attach a thread to this interrupt.
- 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.
- 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.

12.45.1 Detailed Description

C IRQ 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)

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.

12.45.2 Enumeration Type Documentation

12.45.2.1 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.

Definition at line 67 of file icu.h.

12.45.3 Function Documentation

12.45.3.1 l4_irq_attach()

```
l4_msgtag_t l4_irq_attach (
            l4_cap_idx_t irq,
            l4_umword_t label,
            l4_cap_idx_t thread )  [inline]
```

Attach a thread to an interrupt source.

Parameters

irq	IRQ object where `thread` is attached to.
label	Identifier of the IRQ.
thread	The thread object to attach `irq` to.

Returns

Syscall return tag

The *protected label* is stored in the kernel and sent to the attached thread with the IRQ-triggered notification. It allows the receiver thread to securely identify the IRQ.

Definition at line 331 of file irq.h.

References l4_irq_attach_u(), and l4_utcb().

Here is the call graph for this function:

12.45.3.2 l4_irq_attach_u()

```
l4_msgtag_t l4_irq_attach_u (
            l4_cap_idx_t irq,
            l4_umword_t label,
            l4_cap_idx_t thread,
            l4_utcb_t * utcb )  [inline]
```

Attach a thread to this interrupt.

Parameters

irq	IRQ object where `thread` is attached to.
label	Identifier of the IRQ (*protected label* used for messages)
thread	Capability of the thread to attach the IRQ to.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

The *protected label* is stored in the kernel and sent to the attached thread with the IRQ-triggered notification. It allows the receiver thread to securely identify the IRQ.

Deprecated Use bind_thread().

Definition at line 261 of file irq.h.

Referenced by L4::Irq::attach(), and l4_irq_attach().

Here is the caller graph for this function:

12.45.3.3 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 347 of file irq.h.

References l4_irq_detach_u(), and l4_utcb().

Here is the call graph for this function:

12.45.3.4 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 292 of file irq.h.

Referenced by L4::Irq::detach(), and l4_irq_detach().

Here is the caller graph for this function:

12.45.3.5 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.

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 attached 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 341 of file irq.h.

References l4_irq_mux_chain_u(), and l4_utcb().

Here is the call graph for this function:

12.45.3.6 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 280 of file irq.h.

Referenced by L4::Irq_mux::chain(), and l4_irq_mux_chain().

Here is the caller graph for this function:

12.45.3.7 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 359 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:

12.45.3.8 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 307 of file irq.h.

Referenced by l4_irq_receive(), and L4::Irq::receive().

Here is the caller graph for this function:

12.45.3.9 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 353 of file irq.h.

References l4_irq_trigger_u(), and l4_utcb().

Here is the call graph for this function:

12.45.3.10 l4_irq_trigger_u()

```
l4_msgtag_t l4_irq_trigger_u (
            l4_cap_idx_t irq,
            l4_utcb_t * utcb )  [inline]
```

Trigger.

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 300 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:

12.45.3.11 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 372 of file irq.h.

References l4_irq_unmask_u(), and l4_utcb().

Here is the call graph for this function:

12.45.3.12 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 323 of file irq.h.

Referenced by l4_irq_unmask().

Here is the caller graph for this function:

12.45.3.13 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 365 of file irq.h.

References l4_irq_wait_u(), and l4_utcb().

Here is the call graph for this function:

12.45.3.14 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 314 of file irq.h.

Referenced by l4_irq_wait().

Here is the caller graph for this function:

12.46 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.

12.46.1 Detailed Description

C interface of the initial environment that is provided to an L4 task.

Include File

```
#include <l4/re/env.h>
```

For the C++ interface refer to L4Re::Env.

12.46.2 Function Documentation

12.46.2.1 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 181 of file env.h.

Referenced by l4re_env_get_cap().

Here is the caller graph for this function:

12.46.2.2 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 223 of file env.h.

References l4re_env(), l4re_env_get_cap_e(), and l4re_env_cap_entry_t::name.

Here is the call graph for this function:

12.46.2.3 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 210 of file env.h.

References l4re_env_cap_entry_t::cap, L4_INVALID_CAP, l4re_env_get_cap_l(), and l4re_env_cap_entry_t↩ ::name.

Referenced by l4re_env_get_cap().

Here is the call graph for this function:

Here is the caller graph for this function:

12.46.2.4 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 192 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:

12.46.2.5 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 185 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:

12.47 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.

12.47.1 Detailed Description

Include File

```
#include <l4/sys/l4int.h>
```

12.47.2 Typedef Documentation

12.47.2.1 l4_int16_t

```
typedef signed short int l4_int16_t
```

Signed 16bit value.

Definition at line 37 of file l4int.h.

12.47.2.2 l4_int32_t

```
typedef signed int l4_int32_t
```

Signed 32bit value.

Definition at line 39 of file l4int.h.

12.47.2.3 l4_int64_t

```
typedef signed long long l4_int64_t
```

Signed 64bit value.

Definition at line 41 of file l4int.h.

12.47.2.4 l4_int8_t

```
typedef signed char l4_int8_t
```

Signed 8bit value.

Definition at line 35 of file l4int.h.

12.47.2.5 l4_uint16_t

```
typedef unsigned short int l4_uint16_t
```

Unsigned 16bit value.

Definition at line 38 of file l4int.h.

12.47.2.6 l4_uint32_t

```
typedef unsigned int l4_uint32_t
```

Unsigned 32bit value.

Definition at line 40 of file l4int.h.

12.47.2.7 l4_uint64_t

```
typedef unsigned long long l4_uint64_t
```

Unsigned 64bit value.

Definition at line 42 of file l4int.h.

12.47.2.8 l4_uint8_t

```
typedef unsigned char l4_uint8_t
```

Unsigned 8bit value.

Definition at line 36 of file l4int.h.

12.48 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.

12.48.1 Detailed Description

This group is just an enhanced version to l4irq_request() which takes a capability object instead of a plain number.

12.48.2 Function Documentation

12.48.2.1 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

12.49 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.

12.49.1 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.

12.49.2 Function Documentation

12.49.2.1 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.

12.49.2.2 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.

12.50 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.

12.50.1 Detailed Description

12.50.2 Function Documentation

12.50.2.1 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.

12.50.2.2 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.

12.50.2.3 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.

12.50.2.4 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.

12.51 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.

12.51.1 Detailed Description

12.51.2 Function Documentation

12.51.2.1 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.

12.51.2.2 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.

12.51.2.3 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.

12.51.2.4 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.

12.51.2.5 l4irq_detach()

```
long l4irq_detach (
            l4irq_t * irq )
```

Detach from IRQ.

Parameters

irq	IRQ data structure

Returns

0 on success, != 0 on error

12.51.2.6 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.

12.51.2.7 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

12.51.2.8 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.

12.51.2.9 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

12.52 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.

12.52.1 Detailed Description

12.53 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_TBUF 0xB0

  TBUF.
- #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_TBUF (SIGMA0_REQ(TBUF))

  TBUF.

- #define SIGMA0_REQ_DEBUG_DUMP (SIGMA0_REQ(DEBUG_DUMP))

  Debug dump.

- #define SIGMA0_REQ_NEW_CLIENT (SIGMA0_REQ(NEW_CLIENT))

  New client.

12.53.1 Detailed Description

Internal sigma0 definitions.

12.54 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

12.54.1 Detailed Description

12.55 Interrupt controller

The C Icu 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 capabilites of the ICU.

- 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 capabilities of the ICU.
- 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

  Acknowledge 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.

12.55.1 Detailed Description

The C Icu interface.

To setup an IRQ line the following steps are required:

1. l4_icu_set_mode() (optional if IRQ has a default mode)
2. l4_irq_attach() to attach the IRQ capability to a thread
3. l4_icu_bind()
4. l4_icu_unmask() to receive the first IRQ

Include File

```
#include <l4/sys/icu.h>
```

12.55.2 Typedef Documentation

12.55.2.1 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().

12.55.3 Enumeration Type Documentation

12.55.3.1 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 50 of file icu.h.

12.55.4 Function Documentation

12.55.4.1 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().

Examples:

examples/sys/isr/main.c.

Definition at line 473 of file icu.h.

References l4_icu_bind_u(), and l4_utcb().

Here is the call graph for this function:

12.55.4.2 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 objecte 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.

Definition at line 373 of file icu.h.

Referenced by L4::Icu::bind(), and l4_icu_bind().

Here is the caller graph for this function:

12.55.4.3 l4_icu_info()

```
l4_msgtag_t l4_icu_info (
            l4_cap_idx_t icu,
            l4_icu_info_t * info )  [inline]
```

Get information about the capabilites of the ICU.

Parameters

	icu	The ICU object from which information shall be retrieved.
out	*info*	Pointer to an info structure to be filled with information. The memory for this structure has to be allocated by the caller.

Returns

Syscall return tag

Definition at line 481 of file icu.h.

References l4_icu_info_u(), and l4_utcb().

Here is the call graph for this function:

12.55.4.4 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 capabilities of the ICU.

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 397 of file icu.h.

Referenced by L4::Icu::info(), and l4_icu_info().

Here is the caller graph for this function:

12.55.4.5 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 495 of file icu.h.

12.55.4.6 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 460 of file icu.h.

Referenced by L4::Icu::mask().

Here is the caller graph for this function:

12.55.4.7 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 485 of file icu.h.

References l4_icu_msi_info_u(), and l4_utcb().

Here is the call graph for this function:

12.55.4.8 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 411 of file icu.h.

Referenced by l4_icu_msi_info().

Here is the caller graph for this function:

12.55.4.9 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 500 of file icu.h.

References l4_icu_set_mode_u(), and l4_utcb().

Here is the call graph for this function:

12.55.4.10 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 434 of file icu.h.

Referenced by l4_icu_set_mode(), and L4::Icu::set_mode().

Here is the caller graph for this function:

12.55.4.11 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 477 of file icu.h.

References l4_icu_unbind_u(), and l4_utcb().

Here is the call graph for this function:

12.55.4.12 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 385 of file icu.h.

Referenced by l4_icu_unbind(), and L4::Icu::unbind().

Here is the caller graph for this function:

12.55.4.13 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 490 of file icu.h.

12.55.4.14 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]
```

Acknowledge the given interrupt line.

Parameters

	icu	The ICU object where the IRQ line shall be unmasked.
	irqnum	The interrupt line that shall be acknowledged.
`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 465 of file icu.h.

12.56 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.

12.56.1 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>
```

12.56.2 Function Documentation

12.56.2.1 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 331 of file debugger.h.

12.56.2.2 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 337 of file debugger.h.

12.56.2.3 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 324 of file debugger.h.

12.57 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.

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.

12.57.1 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>
```

12.57.2 Function Documentation

12.57.2.1 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 134 of file kip.h.

Referenced by l4_kip_version_string().

Here is the caller graph for this function:

12.57.2.2 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.

Definition at line 138 of file kip.h.

References l4_mb().

Referenced by L4::Poll_timeout_kipclock::set(), and L4::Poll_timeout_kipclock::test().

Here is the call graph for this function:

Here is the caller graph for this function:

12.57.2.3 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.

Definition at line 160 of file kip.h.

References l4_mb().

Here is the call graph for this function:

12.57.2.4 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 126 of file kip.h.

12.57.2.5 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 130 of file kip.h.

References l4_kernel_info_version_offset().

Here is the call graph for this function:

12.58 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 native 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.
- l4_addr_t l4util_memdesc_vm_high (l4_kernel_info_t ∗kinfo)

  Return end of virtual memory.

12.58.1 Detailed Description

12.58.2 Macro Definition Documentation

12.58.2.1 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 ∗) initialized with l4util_kip_↩ version_string().

Definition at line 74 of file kip.h.

12.58.3 Function Documentation

12.58.3.1 l4util_kip_kernel_abi_version()

```
unsigned long l4util_kip_kernel_abi_version (
            l4_kernel_info_t *  )
```

Return kernel ABI version.

Returns

Kernel ABI version.

12.58.3.2 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. The KIP will be mapped if not already mapped. The KIP will not be unmapped again.

12.58.3.3 l4util_kip_kernel_is_ux()

```
int l4util_kip_kernel_is_ux (
            l4_kernel_info_t *  )
```

Return whether the kernel is running native or under UX.

Returns whether the kernel is running natively or under UX. The KIP will be mapped if not already mapped. The KIP will not be unmapped again.

Returns

1 when running under UX, 0 if not running under UX

Examples:

examples/sys/ux-vhw/main.c.

12.58.3.4 l4util_memdesc_vm_high()

```
l4_addr_t l4util_memdesc_vm_high (
            l4_kernel_info_t * kinfo )
```

Return end of virtual memory.

Returns

0 if memory descriptor could not be found, last address of address space otherwise

12.59 Kernel Objects

API of kernel objects.

Collaboration diagram for Kernel Objects:

Modules

- Factory

  C factory interface to create kernel objects.
- IPC-Gate API

  Secure communication object.
- IRQs

  C IRQ interface.
- Interrupt controller

  The C Icu 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.
- Scheduler

  C interface of the Scheduler kernel object.
- Task

  C interface of the Task kernel object.
- Thread

  Thread object.
- Virtual Console

  Virtual console for simple character based input and output.
- Virtual Machines

  Virtual Machine API.

Data Structures

- class L4::Kobject

  Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

12.59.1 Detailed Description

API of kernel objects.

Include File

```
#include <l4/sys/kernel_object.h>
```

12.60 Kumem allocator utility

Kumem allocator utility C interface.

Collaboration diagram for Kumem allocator utility:

Kumem allocator utility C interface.

12.61 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()) throw ()

  Allocate state area.

12.61.1 Detailed Description

12.61.2 Function Documentation

12.61.2.1 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() ) throw )
```

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 minmum allocatable amount is one page. A portable implementation should not depend on allocations greater than 16KiB to succeed.

12.62 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_S↩YS_SHUTDOWN_OP = 1UL, L4_PLATFORM_CTL_CPU_ENABLE_OP = 3UL, L4_PLATFORM_CTL_C↩PU_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 }

  Operations on task objects.
- enum L4_thread_ops {
  L4_THREAD_CONTROL_OP = 0UL, L4_THREAD_EX_REGS_OP = 1UL, L4_THREAD_SWITCH_OP = 2↩UL, L4_THREAD_STATS_OP = 3UL,
  L4_THREAD_VCPU_RESUME_OP = 4UL, L4_THREAD_REGISTER_DELETE_IRQ_OP = 5UL, L4_THR↩EAD_MODIFY_SENDER_OP = 6UL, L4_THREAD_VCPU_CONTROL_OP = 7UL ,
  L4_THREAD_X86_GDT_OP = 0x10UL, L4_THREAD_ARM_TPIDRURO_OP = 0x10UL, L4_THREAD_A↩MD64_SET_SEGMENT_BASE_OP = 0x12UL, L4_THREAD_AMD64_GET_SEGMENT_INFO_OP = 0x13↩UL,
  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_ATT↩R_OP = 2UL, L4_VCON_GET_ATTR_OP = 3UL }

  Operations on vcon objects.

12.62.1 Detailed Description

List of protocol specific opcodes used for communication with L4Re and Kernel objects.

12.62.2 Enumeration Type Documentation

12.62.2.1 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 93 of file icu.h.

12.62.2.2 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 94 of file ipc_gate.h.

12.62.2.3 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_ENABLE_OP	enable an offline CPU
L4_PLATFORM_CTL_CPU_DISABLE_OP	disable an online CPU

Definition at line 135 of file platform_control.h.

12.62.2.4 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

Definition at line 273 of file task.h.

12.62.2.5 L4_thread_ops

`enum L4_thread_ops`

Operations on thread objects.

Enumerator

L4_THREAD_CONTROL_OP	Control operation.

Enumerator

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 612 of file thread.h.

12.62.2.6 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 264 of file vcon.h.

12.63 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_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_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 }

  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.

12.63.1 Detailed Description

12.63.2 Enumeration Type Documentation

12.63.2.1 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.

Definition at line 31 of file virtio_block.h.

12.63.2.2 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 42 of file virtio_block.h.

12.64 L4 VIRTIO Interface

Collaboration diagram for L4 VIRTIO Interface:

Modules

- L4 VIRTIO Block Device
- L4 VIRTIO Transport Layer

  L4 specific VIRTIO Transport layer.

12.64.1 Detailed Description

12.65 L4 VIRTIO Transport Layer

L4 specific VIRTIO Transport layer.

Collaboration diagram for L4 VIRTIO Transport Layer:

Namespaces

- L4virtio

  L4-VIRTIO Transport C++ API.

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_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 given L4-VIRTIO host.

12.65.1 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.

12.65.2 Typedef Documentation

12.65.2.1 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.

12.65.3 Enumeration Type Documentation

12.65.3.1 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 116 of file virtio.h.

12.65.3.2 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 107 of file virtio.h.

12.65.3.3 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 55 of file virtio.h.

12.65.3.4 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 balooning device.
L4VIRTIO_ID_RPMSG	Device using rpmsg protocol.
L4VIRTIO_ID_SCSI	SCSI host device.
L4VIRTIO_ID_9P	Device using 9P transport protocol.
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	Inofficial socket device.

Definition at line 64 of file virtio.h.

12.65.3.5 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_FAILED	Fatal error in driver or device.

Definition at line 85 of file virtio.h.

12.65.3.6 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 95 of file virtio.h.

12.65.4 Function Documentation

12.65.4.1 l4virtio_config_queue()

```
int l4virtio_config_queue (
            l4_cap_idx_t cap,
            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

cap	Capability to the VIRTIO host.
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_get_feature().

Here is the caller graph for this function:

12.65.4.2 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 237 of file virtio.h.

12.65.4.3 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 248 of file virtio.h.

12.65.4.4 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	Data-space 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_get_feature().

Here is the caller graph for this function:

12.65.4.5 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 given L4-VIRTIO host.

Parameters

cap	Capability to the L4-VIRTIO host
guest_irq	IRQ capability for valid IRQ object for host-to-guest notifications
host_irq	Capability selector for receiving the guest-to-host notifications IRQ capability.
config_ds	Capability for receiving the data-space capability for the shared L4-VIRTIO config data space.

Return values

0	on success.
-L4_EINVAL	The host did not receive the *guest_irq* cap.

Referenced by l4virtio_get_feature().

Here is the caller graph for this function:

12.65.4.6 l4virtio_set_status()

```
int l4virtio_set_status (
            l4_cap_idx_t cap,
            unsigned status )
```

Write the VIRTIO status register.

Note

All other registers are accessed via shared memory.

Parameters

cap	Capability to the VIRTIO host
status	Status word to write to the VIRTIO status.

Returns

0 on success, <0 on error.

Referenced by l4virtio_get_feature().

Here is the caller graph for this function:

12.66 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 human 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_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_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.

12.66.1 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.

12.66.2 Enumeration Type Documentation

12.66.2.1 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 153 of file vbus.h.

12.66.3 Function Documentation

12.66.3.1 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:

12.66.3.2 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 after a successfull call.

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:

12.66.3.3 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 human 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

Parameters

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.

12.66.3.4 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

12.66.3.5 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 0 for the system bus)
	child	Handle to the child device (use 0 to get the first child)
	depth	Depth to look for
out	*devinfo*	device information (might be NULL)

Returns

0 on success, else failure

12.66.3.6 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:

12.66.3.7 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:

12.66.3.8 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.
res	Descriptor of the resource.

Returns

0 on success, else failure

Referenced by L4vbus::Vbus::release_resource().

Here is the caller graph for this function:

12.66.3.9 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
res	Descriptor of the resource
flags	Optional flags

Returns

0 on success, else failure

If any resource is found that contains the requested type and addresses this resource is returned.

Flags are only relevant to control the memory caching. If io-memory is requested.

Returns

0 on success, else failure

Referenced by L4vbus::Vbus::request_resource().

Here is the caller graph for this function:

12.66.3.10 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:

12.67 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 ()

  Get the L4::Type_info for the L4Re interface given in `T`.

12.67.1 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.

12.67.2 Function Documentation

12.67.2.1 kobject_typeid()

```
template<typename T >
Type_info const* L4::kobject_typeid ( ) [inline]
```

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(), and L4::PROTO_ANY.

Here is the call graph for this function:

12.68 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.

12.68.1 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.

12.68.2 Function Documentation

12.68.2.1 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()

12.69 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 interface.
- 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.

12.69.1 Detailed Description

Documentation of the L4 Runtime Environment C++ API.

12.70 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 Auto_cap and Auto_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 Auto_cap and Auto_del_cap.
- class L4Re::Util::Smart_count_cap< Unmap_flags >

  Helper for Ref_cap and Ref_del_cap.
- struct L4Re::Util::Auto_cap< T >

  Automatic capability that implements automatic free and unmap of the capability selector.
- struct L4Re::Util::Auto_del_cap< T >

  Automatic capability that implements automatic free and unmap+delete of the capability selector.
- 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 >
  Auto_cap< T >::Cap L4Re::Util::make_auto_cap ()

  Allocate a capability slot and wrap it in an Auto_cap.
- template<typename T >
  Auto_del_cap< T >::Cap L4Re::Util::make_auto_del_cap ()

  Allocate a capability slot and wrap it in an Auto_del_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<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.

12.70.1 Detailed Description

12.70.2 Function Documentation

12.70.2.1 make_auto_cap()

```
template<typename T >
Auto_cap<T>::Cap L4Re::Util::make_auto_cap ( )
```

Allocate a capability slot and wrap it in an Auto_cap.

Template Parameters

T	Type of capability the slot is used for.

Deprecated Use L4Re::Util::make_unique_cap().

Definition at line 293 of file cap_alloc.

12.70.2.2 make_auto_del_cap()

```
template<typename T >
Auto_del_cap<T>::Cap L4Re::Util::make_auto_del_cap ( )
```

Allocate a capability slot and wrap it in an Auto_del_cap.

Template Parameters

T	Type of capability the slot is used for.

Deprecated Use L4Re::Util::make_unique_del_cap().

Definition at line 305 of file cap_alloc.

12.70.2.3 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 316 of file cap_alloc.

12.70.2.4 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 325 of file cap_alloc.

12.70.3 Variable Documentation

12.70.3.1 cap_alloc

```
_Cap_alloc& L4Re::Util::cap_alloc
```

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/ds_srv.cc, and examples/libs/l4re/streammap/client.cc.

Referenced by 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::Object_registry::unregister_obj().

12.71 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 }

12.71.1 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.

12.71.2 Macro Definition Documentation

12.71.2.1 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_A↩ UX_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.

12.71.2.2 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.

12.71.3 Enumeration Type Documentation

12.71.3.1 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.

12.72 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.

12.72.1 Detailed Description

Fix L4Re Protocol Constants.

12.73 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.

12.74 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 Auto_cap and Auto_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.

12.74.1 Detailed Description

Documentation of the L4 Runtime Environment utility functionality in C++.

12.75 L4vbus GPIO functions

Collaboration diagram for L4vbus GPIO functions:

Data Structures

- class L4vbus::Gpio_pin

  A GPIO pin.
- class L4vbus::Gpio_module

  A Gpio_module groups multiple GPIO pins together.

Enumerations

- enum L4vbus_gpio_generic_func { L4VBUS_GPIO_SETUP_INPUT = 0x100, L4VBUS_GPIO_SETUP_O↩UTPUT = 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_PUL↩L_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.

12.75.1 Detailed Description

12.75.2 Enumeration Type Documentation

12.75.2.1 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.

12.75.2.2 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.

12.75.3 Function Documentation

12.75.3.1 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:

12.75.3.2 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:

12.75.3.3 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:

12.75.3.4 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:

12.75.3.5 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.
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:

12.75.3.6 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:

12.75.3.7 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:

12.75.3.8 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.
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:

12.75.3.9 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:

12.75.3.10 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:

12.75.3.11 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:

12.76 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.

12.76.1 Detailed Description

12.76.2 Function Documentation

12.76.2.1 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:

12.76.2.2 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:

12.76.2.3 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:

12.76.2.4 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:

12.76.2.5 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:

12.76.2.6 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:

12.77 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.

12.77.1 Detailed Description

Log C interface.

12.77.2 Function Documentation

12.77.2.1 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:

12.77.2.2 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:

12.77.2.3 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:

12.77.2.4 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:

12.78 Logging interface

Interface for log output.

Collaboration diagram for Logging interface:

Data Structures

- class L4Re::Log

  Log interface class.

12.78.1 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.

12.79 Low-Level Thread Functions

Collaboration diagram for Low-Level Thread Functions:

12.80 Machine Restarting Function

Collaboration diagram for Machine Restarting Function:

Functions

- void l4util_reboot (void))

  Machine reboot.

12.80.1 Detailed Description

12.81 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_N↩OTHROW

  Allocate memory.
- long l4re_ma_free (l4re_ds_t const mem) L4_NOTHROW

  Free 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.
- long l4re_ma_free_srv (l4_cap_idx_t srv, l4re_ds_t const mem) L4_NOTHROW

  Free memory.

12.81.1 Detailed Description

Memory allocator C interface.

12.81.2 Enumeration Type Documentation

12.81.2.1 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.

12.81.3 Function Documentation

12.81.3.1 l4re_ma_alloc()

```
long l4re_ma_alloc (
            long size,
            l4re_ds_t const mem,
            unsigned long flags )  [inline]
```

Allocate memory.

Parameters

size	Size to be requested in bytes (granularity is (super)pages and the size is rounded up to this granularity).
mem	Capability slot to put the requested dataspace in
flags	Flags, see l4re_ma_flags

Returns

0 on success, <0 on 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 167 of file mem_alloc.h.

References l4re_ma_alloc_align_srv().

Here is the call graph for this function:

12.81.3.2 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 to be requested in bytes (granularity is (super)pages and the size is rounded up to this granularity).
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 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 175 of file mem_alloc.h.

References l4re_ma_alloc_align_srv().

Here is the call graph for this function:

12.81.3.3 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:

12.81.3.4 l4re_ma_free()

```
long l4re_ma_free (
            l4re_ds_t const mem )  [inline]
```

Free memory.

Parameters

mem	Dataspace to free.

Returns

0 on success, <0 on error

See also

L4Re::Mem_alloc::free

Note

This function is using the L4Re::Env::env()->mem_alloc() service.

Deprecated This function is deprecated. Use l4_task_unmap() or similar means to remove a dataspace.

Definition at line 183 of file mem_alloc.h.

References l4re_ma_free_srv().

Here is the call graph for this function:

12.81.3.5 l4re_ma_free_srv()

```
long l4re_ma_free_srv (
            l4_cap_idx_t srv,
            l4re_ds_t const mem )
```

Free memory.

Parameters

srv	Memory allocator service.
mem	Dataspace to free.

Returns

0 on success, <0 on error

See also

L4Re::Mem_alloc::free

Deprecated This function is deprecated. Use l4_task_unmap() or similar means to remove a dataspace.

Referenced by l4re_ma_free().

Here is the caller graph for this function:

12.82 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.

12.82.1 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).

12.82.2 Typedef Documentation

12.82.2.1 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.

12.82.3 Enumeration Type Documentation

12.82.3.1 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.

12.82.3.2 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.

12.82.4 Function Documentation

12.82.4.1 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.

12.82.4.2 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.

12.82.4.3 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.

12.82.4.4 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.

12.82.4.5 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.

12.82.4.6 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.

References l4_kernel_info_t::mem_info.

12.82.4.7 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.

12.83 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 memory from kernel privilege level.
- void l4_mem_write (unsigned long virtaddress, unsigned width, unsigned long value)

  Write memory from kernel privilege level.

12.83.1 Detailed Description

Operations for memory access.

This modules 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>
```

12.83.2 Enumeration Type Documentation

12.83.2.1 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.

12.83.3 Function Documentation

12.83.3.1 l4_mem_read()

```
unsigned long l4_mem_read (
            unsigned long virtaddress,
            unsigned width )  [inline]
```

Read 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:

12.83.3.2 l4_mem_write()

```
void l4_mem_write (
            unsigned long virtaddress,
            unsigned width,
            unsigned long value )  [inline]
```

Write 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:

12.84 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_umword_t value, unsigned char bits) L4_NOTHROW

  Round value up to the next alignment with bits size.

12.84.1 Detailed Description

Memory related constants, data types and functions.

12.84.2 Macro Definition Documentation

12.84.2.1 L4_LOG2_PAGESIZE

```
#define L4_LOG2_PAGESIZE
```

Number of bits used for page offset.

Size of page in log2.

Definition at line 297 of file consts.h.

Referenced by L4Re::Rm::attach(), L4Re::Dataspace::map(), L4Re::Dataspace::map_region(), and L4Re::Util::↩ Dataspace_svr::page_shift().

12.84.2.2 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 323 of file consts.h.

12.84.2.3 L4_PAGEMASK

```
#define L4_PAGEMASK
```

Mask for the page number.

Note

The most significant bits are set.

Definition at line 288 of file consts.h.

Referenced by l4_round_page(), l4_sleep_forever(), and l4_trunc_page().

12.84.2.4 L4_SUPERPAGEMASK

```
#define L4_SUPERPAGEMASK
```

Mask for the number of a large page.

Note

The most significant bits are set.

Definition at line 315 of file consts.h.

12.84.2.5 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 306 of file consts.h.

12.84.3 Enumeration Type Documentation

12.84.3.1 l4_addr_consts_t

```
enum l4_addr_consts_t
```

Address related constants.

Enumerator

L4_INVALID_ADDR	Invalid address.

Definition at line 379 of file consts.h.

12.84.4 Function Documentation

12.84.4.1 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 361 of file consts.h.

References L4_NOTHROW, L4_PAGEMASK, L4_PAGESIZE, and l4_round_size().

Referenced by L4Re::Rm::attach(), l4_trunc_size(), L4Re::Dataspace::map(), and L4Re::Dataspace::map_region().

Here is the call graph for this function:

Here is the caller graph for this function:

12.84.4.2 l4_round_size()

```
l4_addr_t l4_round_size (
            l4_umword_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 372 of file consts.h.

Referenced by cxx::List_alloc::alloc_max(), L4Re::Util::Dataspace_svr::is_static(), l4_round_page(), L4Re::↩ Dataspace::map_region(), and L4virtio::Virtqueue::setup_simple().

Here is the caller graph for this function:

12.84.4.3 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 mininmal 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 336 of file consts.h.

References L4_NOTHROW, L4_PAGEMASK, and l4_trunc_size().

Referenced by l4_touch_ro(), l4_touch_rw(), L4Re::Dataspace::map(), and L4Re::Dataspace::map_region().

Here is the call graph for this function:

Here is the caller graph for this function:

12.84.4.4 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 347 of file consts.h.

References L4_NOTHROW, and l4_round_page().

Referenced by cxx::List_alloc::alloc_max(), l4_fpage_max_order(), l4_trunc_page(), and L4Re::Dataspace::map↩_region().

Here is the call graph for this function:

Here is the caller graph for this function:

12.85 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.

12.85.1 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.

12.85.2 Enumeration Type Documentation

12.85.2.1 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 187 of file consts.h.

12.85.3 Function Documentation

12.85.3.1 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.

Returns

The value to be used as first word in a map item for memory.

Definition at line 672 of file __l4_fpage.h.

Referenced by l4_map_obj_control().

Here is the caller graph for this function:

12.85.3.2 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.

Returns

The value to be used as first word in a map item for kernel objects or IO-ports.

Definition at line 679 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:

12.86 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.

12.86.1 Detailed Description

12.87 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,
  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.

12.87.1 Detailed Description

API related to the message tag data type.

Include File

```
#include <l4/sys/types.h>
```

12.87.2 Typedef Documentation

12.87.2.1 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.

12.87.3 Enumeration Type Documentation

12.87.3.1 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 transfered 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_PROPAGATE	Enable IPC propagation. This flag enables IPC propagation, which means an IPC reply-connection from the current caller will be propagated to the new IPC receiver. This makes it possible to propagate an IPC call to a third thread, which may then directly answer to the caller.
L4_MSGTAG_FLAGS	Mask for all flags.

Definition at line 95 of file types.h.

12.87.3.2 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.

Enumerator

L4_PROTO_IO_PAGE_FAULT	I/O page fault message.
L4_PROTO_KOBJECT	Protocol for messages to a 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 ddebugger.
L4_PROTO_SMCCC	Protocol ID for ARM SMCCC calls.

Definition at line 49 of file types.h.

12.87.3.3 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 147 of file platform_control.h.

12.87.4 Function Documentation

12.87.4.1 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/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 408 of file types.h.

Referenced by l4_msgtag_t::has_error(), L4::Server< LOOP_HOOKS >::internal_loop(), l4_irq_trigger_u(), l4util↩_ioport_map(), L4::Thread::modify_senders(), L4::Ipc_svr::Exc_dispatch< R, Exc >::operator()(), and L4::Ipc::↩Ostream::utcb().

Here is the caller graph for this function:

12.87.4.2 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 431 of file types.h.

References l4_msgtag_t::raw.

12.87.4.3 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 436 of file types.h.

References L4_MSGTAG_ERROR, and l4_msgtag_t::raw.

Referenced by l4_msgtag_t::has_error(), l4_ipc_error(), l4_vcon_write_u(), and l4vcpu_irq_disable_save().

Here is the caller graph for this function:

12.87.4.4 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 450 of file types.h.

References l4_msgtag_label(), and L4_PROTO_EXCEPTION.

Referenced by l4_msgtag_t::has_error().

Here is the call graph for this function:

Here is the caller graph for this function:

12.87.4.5 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 456 of file types.h.

References l4_msgtag_label(), and L4_PROTO_IO_PAGE_FAULT.

Referenced by l4_msgtag_t::has_error().

Here is the call graph for this function:

Here is the caller graph for this function:

12.87.4.6 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 441 of file types.h.

References l4_msgtag_label(), and L4_PROTO_PAGE_FAULT.

Referenced by l4_msgtag_t::has_error().

Here is the call graph for this function:

Here is the caller graph for this function:

12.87.4.7 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 444 of file types.h.

References l4_msgtag_label(), and L4_PROTO_PREEMPTION.

Referenced by l4_msgtag_t::has_error().

Here is the call graph for this function:

Here is the caller graph for this function:

12.87.4.8 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 453 of file types.h.

References l4_msgtag_label(), and L4_PROTO_SIGMA0.

Referenced by l4_msgtag_t::has_error().

Here is the call graph for this function:

Here is the caller graph for this function:

12.87.4.9 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 447 of file types.h.

References l4_msgtag_label(), and L4_PROTO_SYS_EXCEPTION.

Referenced by l4_msgtag_t::has_error().

Here is the call graph for this function:

Here is the caller graph for this function:

12.87.4.10 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 427 of file types.h.

References l4_msgtag_t::raw.

Referenced by l4_msgtag_t::has_error(), and l4util_ioport_map().

Here is the caller graph for this function:

12.87.4.11 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 419 of file types.h.

References l4_msgtag_t::raw.

Referenced by l4_msgtag_t::has_error(), 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:

12.87.4.12 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 423 of file types.h.

References l4_msgtag_t::raw.

Referenced by l4_msgtag_t::has_error().

Here is the caller graph for this function:

12.88 Name-space API

API for name spaces that store capabilities.

Collaboration diagram for Name-space API:

Data Structures

- class L4Re::Namespace

  Name-space interface.

12.88.1 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.

12.89 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.

12.89.1 Detailed Description

Namespace C interface.

12.89.2 Enumeration Type Documentation

12.89.2.1 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.

12.89.3 Function Documentation

12.89.3.1 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 EXTERN_C_END, and l4re_ns_query_to_srv().

Here is the call graph for this function:

12.89.3.2 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:

12.89.3.3 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.
-L4_EINVAL	Invalid parameter.
<0	IPC errors, see l4_error_code_t.

Precondition

requires capability rights: {RW}

12.90 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_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

  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.

12.90.1 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

12.90.2 Enumeration Type Documentation

12.90.2.1 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 send 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.

12.90.3 Function Documentation

12.90.3.1 l4_ipc()

```
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 )  [inline]
```

Generic L4 object invocation.

Parameters

	dest	Destination object.
	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:

12.90.3.2 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.
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 445 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:

12.90.3.3 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.
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 487 of file ipc.h.

References l4_ipc(), L4_SYSF_RECV, and l4_msgtag_t::raw.

Referenced by l4_ipc_sleep(), l4_thread_yield(), and L4::Ipc::Istream::wait().

Here is the call graph for this function:

Here is the caller graph for this function:

12.90.3.4 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

	tag	Describes the message to be sent as reply.
	utcb	UTCB of the caller.
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 453 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:

12.90.3.5 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.
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 470 of file ipc.h.

References l4_ipc(), and L4_SYSF_SEND.

Referenced by l4_irq_trigger_u(), L4::Ipc::Iostream::reply_and_wait(), 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:

12.90.3.6 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.
	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 461 of file ipc.h.

References l4_ipc(), and L4_SYSF_SEND_AND_WAIT.

Referenced by L4::Ipc::Iostream::reply_and_wait().

Here is the call graph for this function:

Here is the caller graph for this function:

12.90.3.7 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 496 of file ipc.h.

References L4_INVALID_CAP, and l4_ipc_receive().

Referenced by l4_sleep_forever().

Here is the call graph for this function:

Here is the caller graph for this function:

12.90.3.8 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.

Return values

label	Label assigned to the source object (IPC gate or IRQ).

Parameters

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 478 of file ipc.h.

References L4_INVALID_CAP, l4_ipc(), L4_SYSF_WAIT, and l4_msgtag_t::raw.

Referenced by l4vcpu_irq_disable_save(), 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:

12.90.3.9 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 556 of file ipc.h.

12.91 Parent API

Parent interface.

Collaboration diagram for Parent API:

Data Structures

- class L4Re::Parent

  Parent interface.

12.91.1 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.

See also

L4Re::Parent for information about the concrete interface.

12.92 Platform Control C API

C interface for controlling platform-wide properties.

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_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.

12.92.1 Detailed Description

C interface for controlling platform-wide properties.

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

12.92.2 Function Documentation

12.92.2.1 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

Definition at line 232 of file platform_control.h.

12.92.2.2 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

Definition at line 225 of file platform_control.h.

12.92.2.3 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 194 of file platform_control.h.

12.92.2.4 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.

Returns

Syscall return tag

Definition at line 187 of file platform_control.h.

12.93 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.

12.93.1 Detailed Description

12.93.2 Function Documentation

12.93.2.1 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.

12.93.2.2 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.

12.93.2.3 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.

12.93.2.4 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.

12.94 Producer

Collaboration diagram for Producer:

Functions

- long l4shmc_trigger (l4shmc_signal_t *signal)

  Trigger a signal.

12.94.1 Detailed Description

12.94.2 Function Documentation

12.94.2.1 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.

12.95 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.

12.95.1 Detailed Description

12.95.2 Function Documentation

12.95.2.1 l4util_rand()

```
l4_uint32_t l4util_rand (
            void  )
```

Deliver next random number.

Returns

A new random number

12.95.2.2 l4util_srand()

```
void l4util_srand (
            l4_uint32_t seed )
```

Initialize random number generator.

Parameters

seed	Value to initialize

12.96 Realtime API

Collaboration diagram for Realtime API:

12.97 Region map API

Virtual address-space management.

Collaboration diagram for Region map API:

Data Structures

- class L4Re::Rm

  Region map.

12.97.1 Detailed Description

Virtual address-space management.

The central purpose of the region-map API is to provide means to manage the virtual memory address space of an L4 task. 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, that allows to attach a data-space object to a region of the virtual address space.

There are two basic concepts provided by a region-map abstraction:

- Regions provide a means to create a view to a data space (or parts of a data space).
- Areas provide a means to reserve areas in a virtual memory address space for special purposes. A reserved area is skipped when searching for an available range of virtual memory, or may be explicitly used to search only within that area.

See also

L4Re::Dataspace, L4Re::Rm

12.98 Region map interface

Region map C interface.

Collaboration diagram for Region map interface:

Enumerations

- enum l4re_rm_flags_t {
  L4RE_RM_READ_ONLY = 0x01, L4RE_RM_NO_ALIAS = 0x02, L4RE_RM_PAGER = 0x04, L4RE_RM_↩
  RESERVED = 0x08,
  L4RE_RM_REGION_FLAGS = 0x0f, L4RE_RM_OVERMAP = 0x10, L4RE_RM_SEARCH_ADDR = 0x20,
  L4RE_RM_IN_AREA = 0x40,
  L4RE_RM_EAGER_MAP = 0x80, L4RE_RM_ATTACH_FLAGS = 0xf0 }

  Flags for region operations.

Functions

- int l4re_rm_reserve_area (l4_addr_t ∗start, unsigned long size, unsigned flags, unsigned char align) L4_N↩
  OTHROW
- int l4re_rm_free_area (l4_addr_t addr) L4_NOTHROW
- int l4re_rm_attach (void ∗∗start, unsigned long size, unsigned long flags, l4re_ds_t const mem, l4_addr_t offs, unsigned char align) L4_NOTHROW
- int l4re_rm_detach (void ∗addr) L4_NOTHROW

  Detach and unmap in current task.
- int l4re_rm_detach_ds (void ∗addr, l4re_ds_t ∗ds) L4_NOTHROW

  Detach, unmap and return affected dataspace in 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, l4_addr_t ∗offset, unsigned ∗flags, l4re_ds_t ∗m) L4_NOTHROW
- 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, unsigned 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, unsigned long flags, l4re_ds_t const mem, l4_addr_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, l4_addr_t ∗offset, unsigned ∗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.

12.98.1 Detailed Description

Region map C interface.

12.98.2 Enumeration Type Documentation

12.98.2.1 l4re_rm_flags_t

```
enum l4re_rm_flags_t
```

Flags for region operations.

Enumerator

L4RE_RM_READ_ONLY	Region is read-only.
L4RE_RM_NO_ALIAS	The region contains exclusive memory that is not mapped anywhere else.
L4RE_RM_PAGER	Region has a pager.
L4RE_RM_RESERVED	Region is reserved (blocked)
L4RE_RM_REGION_FLAGS	Mask of all region flags.
L4RE_RM_OVERMAP	Unmap memory already mapped in the region.
L4RE_RM_SEARCH_ADDR	Search for a suitable address range.
L4RE_RM_IN_AREA	Search only in area, or map into area.
L4RE_RM_EAGER_MAP	Eagerly map the attached data space in.
L4RE_RM_ATTACH_FLAGS	Mask of all attach flags.

Definition at line 40 of file rm.h.

12.98.3 Function Documentation

12.98.3.1 l4re_rm_attach()

```
int l4re_rm_attach (
            void ** start,
            unsigned long size,
            unsigned long flags,
            l4re_ds_t const mem,
            l4_addr_t offs,
            unsigned char align )  [inline]
```

Parameters

in,out	*start*	Virtual start address where the region manager shall attach the data space. If L4Re::Rm::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::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)
	flags	Flags, see L4Re::Rm::Attach_flags and L4Re::Rm::Region_flags. If the `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::Search_addr flag is used.

Return values

0	Success
-L4_ENOENT	No area could be found (see L4Re::Rm::In_area)
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAV↩AIL	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).

Returns

0 on success, <0 on error

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 243 of file rm.h.

References l4re_rm_attach_srv().

Here is the call graph for this function:

12.98.3.2 l4re_rm_attach_srv()

```
int l4re_rm_attach_srv (
            l4_cap_idx_t rm,
            void ** start,
            unsigned long size,
            unsigned long flags,
            l4re_ds_t const mem,
            l4_addr_t offs,
            unsigned char align )
```

See also

L4Re::Rm::attach

Referenced by l4re_rm_attach().

Here is the caller graph for this function:

12.98.3.3 l4re_rm_detach()

```
int l4re_rm_detach (
            void * addr )  [inline]
```

Detach and unmap in current task.

Parameters

addr	Address of the region to detach.

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 253 of file rm.h.

References l4re_rm_detach_srv().

Here is the call graph for this function:

12.98.3.4 l4re_rm_detach_ds()

```
int l4re_rm_detach_ds (
            void * addr,
            l4re_ds_t * ds )  [inline]
```

Detach, unmap and return affected dataspace in current task.

Parameters

addr	Address of the region to detach.

Return values

ds	Returns dataspace that is affected.

Returns

0 on success, <0 on error

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 266 of file rm.h.

References l4re_rm_detach_srv().

Here is the call graph for this function:

12.98.3.5 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 273 of file rm.h.

References l4re_rm_detach_srv().

Here is the call graph for this function:

12.98.3.6 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:

12.98.3.7 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 260 of file rm.h.

References l4re_rm_detach_srv().

Here is the call graph for this function:

12.98.3.8 l4re_rm_find()

```
int l4re_rm_find (
            l4_addr_t * addr,
            unsigned long * size,
            l4_addr_t * offset,
            unsigned * flags,
            l4re_ds_t * m )  [inline]
```

Returns

0 on success, <0 on error

See also

L4Re::Rm::find

Definition at line 280 of file rm.h.

References l4re_rm_find_srv().

Here is the call graph for this function:

12.98.3.9 l4re_rm_find_srv()

```
int l4re_rm_find_srv (
            l4_cap_idx_t rm,
            l4_addr_t * addr,
            unsigned long * size,
            l4_addr_t * offset,
            unsigned * flags,
            l4re_ds_t * m )
```

See also

L4Re::Rm::find

Referenced by l4re_rm_find().

Here is the caller graph for this function:

12.98.3.10 l4re_rm_free_area()

```
int l4re_rm_free_area (
            l4_addr_t addr )  [inline]
```

Returns

0 on success, <0 on error

See also

L4Re::Rm::free_area

This function is using the L4::Env::env()->rm() service.

Definition at line 237 of file rm.h.

References l4re_rm_free_area_srv().

Here is the call graph for this function:

12.98.3.11 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:

12.98.3.12 l4re_rm_reserve_area()

```
int l4re_rm_reserve_area (
            l4_addr_t * start,
            unsigned long size,
            unsigned flags,
            unsigned char align )  [inline]
```

Returns

0 on success, <0 on error

See also

L4Re::Rm::reserve_area

This function is using the L4::Env::env()->rm() service.

Definition at line 229 of file rm.h.

References l4re_rm_reserve_area_srv().

Here is the call graph for this function:

12.98.3.13 l4re_rm_reserve_area_srv()

```
int l4re_rm_reserve_area_srv (
            l4_cap_idx_t rm,
            l4_addr_t * start,
            unsigned long size,
            unsigned flags,
            unsigned char align )
```

See also

L4Re::Rm::reserve_area

Referenced by l4re_rm_reserve_area().

Here is the caller graph for this function:

12.98.3.14 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 287 of file rm.h.

References l4re_rm_show_lists_srv().

Here is the call graph for this function:

12.99 Scheduler

C interface of the Scheduler kernel object.

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_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_cpu_time_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.

12.99.1 Detailed Description

C interface of the Scheduler kernel object.

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.

Include File

```
#include <l4/sys/scheduler.h>
```

12.99.2 Enumeration Type Documentation

12.99.2.1 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 201 of file scheduler.h.

12.99.3 Function Documentation

12.99.3.1 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.
granularity	Granularitry 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 211 of file scheduler.h.

References l4_sched_cpu_set_t::gran_offset, l4_sched_cpu_set_t::granularity(), l4_sched_cpu_set_t::map, and l4_sched_cpu_set_t::offset().

Referenced by l4_sched_param(), and l4_sched_cpu_set_t::set().

Here is the call graph for this function:

Here is the caller graph for this function:

12.99.3.2 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 323 of file scheduler.h.

12.99.3.3 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.
`in,out`	*cpus*	*cpus.offset* is first CPU of interest. *cpus.granularity* (see l4_sched_cpu_set_t). *cpus.map* Bitmap of online CPUs.

Return values

0	Success.
-L4_EINVAL	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 309 of file scheduler.h.

12.99.3.4 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 330 of file scheduler.h.

12.99.3.5 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.
-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 316 of file scheduler.h.

12.100 Server-Side IPC framework

Server-Side framework for implementing object-oriented servers.

Namespaces

- L4::Ipc_svr

  Helper classes for L4::Server instantiation.

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::Timed_work< HOOKS >

  DEPRECATED.
- 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.

12.100.1 Detailed Description

Server-Side framework for implementing object-oriented servers.

´

12.100.2 Enumeration Type Documentation

12.100.2.1 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.

12.101 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.

12.101.1 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).

12.101.2 Function Documentation

12.101.2.1 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

12.101.2.2 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.

12.101.2.3 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.

12.101.2.4 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.

12.101.2.5 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.

12.101.2.6 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

12.101.2.7 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.

12.101.2.8 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.

12.102 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_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 pager 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.
- int l4sigma0_map_tbuf (l4_cap_idx_t sigma0, l4_addr_t virt)

  Request Fiasco trace buffer.
- 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 a user readable error messages for the return codes.

12.102.1 Detailed Description

Sigma0 API bindings.

Convenience bindings for the Sigma0 protocol.

12.102.2 Enumeration Type Documentation

12.102.2.1 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 81 of file sigma0.h.

12.102.3 Function Documentation

12.102.3.1 l4sigma0_debug_dump()

```
void l4sigma0_debug_dump (
            l4_cap_idx_t sigma0 )
```

Request sigma0 to dump internal debug information.

The debug information, such as internal memory maps, as well as statistics about the internal allocators is dumped to the kernel debugger.

Parameters

sigma0	the sigma0 thread id.

12.102.3.2 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.

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().

Parameters

sigma0	usually the thread id of sigma0.
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.

Return values

base	physical address of the page received (i.e., the send base of the received mapping if any).

Parameters

sz	Size to map by the server, in 2^sz bytes.

Returns

0 on success, !0 else (see l4sigma0_map_errstr()).

12.102.3.3 l4sigma0_map_errstr()

```
char const * l4sigma0_map_errstr (
            int err )  [inline]
```

Get a 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 213 of file sigma0.h.

References EXTERN_C_END.

12.102.3.4 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 to pager implementing the Sigma0 protocol.
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.

12.102.3.5 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 pager 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.

12.102.3.6 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	ID of service talking the sigma0 protocol.
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.

Returns

0 on success, !0 else (see l4sigma0_map_errstr()).

12.102.3.7 l4sigma0_map_tbuf()

```
int l4sigma0_map_tbuf (
            l4_cap_idx_t sigma0,
            l4_addr_t virt )
```

Request Fiasco trace buffer.

This is a Fiasco specific feature. Where you can request the kernel internal trace buffer for user-level evaluation. This is for special debugging tools, such as Ferret.

Parameters

sigma0	as usual the sigma0 thread id.
virt	the virtual address where the trace buffer should be mapped,

Returns

0 on success, !0 else (see l4sigma0_map_errstr()).

12.102.3.8 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 sigma0 gate.
gate	Capability selector to use for the new gate.

12.103 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.

12.103.1 Detailed Description

12.103.2 Function Documentation

12.103.2.1 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.

12.103.2.2 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.

12.103.2.3 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.

12.103.2.4 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.

12.103.2.5 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.

12.103.2.6 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.

12.104 Small C++ Template Library

Namespaces

- cxx

  Our C++ 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.

12.104.1 Detailed Description

12.104.2 Function Documentation

12.104.2.1 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< Ds_data >::init(), cxx::List_alloc::List_alloc(), and L4::Type_info::Demand::operator|().

Here is the caller graph for this function:

12.104.2.2 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 L4virtio::Svr::Data_buffer::copy_to(), cxx::List_alloc::List_alloc(), operator>>(), L4virtio::Svr::Data↩_buffer::skip(), and cxx::String::substr().

Here is the caller graph for this function:

12.104.2.3 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.

12.105 Task

C interface of the Task kernel object.

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_map (l4_cap_idx_t dst_task, l4_cap_idx_t src_task, l4_fpage_t snd_fpage, l4_addr_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, unsigned long 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 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_has_child (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW

  Test whether a capability has child mappings (in another task).
- 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_NOTHR↩OW

  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.

12.105.1 Detailed Description

C interface of the Task kernel object.

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>
```

12.105.2 Enumeration Type Documentation

12.105.2.1 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. **See also** L4::Task::unmap() l4_task_unmap()
L4_FP_DELETE_OBJ	Flag that indicates that the unmap operation on a capability shall try to delete the corresponding objects immediately. **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.

12.105.3 Function Documentation

12.105.3.1 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 436 of file task.h.

12.105.3.2 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 429 of file task.h.

12.105.3.3 l4_task_cap_has_child()

```
l4_msgtag_t l4_task_cap_has_child (
            l4_cap_idx_t task,
            l4_cap_idx_t cap )  [inline]
```

Test whether a capability has child mappings (in another task).

Parameters

task	Capability selector of the destination task to do the lookup in
cap	Capability selector to look up in the destination task

Returns

label contains 1 if it has at least one child, 0 if not or invalid

Deprecated Do not use. Undetermined future, might be removed.

Definition at line 423 of file task.h.

12.105.3.4 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	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.

Definition at line 417 of file task.h.

12.105.3.5 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

The object will be deleted if the obj has sufficient rights. No error will be reported if the rights are insufficient, however, the capability is removed in all cases.

This operation calls l4_task_unmap() with L4_FP_DELETE_OBJ.

Definition at line 396 of file task.h.

12.105.3.6 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_addr_t snd_base )  [inline]
```

Map resources available in the source task to a destination task.

Parameters

dst_task	Capability selector of destination task
src_task	Capability selector of 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.

Returns

Syscall return tag

This method allows for asynchronous rights delegation from one task to another. It can be used to share memory as well as to delegate access to objects.

Definition at line 366 of file task.h.

Referenced by L4::Cap_base::copy(), and L4::Cap_base::move().

Here is the caller graph for this function:

12.105.3.7 l4_task_release_cap()

```
l4_msgtag_t l4_task_release_cap (
            l4_cap_idx_t task,
            l4_cap_idx_t cap )  [inline]
```

Release capability.

Parameters

task	Capability selector of destination task
cap	Capability selector to release

Returns

Syscall return tag

This operation unmaps the capability from the specified task.

Definition at line 411 of file task.h.

12.105.3.8 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 the address space or object 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 and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Note

Calling this function on the object space can cause a root capability of an object to be destructed, which destroys the object itself.

Definition at line 373 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:

12.105.3.9 l4_task_unmap_batch()

```
l4_msgtag_t l4_task_unmap_batch (
            l4_cap_idx_t task,
            l4_fpage_t const * fpages,
            unsigned num_fpages,
            unsigned long map_mask )  [inline]
```

Revoke rights from a task.

Parameters

task	Capability selector of destination task
fpages	An array of flexpages that describes an area in the address space or object space of the destination task each
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

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Precondition

The caller needs to take care that num_fpages is not bigger than L4_UTCB_GENERIC_DATA_SIZE - 2.

Note

Calling this function on the object space can cause a root capability of an object to be destructed, which destroys the object itself.

Definition at line 380 of file task.h.

12.106 Thread

Thread object.

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_EXCEPTION = 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 or disable 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 or disable 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 or disable 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 or disable 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 modifiction 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.

12.106.1 Detailed Description

Thread object.

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.

12.106.2 Enumeration Type Documentation

12.106.2.1 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 640 of file thread.h.

12.106.2.2 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 663 of file thread.h.

12.106.2.3 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 678 of file thread.h.

12.106.3 Function Documentation

12.106.3.1 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.

12.106.3.2 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 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`).

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 829 of file thread.h.

References l4_thread_ex_regs_u(), and l4_utcb().

Here is the call graph for this function:

12.106.3.3 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`).

Returned values are valid only if function returns successfully.

Definition at line 836 of file thread.h.

References l4_thread_ex_regs_ret_u(), and l4_utcb().

Here is the call graph for this function:

12.106.3.4 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`).

Definition at line 702 of file thread.h.

Referenced by L4::Thread::ex_regs(), and l4_thread_ex_regs_ret().

Here is the caller graph for this function:

12.106.3.5 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 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`).

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 691 of file thread.h.

Referenced by L4::Thread::ex_regs(), and l4_thread_ex_regs().

Here is the caller graph for this function:

12.106.3.6 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) { label = (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 1009 of file thread.h.

12.106.3.7 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.

See also

l4_thread_modify_sender_start
l4_thread_modify_sender_add

Definition at line 1020 of file thread.h.

12.106.3.8 l4_thread_modify_sender_start()

```
l4_msgtag_t l4_thread_modify_sender_start (
            void ) [inline]
```

Start a thread sender modifiction 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().

See also

l4_thread_modify_sender_add
l4_thread_modify_sender_commit

Definition at line 1003 of file thread.h.

12.106.3.9 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.

An example of a deletion event is the removal of an IPC gate that is bound to this thread.

Definition at line 930 of file thread.h.

12.106.3.10 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 898 of file thread.h.

12.106.3.11 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 889 of file thread.h.

12.106.3.12 l4_thread_vcpu_control()

```
l4_msgtag_t l4_thread_vcpu_control (
            l4_cap_idx_t thread,
            l4_addr_t vcpu_state )  [inline]
```

Enable or disable the vCPU feature for the thread.

Parameters

thread	Capability selector of the thread for which the vCPU feature shall be enabled or disabled.
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` if `vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `vcpu_state` is 0. (Disable: optional, currently unsupported.)

Definition at line 947 of file thread.h.

References l4_thread_vcpu_control_u(), and l4_utcb().

Here is the call graph for this function:

12.106.3.13 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 or disable the extended vCPU feature for the thread.

Parameters

thread	Capability selector of the thread for which the extended vCPU feature shall be enabled or disabled.
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

Systemcall result message 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` if `ext_vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `ext_vcpu_state` is 0.

Note

The extended vCPU mode includes the normal vCPU mode.

Definition at line 962 of file thread.h.

References l4_thread_vcpu_control_ext_u(), and l4_utcb().

Here is the call graph for this function:

12.106.3.14 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 or disable the extended vCPU feature for the thread.

Parameters

thread	Capability selector of the thread for which the extended vCPU feature shall be enabled or disabled.
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 if `ext_vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `ext_vcpu_state` is 0.

Note

The extended vCPU mode includes the normal vCPU mode.

Definition at line 952 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:

12.106.3.15 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 or disable the vCPU feature for the thread.

Parameters

thread	Capability selector of the thread for which the vCPU feature shall be enabled or disabled.
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.

This function enables the vCPU feature of `this` thread if `vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `vcpu_state` is 0. (Disable: optional, currently unsupported.)

Definition at line 937 of file thread.h.

Referenced by l4_thread_vcpu_control(), and L4::Thread::vcpu_control().

Here is the caller graph for this function:

12.106.3.16 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 910 of file thread.h.

12.106.3.17 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 904 of file thread.h.

12.106.3.18 l4_thread_yield()

```
l4_msgtag_t l4_thread_yield (
            void ) [inline]
```

Yield current time slice.

Returns

system call return tag

Definition at line 778 of file thread.h.

References L4_INVALID_CAP, and l4_ipc_receive().

Here is the call graph for this function:

12.107 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.

12.107.1 Detailed Description

12.108 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.

12.108.1 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);
```

12.108.2 Function Documentation

12.108.2.1 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.

Alien mode means the thread is not allowed to invoke L4 kernel objects directly and it is also not allowed to allocate FPU state. All those operations result in an exception IPC that gets sent through the pager capability. The responsible pager can then selectively allow an object invocation or allocate FPU state for the thread.

This feature can be used to attach a debugger to a thread and trace all object invocations.

Examples:

examples/sys/aliens/main.c, and examples/sys/singlestep/main.c.

Definition at line 868 of file thread.h.

12.108.2.2 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 address of the UTCB in the target task.
task	The target task of the thread.

Binding a thread to a task has the effect that the thread afterwards executes code within that task and has access to the resources visible within that task.

Note

There should not be more than one thread use a UTCB 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 862 of file thread.h.

12.108.2.3 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 880 of file thread.h.

12.108.2.4 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 855 of file thread.h.

12.108.2.5 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 849 of file thread.h.

12.108.2.6 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 843 of file thread.h.

12.108.2.7 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 874 of file thread.h.

12.109 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, init
- #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.

Enumerations

- enum l4_timeout_abs_validity

  Intervals of validity for absolute timeouts
  Times are actually 2^x values (e.g.

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.

12.109.1 Detailed Description

All kinds of timeouts and time related functions.

12.109.2 Macro Definition Documentation

12.109.2.1 L4_IPC_TIMEOUT_0

```
#define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})
```

Timeout constants.

0 timeout

Definition at line 77 of file __timeout.h.

Referenced by L4::Ipc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >::timeout().

12.109.3 Typedef Documentation

12.109.3.1 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).

The timeout can also specify an absolute point in time (bit 16 == 1).

12.109.3.2 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.

12.109.4 Enumeration Type Documentation

12.109.4.1 l4_timeout_abs_validity

```
enum l4_timeout_abs_validity
```

Intervals of validity for absolute timeouts

Times are actually 2^x values (e.g.

2ms -> 2048µs)

Definition at line 92 of file __timeout.h.

12.109.5 Function Documentation

12.109.5.1 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 210 of file __timeout.h.

References l4_timeout_t::p, l4_timeout_t::rcv, l4_timeout_t::snd, and l4_timeout_s::t.

12.109.5.2 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

Return values

to	L4 timeout

Definition at line 238 of file __timeout.h.

12.109.5.3 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

Return values

to	L4 timeout

Definition at line 231 of file __timeout.h.

12.109.5.4 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 221 of file __timeout.h.

References l4_timeout_t::p, l4_timeout_t::rcv, l4_timeout_t::snd, and l4_timeout_s::t.

Referenced by L4::Ipc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >::timeout().

Here is the caller graph for this function:

12.109.5.5 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< Br_manager_timeout_hooks, Br_manager >::timeout().

Here is the caller graph for this function:

12.109.5.6 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 268 of file __timeout.h.

References l4_timeout_is_absolute(), and l4_timeout_rel_get().

Here is the call graph for this function:

12.109.5.7 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 261 of file __timeout.h.

References l4_timeout_s::t.

Referenced by l4_timeout_get().

Here is the caller graph for this function:

12.109.5.8 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 245 of file __timeout.h.

12.109.5.9 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 252 of file __timeout.h.

References l4_timeout_s::t.

Referenced by l4_timeout_get().

Here is the caller graph for this function:

12.109.5.10 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.

References __END_DECLS.

12.110 Timestamp Counter

Collaboration diagram for Timestamp Counter:

Files

- file rdtsc.h

  time stamp counter related functions
- file rdtsc.h

  time stamp counter related functions

Macros

- #define L4_TSC_INIT_AUTO 0

  Automatic init.
- #define L4_TSC_INIT_KERNEL 1

  Initialized by kernel.
- #define L4_TSC_INIT_CALIBRATE 2

  Initialized by user-level.
- #define L4_TSC_INIT_AUTO 0

  Automatic init.
- #define L4_TSC_INIT_KERNEL 1

  Initialized by kernel.
- #define L4_TSC_INIT_CALIBRATE 2

  Initialized by user-level.

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_cpu_time_t l4_rdpmc (int nr)

  Return current value of CPU-internal performance measurement counter.
- l4_uint32_t l4_rdpmc_32 (int nr)

  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)

  Calibrate scalers for time stamp calculations.
- l4_uint32_t l4_tsc_init (int constraint, l4_kernel_info_t *kip)

  Initialitze scaler for TSC calicaltions.
- l4_uint32_t l4_get_hz (void)

  Get CPU frequency in Hz.

12.110.1 Detailed Description

12.110.2 Function Documentation

12.110.2.1 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 intendet for any use!

Definition at line 317 of file rdtsc.h.

References l4_ns_to_tsc(), and l4_rdtsc().

Here is the call graph for this function:

12.110.2.2 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 intendet for any use!

Definition at line 327 of file rdtsc.h.

References EXTERN_C_END, l4_ns_to_tsc(), and l4_rdtsc().

Here is the call graph for this function:

12.110.2.3 l4_calibrate_tsc()

```
l4_uint32_t l4_calibrate_tsc (
            l4_kernel_info_t * kip )  [inline]
```

Calibrate scalers for time stamp calculations.

Determine some scalers to be able to convert between real time and CPU ticks. This test uses channel 0 of the PIT (i8254) or the kernel KIP, depending on availability. Just calls l4_tsc_init(L4_TSC_INIT_AUTO).

Examples:

examples/sys/aliens/main.c.

Definition at line 179 of file rdtsc.h.

References l4_tsc_init(), and L4_TSC_INIT_AUTO.

Here is the call graph for this function:

12.110.2.4 l4_get_hz()

```
l4_uint32_t l4_get_hz (
            void  )
```

Get CPU frequency in Hz.

Returns

frequency in Hz

12.110.2.5 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 303 of file rdtsc.h.

Referenced by l4_busy_wait_ns(), and l4_busy_wait_us().

Here is the caller graph for this function:

12.110.2.6 l4_rdpmc()

```
l4_cpu_time_t l4_rdpmc (
            int nr ) [inline]
```

Return current value of CPU-internal performance measurement counter.

Parameters

nr	Number of counter (0 or 1)

Returns

64-bit PMC

Definition at line 205 of file rdtsc.h.

12.110.2.7 l4_rdpmc_32()

```
l4_uint32_t l4_rdpmc_32 (
            int nr )  [inline]
```

Return the least significant 32 bit of a performance counter.

Useful for smaller differences, needs less cycles.

Definition at line 227 of file rdtsc.h.

12.110.2.8 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 185 of file rdtsc.h.

Referenced by l4_busy_wait_ns(), and l4_busy_wait_us().

Here is the caller graph for this function:

12.110.2.9 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 246 of file rdtsc.h.

12.110.2.10 l4_tsc_init()

```
l4_uint32_t l4_tsc_init (
            int constraint,
            l4_kernel_info_t * kip )
```

Initialitze scaler for TSC calicaltions.

Initialize the scalers needed by l4_tsc_to_ns()/l4_ns_to_tsc() and so on. Current versions of Fiasco export these scalers from kernel into userland. The programmer may decide whether he allows to use these scalers or if an calibration should be performed.

Parameters

constraint	programmers constraint: • L4_TSC_INIT_AUTO if the kernel exports the scalers then use them. If not, perform calibration using channel 0 of the PIT (i8254). The latter case may lead into short (unpredictable) periods where interrupts are disabled. • L4_TSC_INIT_KERNEL depend on retrieving the scalers from kernel. If the scalers are not available, return 0. • L4_TSC_INIT_CALIBRATE Ignore possible scalers exported by the scaler, instead insist on calibration using the PIT.
kip	KIP pointer

Returns

0 on error (no scalers exported by kernel, calibrating failed ...) 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:

12.110.2.11 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 260 of file rdtsc.h.

12.110.2.12 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 288 of file rdtsc.h.

12.110.2.13 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 274 of file rdtsc.h.

12.111 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
- Machine Restarting Function
- 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)

12.111.1 Detailed Description

Utilities, generic file.

12.111.2 Function Documentation

12.111.2.1 l4_sleep()

```
void l4_sleep (
            int ms )
```

Suspend thread for a period of *ms* milliseconds.

Parameters

ms	Time in milliseconds

12.111.2.2 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().

Referenced by l4_sleep_forever().

Here is the call graph for this function:

Here is the caller graph for this function:

12.111.2.3 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 EXTERN_C_END, L4_PAGESIZE, and l4_trunc_page().

Referenced by l4_sleep_forever().

Here is the call graph for this function:

Here is the caller graph for this function:

12.111.2.4 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!

12.111.2.5 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

12.111.2.6 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:

12.111.2.7 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:

12.112 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.

12.112.1 Detailed Description

Virtual machine API for SVM.

12.113 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

12.113.1 Detailed Description

Virtual Machine API for ARM TrustZone.

12.114 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_CTLS_REG = 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_FIXE↩D0_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_ENU↩M_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.

12.114.1 Detailed Description

Virtual machine API for VMX.

12.114.2 Enumeration Type Documentation

12.114.2.1 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.

12.114.2.2 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.

12.114.2.3 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.

12.114.3 Function Documentation

12.114.3.1 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:

12.114.3.2 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_NOTHROW, and l4_vm_vmx_field_order().

Here is the call graph for this function:

12.114.3.3 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:

12.114.3.4 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.

12.114.3.5 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.

12.114.3.6 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.

12.114.3.7 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:

12.114.3.8 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:

12.114.3.9 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:

12.114.3.10 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:

12.114.3.11 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:

12.114.3.12 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:

12.114.3.13 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:

12.114.3.14 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:

12.114.3.15 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:

12.114.3.16 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:

12.114.3.17 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:

12.115 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.

12.115.1 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>
```

Refer to L4 Vbus functions for the C API.

12.116 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_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.

12.116.1 Detailed Description

12.116.2 Typedef Documentation

12.116.2.1 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.

12.116.3 Enumeration Type Documentation

12.116.3.1 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.

12.116.3.2 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.

12.116.4 Function Documentation

12.116.4.1 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

12.116.4.2 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.

12.116.4.3 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())

12.116.4.4 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.

12.116.4.5 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.

12.116.4.6 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().

12.116.4.7 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

12.116.4.8 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

12.116.4.9 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.

12.116.4.10 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

12.116.4.11 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.

12.116.4.12 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.

12.116.4.13 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.

12.117 Virtual Console

Virtual console for simple character based input and output.

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.

12.117.1 Detailed Description

Virtual console for simple character based input and output.

The interrupt for read events is provided by the virtual key interrupt.

Include File

```
#include <l4/sys/vcon.h>
```

See L4::Vcon for the C++ interface.

12.117.2 Enumeration Type Documentation

12.117.2.1 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 189 of file vcon.h.

12.117.2.2 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 211 of file vcon.h.

12.117.2.3 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 200 of file vcon.h.

12.117.2.4 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 95 of file vcon.h.

12.117.3 Function Documentation

12.117.3.1 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 409 of file vcon.h.

References l4_utcb(), and l4_vcon_get_attr_u().

Here is the call graph for this function:

12.117.3.2 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 successfull call of this function.
	utcb	UTCB pointer of the calling thread.

Returns

Syscall return tag.

Definition at line 391 of file vcon.h.

Referenced by L4::Vcon::get_attr(), and l4_vcon_get_attr().

Here is the caller graph for this function:

12.117.3.3 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 365 of file vcon.h.

References l4_utcb(), and l4_vcon_read_u().

Here is the call graph for this function:

12.117.3.4 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.

Return values

>*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 355 of file vcon.h.

Referenced by l4_vcon_read(), and L4::Vcon::read().

Here is the caller graph for this function:

12.117.3.5 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 349 of file vcon.h.

12.117.3.6 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.

Definition at line 289 of file vcon.h.

References l4_utcb(), and l4_vcon_send_u().

Here is the call graph for this function:

12.117.3.7 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 275 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:

12.117.3.8 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 385 of file vcon.h.

References l4_utcb(), and l4_vcon_set_attr_u().

Here is the call graph for this function:

12.117.3.9 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 371 of file vcon.h.

Referenced by l4_vcon_set_attr(), and L4::Vcon::set_attr().

Here is the caller graph for this function:

12.117.3.10 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 310 of file vcon.h.

References l4_utcb(), and l4_vcon_write_u().

Here is the call graph for this function:

12.117.3.11 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 295 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:

12.118 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.

12.118.1 Detailed Description

Virtual Machine API.

12.119 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.

12.119.1 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.

12.119.2 Typedef Documentation

12.119.2.1 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 67 of file utcb.h.

12.119.3 Function Documentation

12.119.3.1 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:

12.119.3.2 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:

12.119.3.3 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.

12.120 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.

12.120.1 Detailed Description

12.121 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_DEBUG_EXC = 0x08,
  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.

12.121.1 Detailed Description

vCPU API

12.121.2 Enumeration Type Documentation

12.121.2.1 L4_vcpu_state_flags

`enum L4_vcpu_state_flags`

State flags of a vCPU.

Enumerator

L4_VCPU_F_IRQ	IRQs (events) enabled.
L4_VCPU_F_PAGE_FAULTS	Page faults enabled.
L4_VCPU_F_EXCEPTIONS	Exception enabled.
L4_VCPU_F_DEBUG_EXC	Debug exception enabled.
L4_VCPU_F_USER_MODE	User task will be used.
L4_VCPU_F_FPU_ENABLED	FPU enabled.

Definition at line 58 of file vcpu.h.

12.121.2.2 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 81 of file vcpu.h.

12.121.2.3 L4_vcpu_sticky_flags

`enum L4_vcpu_sticky_flags`

Sticky flags of a vCPU.

Enumerator

L4_VCPU_SF_IRQ_PENDING	An event (e.g. IRQ) is pending.

Definition at line 72 of file vcpu.h.

12.122 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.

12.122.1 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.

12.122.2 Function Documentation

12.122.2.1 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(), L4_CV, 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:

12.122.2.2 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 l4_vcpu_state_t::i, L4_CV, l4_ipc_wait(), L4_LIKELY, l4_msgtag_has_error(), L4_NOTHROW, and 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:

12.122.2.3 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_CV, L4_IPC_BOTH_TIMEOUT_0, L4_LIKELY, L4_VCPU_F_IRQ, L4_VCPU_SF_IRQ_PENDING, l4_vcpu_state_t::state, and l4_vcpu_state_t::sticky_flags.

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:

12.122.2.4 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_CV, L4_VCPU_F_IRQ, l4vcpu_irq_disable(), l4vcpu_irq_enable(), and l4_vcpu_state_t::state.

Referenced by L4vcpu::Vcpu::irq_restore().

Here is the call graph for this function:

Here is the caller graph for this function:

12.122.2.5 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:

12.122.2.6 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:

12.122.2.7 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:

12.122.2.8 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 __END_DECLS, and L4_IPC_NEVER.

Referenced by L4vcpu::Vcpu::wait_for_event().

Here is the caller graph for this function:

12.123 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.

12.123.1 Detailed Description

12.123.2 Enumeration Type Documentation

12.123.2.1 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 13

Namespace Documentation

13.1 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

  A generic AVL tree.
- class Avl_tree_node

  Node of an AVL tree.
- class Base_slab

  Basic slab allocator.
- class Base_slab_static

  Merged slab allocator (allocators for objects of the same size are merged together).
- class Bitfield

  Definition for a member (part) of a bit field.
- class Bitmap

  A static bit map.
- class Bitmap_base

  Basic bitmap abstraction.
- class H_list

  General double-linked list of unspecified cxx::H_list_item elements.

- class H_list_item_t

  Basic element type for a double-linked H_list.
- struct H_list_t

  Double-linked list of typed H_list_item_t elements.
- class List

  Doubly linked list, with internal allocation.
- class List_alloc

  Standard list-based allocator.
- class List_item

  Basic list item.
- struct Lt_functor

  Generic comparator class that defaults to the less-than operator.
- class New_allocator

  Standard allocator based on `operator new ()` *.*
- class Nothrow

  Helper type to distinguish the `oeprator new` *version that does not throw exceptions.*
- struct Pair

  Pair of two values.
- class Pair_first_compare

  Comparison functor for Pair.
- struct Ref_obj_list_item

  Item for list linked via cxx::Ref_ptr with default refence counting.
- class Ref_ptr

  A reference-counting pointer with automatic cleanup.
- class S_list

  Simple single-linked list.
- class Slab

  Slab allocator for object of type Type.
- class Slab_static

  Merged slab allocator (allocators for objects of the same size are merged together).
- class static_vector

  Simple encapsulation for a dynamically allocated array.
- class String

  Allocation free string class with explicit length field.
- class Weak_ref

  Typed weak reference to an object of type `T`*.*
- class Weak_ref_base

  Generic (base) weak reference to some object.

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::unqiue_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.

13.1.1 Detailed Description

Our C++ library.

Small Low-Level C++ Library.

Strings.

Various kinds of C++ utilities.

13.1.2 Typedef Documentation

13.1.2.1 H_list_item

```
typedef H_list_item_t<void> cxx::H_list_item
```

Untyped list item.

Definition at line 72 of file hlist.

13.2 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 Basic_list

  Internal: Common functions for all head-based list implementations.
- class Bst

  Basic binary search tree (BST).
- class Bst_node

  Basic type of a node in a binary search tree (BST).
- struct Direction

  The direction to go in a binary search tree.
- class Smart_ptr_list

  List of smart-pointer-managed objects.
- class Smart_ptr_list_item

  List item for an arbitrary item in a Smart_ptr_list.

13.2.1 Detailed Description

Internal helpers for the cxx package.

13.3 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

- class Alloc_list

  A simple list-based allocator.
- class Arm_smccc

  Wrapper for function calls that follow the ARM SMC/HVC calling convention.
- class Base_exception

  Base class for all exceptions, thrown by the L4Re framework.
- class Basic_registry

  This registry returns the corresponding server object based on the label of an Ipc_gate.
- class Bounds_error

  Access out of bounds.
- class Cap

  C++ interface for capabilities.
- class Cap_base

  Base class for all kinds of capabilities.
- class Com_error

  Error conditions during IPC.
- class Debugger

  C++ debugger interface.
- class Element_already_exists

  Exception for duplicate element insertions.
- class Element_not_found

  Exception for a failed lookup (element not found).
- class Exception

  Exception interface.
- class Exception_tracer

  Back-trace support for exceptions.
- class Factory

  C++ Factory interface to create kernel objects.
- class Icu

  C++ Icu interface.
- class Invalid_capability

  Indicates that an invalid object was invoked.
- class Io_pager

  Io_pager interface.
- class Iommu

  Interface for IO-MMUs used for DMA remapping.
- class IOModifier

  Modifier class for the IO stream.
- class Ipc_gate

  The C++ IPC gate interface.
- class Irq

  C++ Irq interface.
- class Irq_eoi

  Interface for sending an acknowledge message to an object.
- struct Irq_handler_object

  Server object base class for handling IRQ messages.
- struct Irq_mux

  IRQ multiplexer for shared IRQs.
- class Kobject

Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

- 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...
- class Kobject_t

  Helper class to create an L4Re interface class that is derived from a single base class.
- struct Kobject_typeid

  Meta object for handling access to type information of Kobjects.
- struct Kobject_typeid< void >

  Minimalistic ID for `void` interface.
- struct Kobject_x

  Generic Kobject inheritance template.
- class Meta

  Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects.
- class Out_of_memory

  Exception signalling insufficient memory.
- class Pager

  Pager interface including the Io_pager interface.
- class Platform_control

  L4 C++ interface for controlling platform-wide properties.
- class Poll_timeout_kipclock

  A polling timeout based on the L4Re clock.
- struct Proto_t

  Data type for defining protocol numbers.
- class Rcv_endpoint

  Interface for kernel objects that allow to receive IPC from them.
- class Registry_iface

  Abstract interface for object registries.
- class Runtime_error

  Exception for an abstract runtime error.
- class Scheduler

  C++ interface of the Scheduler kernel object.
- struct Semaphore

  Kernel-provided semaphore object.
- class Server

  Basic server loop for handling client requests.
- 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.
- class Smart_cap

  Smart capability class.
- class String

  A null-terminated string container class.
- class Task

C++ interface of the Task kernel object.

- class Thread

  C++ L4 kernel thread interface.

- struct Triggerable

  Interface that allows an object to be triggered by some source.

- struct Type_info

  Dynamic Type Information for L4Re Interfaces.

- class Unknown_error

  Exception for an unknown condition.

- class Vcon

  C++ L4 Vcon interface.

- class Vm

  Virtual machine.

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 ()

  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) throw ()

  `dynamic_cast` *for capabilities.*

- template<typename T , typename F >
  Cap< T > cap_cast (Cap< F > const &c) throw ()

  static_cast for capabilities.

- template<typename T , typename F >
  Cap< T > cap_reinterpret_cast (Cap< F > const &c) throw ()

  reinterpret_cast for capabilities.

- template<typename T , typename F , typename SMART >
  Smart_cap< T, SMART > cap_cast (Smart_cap< F, SMART > const &c) throw ()

  `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) throw ()

  `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.

13.3.1 Detailed Description

L4 low-level kernel interface.

13.3.2 Enumeration Type Documentation

13.3.2.1 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.

13.3.3 Function Documentation

13.3.3.1 cap_cast() [1/2]

```
template<typename T , typename F , typename SMART >
Smart_cap<T, SMART> L4::cap_cast (
            Smart_cap< F, SMART > const & c ) throw )   [inline]
```

`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.

13.3.3.2 cap_cast() [2/2]

```
template<typename T , typename F >
Cap<T> L4::cap_cast (
            Cap< F > const & c ) throw )   [inline]
```

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 377 of file capability.h.

References L4::Cap_base::cap().

Referenced by L4Re::Cap_alloc::alloc(), L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc(), L4::Ipc_svr::Server_iface::rcv_cap(), and L4virtio::Svr::Block_dev< Ds_data >::register_obj().

Here is the call graph for this function:

Here is the caller graph for this function:

13.3.3.3 cap_dynamic_cast()

```
template<typename T , typename F >
Cap<T> L4::cap_dynamic_cast (
            Cap< F > const & c ) throw )   [inline]
```

`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 119 of file capability.

References cap_reinterpret_cast(), l4_error(), and L4::Meta::supports().

Here is the call graph for this function:

13.3.3.4 cap_reinterpret_cast() [1/2]

```
template<typename T , typename F , typename SMART >
Smart_cap<T, SMART> L4::cap_reinterpret_cast (
            Smart_cap< F, SMART > const & c ) throw )   [inline]
```

`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.

13.3.3.5 cap_reinterpret_cast() [2/2]

```
template<typename T , typename F >
Cap<T> L4::cap_reinterpret_cast (
            Cap< F > const & c ) throw )   [inline]
```

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 408 of file capability.h.

References L4::Cap_base::cap().

Referenced by cap_dynamic_cast().

Here is the call graph for this function:

Here is the caller graph for this function:

13.3.3.6 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 44 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:

13.3.3.7 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 61 of file ipc_helper.

References throw_ipc_exception().

Here is the call graph for this function:

13.4 L4::Ipc Namespace Reference

IPC related functionality.

Namespaces

- Msg

  IPC Message related functionality.

Data Structures

- struct Array

  Array data type for dynamically sized arrays in RPCs.
- struct Array_in_buf

  Server-side copy in buffer for Array.
- struct Array_ref

  Array reference data type for arrays located in the message.
- struct As_value

  Pass the argument as plain data value.
- class Buf_item

  RPC warpper for a receive item.
- struct Call

  RPC attribute for a standard RPC call.
- struct Call_t

  RPC attribute for an RPC call with required rights.
- struct Call_zero_send_timeout

  RPC attribute for an RPC call, with zero send timeout.
- class Cap

  Capability type for RPC interfaces (see `L4::Cap<T>`).
- class Gen_fpage

  Generic RPC wrapper for L4 flex-pages.
- struct In_out

  Mark an argument as in-out argument.
- class Iostream

  Input/Output stream for IPC [un]marshalling.
- class Istream

  Input stream for IPC unmarshalling.
- class Msg_ptr

  Pointer to an element of type T in an lpc::Istream.
- struct Opt

  Attribute for defining an optional RPC argument.
- class Ostream

  Output stream for IPC marshalling.
- struct Out

  Mark an argument as a output value in an RPC signature.
- struct Ret_array

  Dynamically sized output array of type T.
- struct Send_only

  RPC attribute for a send-only RPC.
- class Small_buf

  A receive item for receiving a single capability.
- class Snd_item

  RPC wrapper for a send item.
- class Str_cp_in

  Abstraction for extracting a zero-terminated string from an lpc::Istream.
- 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.

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)

  Make an L4::Ipc::Cap<T> for the given capability and rights.
- template<typename T >
  Cap< T > make_cap_rw (L4::Cap< T > cap)

  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)

  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)

  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.

13.4.1 Detailed Description

IPC related functionality.

13.4.2 Function Documentation

13.4.2.1 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 172 of file ipc_stream.

13.4.2.2 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 114 of file ipc_stream.

13.4.2.3 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.

13.4.2.4 make_cap()

```
template<typename T >
Cap<T> L4::Ipc::make_cap (
            L4::Cap< T > cap,
            unsigned rights )
```

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.

Referenced by L4virtio::Svr::Device_t< Ds_data >::mem_info().

Here is the caller graph for this function:

13.4.2.5 make_cap_full()

```
template<typename T >
Cap<T> L4::Ipc::make_cap_full (
            L4::Cap< T > cap )
```

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::Ipc::Cap< T >::cap(), L4::Ipc::Cap< T >::fpage(), L4::Cap_base::is_valid(), L4::Ipc::Cap< T >::is_valid(), L4_CAP_FPAGE_RWSD, L4_EMSGMISSARG, L4_FPAGE_C_OBJ_RIGHTS, and L4_UNLIKELY.

Here is the call graph for this function:

13.4.2.6 make_cap_rw()

```
template<typename T >
Cap<T> L4::Ipc::make_cap_rw (
            L4::Cap< T > cap )
```

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_ds/ds_srv.cc.

Definition at line 634 of file ipc_types.

References L4_CAP_FPAGE_RW.

Referenced by L4Re::Util::Event_buffer_t< PAYLOAD >::attach(), L4virtio::Svr::Dev_config::Dev_config(), L4Re::Util::Event_t< PAYLOAD >::init(), and L4Re::Util::Event_t< PAYLOAD >::init_poll().

Here is the caller graph for this function:

13.4.2.7 make_cap_rws()

```
template<typename T >
Cap<T> L4::Ipc::make_cap_rws (
            L4::Cap< T > cap )
```

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.

13.4.2.8 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 264 of file ipc_stream.

Referenced by L4::Ipc::Istream::get(), and operator<<().

Here is the caller graph for this function:

13.4.2.9 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 1392 of file ipc_stream.

13.4.2.10 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 225 of file ipc_stream.

13.5 L4::Ipc::Msg Namespace Reference

IPC Message related functionality.

Data Structures

- struct Clnt_val_ops

  Defines client-side handling of 'MTYPE' as RPC argument.
- struct Cls_buffer

  Marker type for receive buffer values.
- struct Cls_data

  Marker type for data values.
- struct Cls_item

  Marker type for item values.
- struct Dir_in

  Marker type for input values.
- struct Dir_out

  Marker type for output values.
- struct Do_in_data

  Marker for Input data.
- struct Do_in_items

  Marker for Input items.
- struct Do_out_data

  Marker for Output data.
- struct Do_out_items

  Marker for Output items.
- struct Do_rcv_buffers

  Marker for receive buffers.
- struct Elem< Array< A, LEN > &>

  Array as output argument.
- struct Elem< Array< A, LEN > >

  Array as input arguments.
- struct Elem< Array_ref< A, LEN > &>

  Array_ref as output argument.

- 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.
- struct Svr_val_ops

  Defines server-side handling for `MTYPE` server arguments.

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

- unsigned long align_to (unsigned long bytes, unsigned long align)

  Pad bytes to the given alignment align (in bytes)
- template<typename T >
  unsigned long align_to (unsigned long bytes)

  Pad bytes to the alignment of the type T.
- template<typename T >
  bool check_size (unsigned offset, unsigned limit)

  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)

  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)

  Add some data to a message at offs.
- template<typename T >
  int msg_get (char ∗msg, unsigned offs, unsigned limit, T &v)

  Get some data from a message at offs.

13.5.1 Detailed Description

IPC Message related functionality.

13.5.2 Enumeration Type Documentation

13.5.2.1 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.

13.5.3 Function Documentation

13.5.3.1 align_to() [1/2]

```
unsigned long L4::Ipc::Msg::align_to (
            unsigned long bytes,
            unsigned long align )  [inline]
```

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:

13.5.3.2 align_to() [2/2]

```
template<typename T >
unsigned long L4::Ipc::Msg::align_to (
            unsigned long bytes )  [inline]
```

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:

13.5.3.3 check_size() [1/2]

```
template<typename T >
bool L4::Ipc::Msg::check_size (
            unsigned offset,
            unsigned limit )  [inline]
```

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.

13.5.3.4 check_size() [2/2]

```
template<typename T , typename CTYPE >
bool L4::Ipc::Msg::check_size (
            unsigned offset,
            unsigned limit,
            CTYPE cnt )  [inline]
```

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.

13.5.3.5 msg_add()

```
template<typename T >
int L4::Ipc::Msg::msg_add (
```

```
            char * msg,
            unsigned offs,
            unsigned limit,
            T v )  [inline]
```

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.

13.5.3.6 msg_get()

```
template<typename T >
int L4::Ipc::Msg::msg_get (
            char * msg,
            unsigned offs,
            unsigned limit,
            T & v )  [inline]
```

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.

13.6 L4::Ipc_svr Namespace Reference

Helper classes for L4::Server instantiation.

Data Structures

- class Br_manager_no_buffers

  Empty implementation of Server_iface.
- struct Compound_reply

  Mix in for LOOP_HOOKS to always use compound reply and wait.
- struct Default_loop_hooks

  Default LOOP_HOOKS.
- struct Default_setup_wait

  Mix in for LOOP_HOOKS for setup_wait no op.
- struct Default_timeout

  Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.
- struct Direct_dispatch

  Direct disptach helper, for forwarding dispatch calls a registry R.
- struct Direct_dispatch< R * >

  Direct disptach helper, for forwarding dispatch calls via a pointer to a registry R.
- struct Exc_dispatch

  Dispatch helper wrapping try {} catch {} around the dispatch call.
- struct Ignore_errors

  Mix in for LOOP_HOOKS to ignore IPC errors.
- class Server_iface

  Interface for server-loop related functions.
- class Timed_work

  DEPRECATED.
- 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.

13.6.1 Detailed Description

Helper classes for L4::Server instantiation.

13.7 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 Rpc_nocode

  List of RPCs of an interface using a single operation without an opcode.
- struct Rpcs

  Standard list of RPCs of an interface.
- struct Rpcs_code

  List of RPCs of an interface using a special opcode type.
- struct Rpcs_sys

  List of RPCs typically used for kernel interfaces.

13.7.1 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>`.

13.8 L4::Types Namespace Reference

L4 basic type helpers for C++.

Data Structures

- struct Bool

  Boolean meta type.
- struct False

  False meta value.
- class Flags

  Template for defining typical Flags bitmaps.
- struct Same

  Compare two data types for equality.
- struct True

  True meta value.

13.8.1 Detailed Description

L4 basic type helpers for C++.

13.9 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 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.
- class Rm

  Region map.
- class Smart_cap_auto

  Helper for Auto_cap and Auto_del_cap.
- class Smart_count_cap

  Helper for Ref_cap and Ref_del_cap.

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

- 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.

13.9.1 Detailed Description

L4Re C++ Interfaces.

L4 Runtime Environment.

13.9.2 Typedef Documentation

13.9.2.1 Shared_cap

```
template<typename T >
using L4Re::Shared_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SP↩
ACES> >
```

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.

13.9.2.2 shared_cap

```
template<typename T >
using L4Re::shared_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SP↩
ACES> >
```

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.

13.9.2.3 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.

13.9.2.4 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.

13.9.2.5 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.

13.9.2.6 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.

13.9.2.7 Unique_del_cap

```
template<typename T >
using L4Re::Unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DEL↩
ETE_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.

13.9.2.8 unique_del_cap

```
template<typename T >
using L4Re::unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DEL↩
ETE_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.

13.9.3 Function Documentation

13.9.3.1 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 140 of file error_helper.

References L4_UNLIKELY.

Referenced by L4virtio::Svr::Dev_config::Dev_config(), L4virtio::Svr::Device_t< Ds_data >::handle_mem_cmd_write(), and L4virtio::Svr::Block_dev< Ds_data >::register_obj().

Here is the caller graph for this function:

13.9.3.2 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 171 of file error_helper.

References chksys(), l4_msgtag_t::has_error(), and L4_UNLIKELY.

Here is the call graph for this function:

13.9.3.3 chksys() [1/3]

```
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 63 of file error_helper.

References L4_UNLIKELY.

Referenced by L4virtio::Svr::Driver_mem_list_t< Ds_data >::add(), chkipc(), chksys(), L4virtio::Svr::Dev_config↩ ::Dev_config(), L4virtio::Svr::Driver_mem_region_t< Ds_data >::Driver_mem_region_t(), L4virtio::Svr::Device_t< Ds_data >::handle_mem_cmd_write(), L4virtio::Svr::Block_dev< Ds_data >::register_obj(), L4virtio::Svr::Driver↩ _mem_list_t< Ds_data >::remove(), and L4virtio::Svr::Device_t< Ds_data >::setup_queue().

Here is the caller graph for this function:

13.9.3.4 chksys() [2/3]

```
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 84 of file error_helper.

References l4_msgtag_t::has_error(), and L4_UNLIKELY.

Here is the call graph for this function:

13.9.3.5 chksys() [3/3]

```
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 107 of file error_helper.

References chksys(), and L4_UNLIKELY.

Here is the call graph for this function:

13.9.3.6 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:

13.9.3.7 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:

13.9.3.8 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:

13.9.3.9 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:

13.10 L4Re::Util Namespace Reference

Documentation of the L4 Runtime Environment utility functionality in C++.

Data Structures

- struct Auto_cap

  Automatic capability that implements automatic free and unmap of the capability selector.
- struct Auto_del_cap

  Automatic capability that implements automatic free and unmap+delete of the capability selector.
- 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 Cap_alloc_base

  Capability allocator.
- 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_buffer_consumer_t

An event buffer consumer.

- class Event_buffer_t

  Event_buffer utility class.
- class Event_t

  Convenience wrapper for getting access to an event object.
- 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.
- 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.
- class Registry_server

  A server loop object which has a Object_registry included.
- class Smart_cap_auto

  Helper for Auto_cap and Auto_del_cap.
- class Smart_count_cap

  Helper for Ref_cap and Ref_del_cap.
- 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 >
  Auto_cap< T >::Cap make_auto_cap ()

  Allocate a capability slot and wrap it in an Auto_cap.
- template<typename T >
  Auto_del_cap< T >::Cap make_auto_del_cap ()

  Allocate a capability slot and wrap it in an Auto_del_cap.
- 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()) throw ()

  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.

13.10.1 Detailed Description

Documentation of the L4 Runtime Environment utility functionality in C++.

13.10.2 Typedef Documentation

13.10.2.1 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.

13.10.2.2 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.

13.10.2.3 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.

13.10.2.4 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.

13.10.2.5 Unique_cap

```
template<typename T >
using L4Re::Util::Unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_A↩
LL_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 = 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.

13.10.2.6 unique_cap

```
template<typename T >
using L4Re::Util::unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_A↩
LL_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 = 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.

13.10.2.7 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.

13.10.2.8 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.

13.10.3 Function Documentation

13.10.3.1 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:

13.10.3.2 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:

13.10.3.3 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:

13.10.3.4 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:

13.11 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 Directory

  Interface for a POSIX file that is a directory.
- class File

  The basic interface for an open POSIX file.
- class File_system

  Basic interface for an L4Re::Vfs file system.
- class Fs

  POSIX File-system related functionality.
- class Generic_file

  The common interface for an open POSIX file.
- class Mman

  Interface for the POSIX memory management.
- class Ops

  Interface for the POSIX backends for an application.
- 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.

Functions

- L4Re::Vfs::Ops *vfs_ops asm ("l4re_env_posix_vfs_ops")

  Reference to the applications L4Re::Vfs::Ops singleton.

13.11.1 Detailed Description

Virtual file system for interfaces POSIX libc.

13.12 L4virtio Namespace Reference

L4-VIRTIO Transport C++ API.

Data Structures

- class Ptr

  Pointer used in virtio descriptors.
- class Virtqueue

  Low-level Virtqueue.

13.12.1 Detailed Description

L4-VIRTIO Transport C++ API.

Chapter 14

Data Structure Documentation

14.1 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) ...`.

14.1.1 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.

14.1.2 Member Typedef Documentation

14.1.2.1 Ref_type

```
template<typename T >
typedef T cxx::Auto_ptr< T >::Ref_type
```

The referenced type.

Definition at line 41 of file auto_ptr.

14.1.3 Constructor & Destructor Documentation

14.1.3.1 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.

14.1.3.2 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.

14.1.3.3 ~Auto_ptr()

```
template<typename T >
cxx::Auto_ptr< T >::~Auto_ptr ( ) throw )   [inline]
```

Destruction, shall delete the object.

Definition at line 76 of file auto_ptr.

14.1.4 Member Function Documentation

14.1.4.1 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.

14.1.4.2 operator Priv_type ∗()

```
template<typename T >
cxx::Auto_ptr< T >::operator Priv_type * ( ) const throw )   [inline]
```

Operator for `if (!ptr) ...`.

Definition at line 110 of file auto_ptr.

14.1.4.3 operator∗()

```
template<typename T >
T& cxx::Auto_ptr< T >::operator* ( ) const throw )   [inline]
```

Dereference the pointer.

Definition at line 80 of file auto_ptr.

14.1.4.4 operator->()

```
template<typename T >
T* cxx::Auto_ptr< T >::operator-> ( ) const throw )   [inline]
```

Member access for the object.

Definition at line 83 of file auto_ptr.

14.1.4.5 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:

14.1.4.6 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

14.2 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.

14.2.1 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.

14.2.2 Constructor & Destructor Documentation

14.2.2.1 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.

14.2.3 Member Function Documentation

14.2.3.1 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.

14.2.3.2 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 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.

14.2.3.3 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& 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.

The documentation for this class was generated from the following file:

- l4/cxx/avl_map

14.3 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

14.3.1 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 429 of file avl_set.

The documentation for this class was generated from the following file:

- l4/cxx/avl_set

14.4 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

Forward iterator for the tree.

- 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 ()

  Create an empty AVL tree.
- ~Avl_tree () noexcept

  Destroy the tree.

Additional Inherited Members

14.4.1 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 109 of file avl_tree.

14.4.2 Member Function Documentation

14.4.2.1 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 229 of file avl_tree.

Referenced by cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::insert().

Here is the caller graph for this function:

14.4.2.2 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 291 of file avl_tree.

Referenced by cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::remove().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/avl_tree

14.5 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 ()

  Create an uninitialized node, this is what you should do.

Additional Inherited Members

14.5.1 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

14.6 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 >:

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 allocator for the slab caches.

Public Member Functions

- unsigned total_objects () const throw ()

  Get the total number of objects managed by the slab allocator.
- unsigned free_objects () const throw ()

  Get the total number of objects managed by the slab allocator.

14.6.1 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.

Parameters

Obj_size	The size of the objects managed by the allocator (in bytes).
Slab_size	The size of a slab cache (in bytes).
Max_free	The maximum number of free slab caches. When this limit is reached slab caches are freed.
Alloc	The allocator that is used to allocate the slab caches.

Definition at line 40 of file slab_alloc.

14.6.2 Member Enumeration Documentation

14.6.2.1 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 cache.
objects_per_slab	objects per slab cache.
max_free_slabs	maximum number of free slab caches.

Definition at line 63 of file slab_alloc.

14.6.3 Member Function Documentation

14.6.3.1 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 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 271 of file slab_alloc.

Referenced by cxx::Base_slab_static< sizeof(Type), Slab_size, Max_free, Alloc >::free_objects().

Here is the caller graph for this function:

14.6.3.2 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 263 of file slab_alloc.

Referenced by cxx::Base_slab_static< sizeof(Type), 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

14.7 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.

14.7.1 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).

Parameters

Obj_size	The size of an object managed by the slab allocator.
Slab_size	The size of a slab cache.
Max_free	The maximum number of free slab caches.
Alloc	The allocator for the slab caches.

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 348 of file slab_alloc.

14.7.2 Member Enumeration Documentation

14.7.2.1 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 cache.
objects_per_slab	number of objects per slab cache.
max_free_slabs	maximum number of free slab caches.

Definition at line 355 of file slab_alloc.

14.7.3 Member Function Documentation

14.7.3.1 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.

Definition at line 365 of file slab_alloc.

14.7.3.2 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 371 of file slab_alloc.

14.7.3.3 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 slab caches managed by this allocator.

Note

The value is the merged value for all equal parameterized Base_slab_static instances.

Definition at line 389 of file slab_alloc.

14.7.3.4 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 380 of file slab_alloc.

The documentation for this class was generated from the following file:

- l4/cxx/slab_alloc

14.8 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.

14.8.1 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 mos significant bit if our bits.

Definition at line 34 of file bitfield.

14.8.2 Member Typedef Documentation

14.8.2.1 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 78 of file bitfield.

14.8.2.2 Ref

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Value<T&> cxx::Bitfield< T, LSB, MSB >::Ref
```

Reference type to access the bits inside a raw bit field.

Definition at line 218 of file bitfield.

14.8.2.3 Ref_unshifted

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Value_unshifted<T&> cxx::Bitfield< T, LSB, MSB >::Ref_unshifted
```

Reference type to access the bits inside a raw bit field (in place).

Definition at line 223 of file bitfield.

14.8.2.4 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 86 of file bitfield.

14.8.2.5 Val

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Value<T const> cxx::Bitfield< T, LSB, MSB >::Val
```

Value type to access the bits inside a raw bit field.

Definition at line 220 of file bitfield.

14.8.2.6 Val_unshifted

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Value_unshifted<T const> cxx::Bitfield< T, LSB, MSB >::Val_unshifted
```

Value type to access the bits inside a raw bit field (in place).

Definition at line 225 of file bitfield.

14.8.3 Member Enumeration Documentation

14.8.3.1 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 58 of file bitfield.

14.8.3.2 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 65 of file bitfield.

14.8.4 Member Function Documentation

14.8.4.1 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 100 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:

14.8.4.2 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 110 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Mask.

14.8.4.3 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 146 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Low_mask, cxx::Bitfield< T, LSB, MSB >::set_dirty(), and cxx::Bitfield< T, LSB, MSB >::val().

Here is the call graph for this function:

14.8.4.4 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 contain not more than bits than Bits.

Note

This function does not mask *val* to the right number of bits.

Definition at line 120 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Lsb, and cxx::Bitfield< T, LSB, MSB >::Mask.

Referenced by cxx::Bitfield< T, LSB, MSB >::set().

Here is the caller graph for this function:

14.8.4.5 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 bits.

Returns

the new value of the whole bit field.

Definition at line 155 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Mask, and cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty().

Here is the call graph for this function:

14.8.4.6 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 contain not more than bits than Bits shifted Lsb bits to the left.

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 >::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:

14.8.4.7 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 containing.

Definition at line 170 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 >::set(), and cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty().

Here is the call graph for this function:

Here is the caller graph for this function:

14.8.4.8 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 containing.

Precondition

val must contain not more than bits than Bits.

Note

This function does not mask *val* to the right number of bits.

Definition at line 164 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Lsb.

Referenced by cxx::Bitfield< T, LSB, MSB >::val().

Here is the caller graph for this function:

14.8.4.9 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 containing.

Definition at line 177 of file bitfield.

References cxx::Bitfield< T, LSB, MSB >::Mask.

The documentation for this class was generated from the following file:

- l4/cxx/bitfield

14.9 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 >:

14.9.1 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 199 of file bitfield.

The documentation for this class was generated from the following file:

- l4/cxx/bitfield

14.10 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 >:

14.10.1 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 181 of file bitfield.

The documentation for this class was generated from the following file:

- l4/cxx/bitfield

14.11 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 >:

14.11.1 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 209 of file bitfield.

The documentation for this class was generated from the following file:

- l4/cxx/bitfield

14.12 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

14.12.1 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.

14.12.2 Constructor & Destructor Documentation

14.12.2.1 Bitmap()

```
template<int BITS>
cxx::Bitmap< BITS >::Bitmap ( ) throw )   [inline]
```

Create a bitmap with *BITS* bits.

Definition at line 187 of file bitmap.

References cxx::Bitmap_base::scan_zero().

Here is the call graph for this function:

14.12.3 Member Function Documentation

14.12.3.1 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

14.13 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.

14.13.1 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.

14.13.2 Member Enumeration Documentation

14.13.2.1 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.

14.13.3 Member Function Documentation

14.13.3.1 bit() [1/2]

```
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.

References bit_index(), and word_index().

Referenced by operator[ ]().

Here is the call graph for this function:

Here is the caller graph for this function:

14.13.3.2 bit() [2/2]

```
unsigned long 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.

References bit_index(), and word_index().

Here is the call graph for this function:

14.13.3.3 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 W_bits.

Referenced by bit(), clear_bit(), and set_bit().

Here is the caller graph for this function:

14.13.3.4 chars()

```
static long cxx::Bitmap_base::chars (
            long bits ) throw )   [inline], [static]
```

Get the number of chars that are used for the bitmap.

Definition at line 98 of file bitmap.

References C_bits.

14.13.3.5 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.

References bit_index(), and word_index().

Here is the call graph for this function:

14.13.3.6 operator[]() [1/2]

```
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:

14.13.3.7 operator[]() [2/2]

```
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 _bits, and scan_zero().

Here is the call graph for this function:

14.13.3.8 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.

References W_bits, and word_index().

Referenced by cxx::Bitmap< BITS >::Bitmap(), operator[ ]() , and cxx::Bitmap< BITS >::scan_zero().

Here is the call graph for this function:

Here is the caller graph for this function:

14.13.3.9 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.

References bit_index(), and word_index().

Here is the call graph for this function:

14.13.3.10 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 W_bits.

Referenced by bit(), clear_bit(), scan_zero(), and set_bit().

Here is the caller graph for this function:

14.13.3.11 words()

```
static long cxx::Bitmap_base::words (
            long bits ) throw )   [inline], [static]
```

Get the number of *Words* that are used for the bitmap.

Definition at line 81 of file bitmap.

References W_bits.

The documentation for this class was generated from the following file:

- l4/cxx/bitmap

14.14 cxx::Bitmap_base::Bit Class Reference

A writeable bit in a bitmap.

Collaboration diagram for cxx::Bitmap_base::Bit:

14.14.1 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

14.15 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 >:

14.15.1 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

14.16 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 >:

14.16.1 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

14.17 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 >:

14.17.1 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

14.18 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 >:

14.18.1 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 93 of file avl_set.

The documentation for this struct was generated from the following file:

- l4/cxx/avl_set

14.19 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.

14.19.1 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 117 of file avl_set.

14.19.2 Member Enumeration Documentation

14.19.2.1 anonymous enum

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 126 of file avl_set.

14.19.3 Constructor & Destructor Documentation

14.19.3.1 Base_avl_set() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 235 of file avl_set.

14.19.3.2 Base_avl_set() [2/2]

```
template<typename Item , class Compare , template< typename A > class Alloc, typename KEY_T↩
YPE >
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 387 of file avl_set.

14.19.4 Member Function Documentation

14.19.4.1 begin() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 337 of file avl_set.

Referenced by cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::Base_avl_set().

Here is the caller graph for this function:

14.19.4.2 begin() [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 348 of file avl_set.

14.19.4.3 end() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 342 of file avl_set.

Referenced by cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::Base_avl_set().

Here is the caller graph for this function:

14.19.4.4 end() [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 353 of file avl_set.

14.19.4.5 erase()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 305 of file avl_set.

14.19.4.6 find_node()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 316 of file avl_set.

14.19.4.7 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 397 of file avl_set.

14.19.4.8 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 327 of file avl_set.

14.19.4.9 rbegin() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 359 of file avl_set.

14.19.4.10 rbegin() [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 370 of file avl_set.

14.19.4.11 remove()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 285 of file avl_set.

14.19.4.12 rend() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 364 of file avl_set.

14.19.4.13 rend() [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 375 of file avl_set.

The documentation for this class was generated from the following file:

- l4/cxx/avl_set

14.20 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.
- Node & operator= (Node const &o)

  Default assignment.
- 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.

14.20.1 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 >::Node

A smart pointer to a tree item.

Definition at line 161 of file avl_set.

14.20.2 Member Function Documentation

14.20.2.1 operator∗()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 181 of file avl_set.

14.20.2.2 operator->()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 187 of file avl_set.

14.20.2.3 valid()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↩
T_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 193 of file avl_set.

The documentation for this class was generated from the following file:

- l4/cxx/avl_set

14.21 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.

14.21.1 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.

14.21.2 Member Function Documentation

14.21.2.1 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 135 of file list_basics.h.

14.21.2.2 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 148 of file list_basics.h.

The documentation for this class was generated from the following file:

- l4/cxx/bits/list_basics.h

14.22 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

- typedef __Bst_iter< Node, Node, Fwd > Iterator

  Forward 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.

14.22.1 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.

14.22.2 Member Function Documentation

14.22.2.1 begin() [1/2]

```
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.

Referenced by cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::begin().

Here is the caller graph for this function:

14.22.2.2 begin() [2/2]

```
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.

14.22.2.3 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,
            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.

Referenced by cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::dir(), cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::find(), cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::find_node(), and cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::lower_bound_node().

Here is the caller graph for this function:

14.22.2.4 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,
            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.

14.22.2.5 end() [1/2]

```
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< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::end().

Here is the caller graph for this function:

14.22.2.6 end() [2/2]

```
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.

14.22.2.7 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.

Referenced by cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::rend(), and cxx::Bits::Base_avl_set< Pair< KE↩ Y_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::rend().

Here is the caller graph for this function:

14.22.2.8 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.

Referenced by cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, AL↩LOC, Bits::Avl_map_get_key< KEY_TYPE > >::find_node(), and cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::rend().

Here is the caller graph for this function:

14.22.2.9 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 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 292 of file bst.h.

Referenced by cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::lower_bound_node(), and cxx::Bits::Bst< _Node, GET_KEY, COMP↩ ARE >::rend().

Here is the caller graph for this function:

14.22.2.10 rbegin() [1/2]

```
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.

Referenced by cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::rbegin().

Here is the caller graph for this function:

14.22.2.11 rbegin() [2/2]

```
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.

14.22.2.12 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.

Referenced by cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::Base_avl_set().

Here is the caller graph for this function:

14.22.2.13 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.

Referenced by cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::remove_all(), and cxx::Bits::Bst< _Node, GET↩ _KEY, COMPARE >::remove_tree().

Here is the caller graph for this function:

14.22.2.14 rend() [1/2]

```
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< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::rend().

Here is the caller graph for this function:

14.22.2.15 rend() [2/2]

```
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.

The documentation for this class was generated from the following file:

- l4/cxx/bits/bst.h

14.23 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.

14.23.1 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

14.24 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 ()

  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

14.24.1 Detailed Description

The direction to go in a binary search tree.

Definition at line 39 of file bst_base.h.

14.24.2 Member Enumeration Documentation

14.24.2.1 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.

14.24.3 Member Function Documentation

14.24.3.1 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

14.25 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.

14.25.1 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 46 of file smart_ptr_list.h.

14.25.2 Member Function Documentation

14.25.2.1 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 149 of file smart_ptr_list.h.

The documentation for this class was generated from the following file:

- l4/cxx/bits/smart_ptr_list.h

14.26 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 >:

14.26.1 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 27 of file smart_ptr_list.h.

The documentation for this class was generated from the following file:

- l4/cxx/bits/smart_ptr_list.h

14.27 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

14.27.1 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.

14.27.2 Member Function Documentation

14.27.2.1 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.

14.27.2.2 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.

14.27.2.3 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.

14.27.2.4 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.

14.27.2.5 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.

14.27.2.6 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.

14.27.2.7 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.

14.27.2.8 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

14.28 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.

14.28.1 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.

14.28.2 Constructor & Destructor Documentation

14.28.2.1 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.

Referenced by cxx::H_list_item_t< Weak_ref_base >::~H_list_item_t().

Here is the caller graph for this function:

14.28.2.2 ~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

14.29 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

14.29.1 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

14.30 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.

14.30.1 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.

14.30.2 Member Function Documentation

14.30.2.1 items()

```
template<typename D , template< typename A > class Alloc = New_allocator>
Iter cxx::List< D, Alloc >::items ( ) throw )   [inline]
```

Get iterator for the list elements.

Definition at line 412 of file list.

14.30.2.2 operator[]() [1/2]

```
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.

14.30.2.3 operator[]() [2/2]

```
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.

14.30.2.4 push_back()

```
template<typename D , template< typename A > class Alloc = New_allocator>
void cxx::List< D, Alloc >::push_back (
            D const & d ) throw )   [inline]
```

Add element at the end of the list.

Definition at line 379 of file list.

14.30.2.5 push_front()

```
template<typename D , template< typename A > class Alloc = New_allocator>
void cxx::List< D, Alloc >::push_front (
            D const & d ) throw )   [inline]
```

Add element at the beginning of the list.

Definition at line 388 of file list.

14.30.2.6 remove()

```
template<typename D , template< typename A > class Alloc = New_allocator>
void cxx::List< D, Alloc >::remove (
            Iter const & i ) throw )   [inline]
```

Remove element pointed to by the iterator.

Definition at line 397 of file list.

14.30.2.7 size()

```
template<typename D , template< typename A > class Alloc = New_allocator>
unsigned long cxx::List< D, Alloc >::size ( ) const throw )   [inline]
```

Get the length of the list.

Definition at line 401 of file list.

The documentation for this class was generated from the following file:

- l4/cxx/list

14.31 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)

14.31.1 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

14.32 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 align)

  Alloc a memory block.
- void ∗ alloc_max (unsigned long min, unsigned long ∗max, unsigned align, unsigned granularity)

  Allocate a memory block of `min` *<= size <=* `max`*.*
- unsigned long avail ()

  Get the amount of available memory.

14.32.1 Detailed Description

Standard list-based allocator.

Definition at line 31 of file list_alloc.

14.32.2 Constructor & Destructor Documentation

14.32.2.1 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.

References alloc(), alloc_max(), avail(), L4::cerr, free(), cxx::max(), and cxx::min().

Here is the call graph for this function:

14.32.3 Member Function Documentation

14.32.3.1 alloc()

```
void * cxx::List_alloc::alloc (
            unsigned long size,
            unsigned align ) [inline]
```

Alloc a memory block.

Parameters

size	Size of the memory block
align	Alignment constraint

Returns

Pointer to memory block

Definition at line 354 of file list_alloc.

Referenced by List_alloc().

Here is the caller graph for this function:

14.32.3.2 alloc_max()

```
void * cxx::List_alloc::alloc_max (
            unsigned long min,
            unsigned long * max,
            unsigned align,
            unsigned granularity )  [inline]
```

Allocate a memory block of `min` <= size <=`max`.

Parameters

	min	Minimal size to allocate.
in,out	*max*	Maximum size to allocate. The actual allocated size is returned here.
	align	Alignment constraint.
	granularity	Granularity to use for the allocation.

Returns

Pointer to memory block

Definition at line 257 of file list_alloc.

References l4_round_size(), l4_trunc_size(), and cxx::max().

Referenced by List_alloc().

Here is the call graph for this function:

Here is the caller graph for this function:

14.32.3.3 avail()

```
unsigned long cxx::List_alloc::avail ( )  [inline]
```

Get the amount of available memory.

Returns

Available memory in bytes

Definition at line 425 of file list_alloc.

Referenced by List_alloc().

Here is the caller graph for this function:

14.32.3.4 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.

Definition at line 218 of file list_alloc.

Referenced by List_alloc().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/list_alloc

14.33 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.

14.33.1 Detailed Description

Basic list item.

Basic item that can be member of a doubly linked, cyclic list.

Definition at line 37 of file list.

14.33.2 Member Function Documentation

14.33.2.1 get_next_item()

```
List_item* cxx::List_item::get_next_item ( ) const throw )   [inline]
```

Get next item.

Definition at line 176 of file list.

Referenced by remove().

Here is the caller graph for this function:

14.33.2.2 get_prev_item()

```
List_item* cxx::List_item::get_prev_item ( ) const throw )   [inline]
```

Get previous item.

Definition at line 173 of file list.

Referenced by remove().

Here is the caller graph for this function:

14.33.2.3 insert_next_item()

```
void cxx::List_item::insert_next_item (
            List_item * p ) throw )   [inline]
```

Insert item p after this item.

Definition at line 189 of file list.

14.33.2.4 insert_prev_item()

```
void cxx::List_item::insert_prev_item (
            List_item * p ) throw )   [inline]
```

Insert item p before this item.

Definition at line 179 of file list.

14.33.2.5 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.

Referenced by remove(), and remove_me().

Here is the caller graph for this function:

14.33.2.6 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.

Referenced by remove(), and remove_me().

Here is the caller graph for this function:

14.33.2.7 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.

References get_next_item(), get_prev_item(), push_back(), push_front(), and cxx::List_item::Iter::remove_me().

Here is the call graph for this function:

14.33.2.8 remove_me()

```
void cxx::List_item::remove_me ( ) throw )   [inline]
```

Remove this item from the list.

Definition at line 198 of file list.

References push_back(), and push_front().

Referenced by cxx::List_item::Iter::remove_me().

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/list

14.34 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.

14.34.1 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.

14.34.2 Member Function Documentation

14.34.2.1 remove_me()

```
List_item* cxx::List_item::Iter::remove_me ( ) throw )   [inline]
```

Remove item pointed to by iterator, and return pointer to element.

Definition at line 86 of file list.

References cxx::List_item::remove_me().

Referenced by cxx::List_item::remove().

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/list

14.35 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

14.35.1 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

14.36 cxx::Lt_functor< Obj > Struct Template Reference

Generic comparator class that defaults to the less-than operator.

Collaboration diagram for cxx::Lt_functor< Obj >:

14.36.1 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

14.37 cxx::New_allocator< _Type > Class Template Reference

Standard allocator based on `operator new ()` .

Collaboration diagram for cxx::New_allocator< _Type >:

14.37.1 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 60 of file std_alloc.

The documentation for this class was generated from the following file:

- l4/cxx/std_alloc

14.38 cxx::Nothrow Class Reference

Helper type to distinguish the `oeprator new` version that does not throw exceptions.

Collaboration diagram for cxx::Nothrow:

14.38.1 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

14.39 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 construction.

Data Fields

- First first

  First value.
- Second second

  Second value.

14.39.1 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.

14.39.2 Constructor & Destructor Documentation

14.39.2.1 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

14.40 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.

14.40.1 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.

14.40.2 Constructor & Destructor Documentation

14.40.2.1 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.

14.40.3 Member Function Documentation

14.40.3.1 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

14.41 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 >:

14.41.1 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 26 of file ref_ptr_list.

The documentation for this struct was generated from the following file:

- l4/cxx/ref_ptr_list

14.42 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 () throw ()

  Default constructor creates a pointer with no managed object.
- Ref_ptr (Wp const &o) throw ()

  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) throw ()

  Create a shared pointer from a raw pointer.
- Ref_ptr (T *o, bool d) throw ()

  Create a shared pointer from a raw pointer without creating a new reference.
- T * get () const throw ()

  Return a raw pointer to the object this shared pointer points to.
- T * ptr () const throw ()

  Return a raw pointer to the object this shared pointer points to.
- T * release () throw ()

  Release the shared pointer without removing the reference.

14.42.1 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 80 of file ref_ptr.

14.42.2 Constructor & Destructor Documentation

14.42.2.1 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 ) throw )   [inline]
```

Create a shared pointer from a weak pointer.

Increases references.

Definition at line 101 of file ref_ptr.

14.42.2.2 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 ) throw )   [inline], [explicit]
```

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 115 of file ref_ptr.

14.42.2.3 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 ) throw )   [inline]
```

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 138 of file ref_ptr.

14.42.3 Member Function Documentation

14.42.3.1 get()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T* cxx::Ref_ptr< T, CNT >::get ( ) const throw )   [inline]
```

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 145 of file ref_ptr.

Referenced by cxx::Ref_ptr< Mount_tree >::release().

Here is the caller graph for this function:

14.42.3.2 ptr()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T* cxx::Ref_ptr< T, CNT >::ptr ( ) const throw )   [inline]
```

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 151 of file ref_ptr.

Referenced by cxx::Ref_ptr< Mount_tree >::release().

Here is the caller graph for this function:

14.42.3.3 release()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T* cxx::Ref_ptr< T, CNT >::release ( ) throw )   [inline]
```

Release the shared pointer without removing the reference.

Returns

A raw pointer to the managed object.

Definition at line 162 of file ref_ptr.

Referenced by cxx::Ref_ptr< Mount_tree >::release().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/ref_ptr

14.43 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

14.43.1 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.

14.43.2 Member Function Documentation

14.43.2.1 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 91 of file slist.

The documentation for this class was generated from the following file:

- l4/cxx/slist

14.44 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

14.44.1 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*.

Parameters

Type	the type of the objects to manage.
Slab_size	size of a slab cache.
Max_free	the maximum number of free slab caches.
Alloc	the allocator for the slab caches.

Definition at line 297 of file slab_alloc.

14.44.2 Member Function Documentation

14.44.2.1 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.

Definition at line 314 of file slab_alloc.

14.44.2.2 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 325 of file slab_alloc.

References L4_PAGESIZE.

The documentation for this class was generated from the following file:

- l4/cxx/slab_alloc

14.45 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.

14.45.1 Detailed Description

template< typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator> class cxx::Slab_static< Type, Slab_size, Max_free, Alloc >

Merged slab allocator (allocators for objects of the same size are merged together).

Parameters

Type	The type of the objects to manage.
Slab_size	The size of a slab cache.
Max_free	The maximum number of free slab caches.
Alloc	The allocator for the slab caches.

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 415 of file slab_alloc.

14.45.2 Member Function Documentation

14.45.2.1 alloc()

```
template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A
> class Alloc = New_allocator>
Type* cxx::Slab_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 of failure.

Definition at line 425 of file slab_alloc.

The documentation for this class was generated from the following file:

- l4/cxx/slab_alloc

14.46 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.

14.46.1 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

14.47 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.

14.47.1 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 42 of file string.

14.47.2 Constructor & Destructor Documentation

14.47.2.1 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 60 of file string.

14.47.3 Member Function Documentation

14.47.3.1 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 203 of file string.

References end(), and eof().

Here is the call graph for this function:

14.47.3.2 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 240 of file string.

References eof(), and start().

Here is the call graph for this function:

14.47.3.3 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 269 of file string.

References eof(), and start().

Here is the call graph for this function:

14.47.3.4 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 167 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

14.48 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

14.48.1 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

14.49 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

14.49.1 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

14.50 Elf32_Dyn Struct Reference

ELF32 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Dyn:

Data Fields

- Elf32_Sword d_tag

  see DT_ values
- Elf32_Word d_val

  integer values with various interpret.
- Elf32_Addr d_ptr

  program virtual addresses

14.50.1 Detailed Description

ELF32 dynamic entry.

Definition at line 500 of file elf.h.

14.50.2 Field Documentation

14.50.2.1 d_val

```
Elf32_Word Elf32_Dyn::d_val
```

integer values with various interpret.

Definition at line 503 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

14.51 Elf32_Ehdr Struct Reference

ELF32 header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Ehdr:

Data Fields

- 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 eip
- 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

of entries in prog.

- Elf32_Half e_shentsize

  size of section header entry
- Elf32_Half e_shnum

  of entries in sect.
- Elf32_Half e_shstrndx

  sect.head.tab.idx of strtab

14.51.1 Detailed Description

ELF32 header.

Definition at line 122 of file elf.h.

14.51.2 Field Documentation

14.51.2.1 e_phnum

```
Elf32_Half Elf32_Ehdr::e_phnum
```

of entries in prog.

head. tab.

Definition at line 133 of file elf.h.

14.51.2.2 e_shnum

```
Elf32_Half Elf32_Ehdr::e_shnum
```

of entries in sect.

head. tab.

Definition at line 135 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

14.52 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

14.52.1 Detailed Description

ELF32 program header.

Definition at line 419 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

14.53 Elf32_Shdr Struct Reference

ELF32 section header - figure 1-9, page 1-9.

```
#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

14.53.1 Detailed Description

ELF32 section header - figure 1-9, page 1-9.

Definition at line 342 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

14.54 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

14.54.1 Detailed Description

ELF32 symbol table entry.

Definition at line 801 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

14.55 Elf64_Dyn Struct Reference

ELF64 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Dyn:

Data Fields

- Elf64_Sxword d_tag

  see DT_ values
- Elf64_Xword d_val

  integer values with various interpret.
- Elf64_Addr d_ptr

  program virtual addresses

14.55.1 Detailed Description

ELF64 dynamic entry.

Definition at line 509 of file elf.h.

14.55.2 Field Documentation

14.55.2.1 d_val

```
Elf64_Xword Elf64_Dyn::d_val
```

integer values with various interpret.

Definition at line 512 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

14.56 Elf64_Ehdr Struct Reference

ELF64 header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Ehdr:

Data Fields

- 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 eip
- 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

  of entries in prog.
- Elf64_Half e_shentsize

  size of section header entry
- Elf64_Half e_shnum

  of entries in sect.
- Elf64_Half e_shstrndx

  sect.head.tab.idx of strtab

14.56.1 Detailed Description

ELF64 header.

Definition at line 142 of file elf.h.

14.56.2 Field Documentation

14.56.2.1 e_phnum

```
Elf64_Half Elf64_Ehdr::e_phnum
```

of entries in prog.

head. tab.

Definition at line 153 of file elf.h.

14.56.2.2 e_shnum

```
Elf64_Half Elf64_Ehdr::e_shnum
```

of entries in sect.

head. tab.

Definition at line 155 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

14.57 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

14.57.1 Detailed Description

ELF64 program header.

Definition at line 431 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

14.58 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

14.58.1 Detailed Description

ELF64 section header.

Definition at line 356 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

14.59 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

14.59.1 Detailed Description

ELF64 symbol table entry.

Definition at line 811 of file elf.h.

The documentation for this struct was generated from the following file:

- l4/util/elf.h

14.60 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

14.60.1 Detailed Description

offsets in pmap[] and bmap[]

Definition at line 69 of file bitmap.h.

The documentation for this struct was generated from the following file:

- l4/libgfxbitmap/bitmap.h

14.61 L4::Alloc_list Class Reference

A simple list-based allocator.

```
#include <alloc.h>
```

Collaboration diagram for L4::Alloc_list:

14.61.1 Detailed Description

A simple list-based allocator.

Definition at line 33 of file alloc.h.

The documentation for this class was generated from the following file:

- l4/cxx/alloc.h

14.62 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.

14.62.1 Detailed Description

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

Definition at line 20 of file arm_smccc.

14.62.2 Member Function Documentation

14.62.2.1 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.

The documentation for this class was generated from the following file:

- l4/sys/arm_smccc

14.63 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.

14.63.1 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

14.64 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.

14.64.1 Detailed Description

This registry returns the corresponding server object based on the label of an lpc_gate.

Definition at line 509 of file ipc_epiface.

14.64.2 Member Function Documentation

14.64.2.1 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 534 of file ipc_epiface.

14.64.2.2 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 518 of file ipc_epiface.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_epiface

14.65 L4::Bounds_error Class Reference

Access out of bounds.

Inheritance diagram for L4::Bounds_error:

Collaboration diagram for L4::Bounds_error:

Additional Inherited Members

14.65.1 Detailed Description

Access out of bounds.

Definition at line 290 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

14.66 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) throw ()

  Create a copy from o, supporting implicit type casting.
- Cap (Cap_type cap) throw ()

  Constructor to create an invalid capability selector.
- Cap (l4_default_caps_t cap) throw ()

  Initialize capability with one of the default capability selectors.
- Cap (l4_cap_idx_t idx=L4_INVALID_CAP) throw ()

  Initialize capability, defaults to the invalid capability selector.
- Cap (No_init_type) throw ()

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 throw ()

  Member access of a T.

Friends

- class **L4::Kobject**

Additional Inherited Members

14.66.1 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 13 of file capability.h.

14.66.2 Constructor & Destructor Documentation

14.66.2.1 Cap() [1/4]

```
template<typename T>
template<typename O >
L4::Cap< T >::Cap (
            Cap< O > const & o ) throw )   [inline]
```

Create a copy from o, supporting implicit type casting.

Parameters

o	The source selector that shall be copied (and casted).

Definition at line 242 of file capability.h.

14.66.2.2 Cap() [2/4]

```
template<typename T>
L4::Cap< T >::Cap (
            Cap_type cap ) throw )   [inline]
```

Constructor to create an invalid capability selector.

Parameters

cap	Capability selector.

Definition at line 249 of file capability.h.

14.66.2.3 Cap() [3/4]

```
template<typename T>
L4::Cap< T >::Cap (
            l4_default_caps_t cap ) throw )   [inline]
```

Initialize capability with one of the default capability selectors.

Parameters

cap	Capability selector.

Definition at line 255 of file capability.h.

14.66.2.4 Cap() [4/4]

```
template<typename T>
L4::Cap< T >::Cap (
            l4_cap_idx_t idx = L4_INVALID_CAP ) throw )   [inline], [explicit]
```

Initialize capability, defaults to the invalid capability selector.

Parameters

idx	Capability selector.

Definition at line 261 of file capability.h.

14.66.3 Member Function Documentation

14.66.3.1 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 284 of file capability.h.

14.66.3.2 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 274 of file capability.h.

The documentation for this class was generated from the following file:

- l4/sys/cxx/capability.h

14.67 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 throw ()

  Return capability selector.
- bool is_valid () const throw ()

  Test whether the capability is a valid capability index (i.e., not L4_INVALID_CAP).
- l4_fpage_t fpage (unsigned rights=L4_FPAGE_RWX) const throw ()

  Return flex-page for the capability.
- l4_umword_t snd_base (unsigned grant=0, l4_cap_idx_t base=L4_INVALID_CAP) const throw ()

  Return send base.
- bool operator== (Cap_base const &o) const throw ()

  Test if two capabilities are equal.
- bool operator!= (Cap_base const &o) const throw ()

  Test if two capabilities are not equal.
- l4_msgtag_t validate (l4_utcb_t *u=l4_utcb()) const throw ()

  Check whether a capability is present (refers to an object).
- l4_msgtag_t validate (Cap< Task > task, l4_utcb_t *u=l4_utcb()) const throw ()

  Check whether a capability is present (refers to an object).
- void invalidate () throw ()

  Set this capability to invalid (L4_INVALID_CAP).

Protected Member Functions

- Cap_base (l4_cap_idx_t c) throw ()

  Generate a capability from its C representation.
- Cap_base (Cap_type cap) throw ()

  Constructor to create an invalid capability.
- Cap_base (l4_default_caps_t cap) throw ()

  Initialize capability with one of the default capabilities.
- Cap_base () throw ()

  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.

14.67.1 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.

14.67.2 Member Enumeration Documentation

14.67.2.1 Cap_type

```
enum L4::Cap_base::Cap_type
```

Invalid capability type.

Enumerator

Invalid	Invalid capability selector.

Definition at line 43 of file capability.h.

14.67.2.2 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.

14.67.3 Constructor & Destructor Documentation

14.67.3.1 Cap_base() [1/2]

```
L4::Cap_base::Cap_base (
            l4_cap_idx_t c ) throw )   [inline], [explicit], [protected]
```

Generate a capability from its C representation.

Parameters

c	The C capability

Definition at line 142 of file capability.h.

14.67.3.2 Cap_base() [2/2]

```
L4::Cap_base::Cap_base (
            l4_default_caps_t cap ) throw )   [inline], [explicit], [protected]
```

Initialize capability with one of the default capabilities.

Parameters

cap	Capability.

Definition at line 153 of file capability.h.

14.67.4 Member Function Documentation

14.67.4.1 cap()

```
l4_cap_idx_t L4::Cap_base::cap ( ) const throw )   [inline]
```

Return capability selector.

Returns

Capability selector.

Definition at line 52 of file capability.h.

Referenced by L4Re::Util::Cap_alloc_base::alloc(), L4vbus::Vbus::assign_dma_domain(), L4::Icu::bind(), L4::cap↩_cast(), L4::cap_reinterpret_cast(), L4::Irq_mux::chain(), L4Re::Util::Smart_count_cap< Unmap_flags >::copy(), L4Re::Smart_count_cap< Unmap_flags >::copy(), L4::Factory::create_factory(), L4::Factory::create_gate(), L4↩::Factory::create_irq(), L4::Factory::create_task(), L4::Factory::create_thread(), L4::Factory::create_vm(), L4Re::↩Util::Smart_cap_auto< Unmap_flags >::free(), L4::Invalid_capability::Invalid_capability(), L4virtio::Svr::Device_t< Ds_data >::mem_info(), L4::Smart_cap< T, SMART >::operator->(), L4::Cap< L4Re::Video::Goos >::operator->(), L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process(), L4::Ipc_svr::Server_iface::rcv_cap(), L4::↩Smart_cap< T, SMART >::Smart_cap(), L4::Icu::unbind(), and L4vbus::Icu::vicu().

Here is the caller graph for this function:

14.67.4.2 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 185 of file capability.h.

References fpage(), is_valid(), L4_BASE_TASK_CAP, L4_CAP_FPAGE_RWSD, and l4_task_map().

Referenced by L4::Cap< L4Re::Video::Goos >::copy(), and L4::Cap< L4Re::Video::Goos >::operator->().

Here is the call graph for this function:

Here is the caller graph for this function:

14.67.4.3 fpage()

```
l4_fpage_t L4::Cap_base::fpage (
            unsigned rights = L4_FPAGE_RWX ) const throw )   [inline]
```

Return flex-page for the capability.

Parameters

rights	Rights, defaults to 'rwx'

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:

14.67.4.4 is_valid()

```
bool L4::Cap_base::is_valid ( ) const throw )   [inline]
```

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/libs/l4re/c++/mem_alloc/ma+rm.cc, and examples/libs/l4re/c++/shared_ds/ds_clnt.cc.

Definition at line 60 of file capability.h.

Referenced by L4Re::Rm::attach(), copy(), L4Re::Smart_count_cap< Unmap_flags >::copy(), L4Re::Util::Smart↩_cap_auto< Unmap_flags >::free(), L4Re::Util::Smart_cap_auto< Unmap_flags >::invalidate(), L4::Ipc::make_↩cap_full(), move(), and L4Re::Video::View::valid().

Here is the caller graph for this function:

14.67.4.5 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 referd to by the source capability `src`, and the source capability no longer refers to an object.

Definition at line 169 of file capability.h.

References fpage(), is_valid(), L4_BASE_TASK_CAP, L4_CAP_FPAGE_RWSD, L4_MAP_ITEM_GRANT, and l4_task_map().

Referenced by L4::Cap< L4Re::Video::Goos >::move(), and L4::Cap< L4Re::Video::Goos >::operator->().

Here is the call graph for this function:

Here is the caller graph for this function:

14.67.4.6 snd_base()

```
l4_umword_t L4::Cap_base::snd_base (
            unsigned grant = 0,
            l4_cap_idx_t base = L4_INVALID_CAP ) const throw )   [inline]
```

Return send base.

Parameters

grant	True object should be granted.
base	Base capability (first in a bundle of aligned capabilities)

Returns

Map object.

Definition at line 83 of file capability.h.

References L4_INVALID_CAP, and l4_map_obj_control().

Here is the call graph for this function:

14.67.4.7 validate() [1/2]

```
l4_msgtag_t L4::Cap_base::validate (
            l4_utcb_t * u = l4_utcb() ) const throw )   [inline]
```

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).

A capability is considered present when it refers to an existing kernel object.

Definition at line 87 of file capability.

14.67.4.8 validate() [2/2]

```
l4_msgtag_t L4::Cap_base::validate (
            Cap< Task > task,
            l4_utcb_t * u = l4_utcb() ) const throw )   [inline]
```

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).

A capability is considered present when it refers to an existing kernel object.

Definition at line 83 of file capability.

14.67.5 Field Documentation

14.67.5.1 _c

```
l4_cap_idx_t L4::Cap_base::_c  [protected]
```

The C representation of a capability selector.

Definition at line 196 of file capability.h.

Referenced by L4::Smart_cap< T, SMART >::operator->(), and L4::Smart_cap< T, SMART >::Smart_cap().

The documentation for this class was generated from the following files:

- l4/sys/cxx/capability.h
- l4/sys/capability

14.68 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 givel L4 IPC error code.

Additional Inherited Members

14.68.1 Detailed Description

Error conditions during IPC.

This exception encapsulates all IPC error conditions of L4 IPC.

Definition at line 275 of file exceptions.

14.68.2 Constructor & Destructor Documentation

14.68.2.1 Com_error()

```
L4::Com_error::Com_error (
            long err ) throw )   [inline], [explicit]
```

Create a Com_error for the givel L4 IPC error code.

Parameters

err	The L4 IPC error code (l4_ipc... return value).

Definition at line 282 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

14.69 L4::Debugger Class Reference

C++ debugger interface.

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()) throw ()

  Set the name of a kernel object.
- unsigned long global_id (l4_utcb_t ∗utcb=l4_utcb()) throw ()

  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()) throw ()

  Get the globally unique ID of the object behind the kobject pointer.
- int query_log_typeid (const char ∗name, unsigned idx, l4_utcb_t ∗utcb=l4_utcb()) throw ()

  Query the log-id for a log type.
- int query_log_name (unsigned idx, char ∗name, unsigned namelen, char ∗shortname, unsigned shortnamelen, l4_utcb_t ∗utcb=l4_utcb()) throw ()

  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()) throw ()

Set or unset log.

- l4_msgtag_t get_object_name (unsigned id, char *name, unsigned size, l4_utcb_t *utcb=l4_utcb()) throw ()

  Get name of object with Id `id`.

Additional Inherited Members

14.69.1 Detailed Description

C++ debugger interface.

Attention

This API is subject to change! Do not rely on it in production code.

Include File

```
#include <l4/sys/debugger>
```

Definition at line 39 of file debugger.

14.69.2 Member Function Documentation

14.69.2.1 get_object_name()

```
l4_msgtag_t L4::Debugger::get_object_name (
            unsigned id,
            char * name,
            unsigned size,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 145 of file debugger.

14.69.2.2 global_id()

```
unsigned long L4::Debugger::global_id (
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 68 of file debugger.

14.69.2.3 kobj_to_id()

```
unsigned long L4::Debugger::kobj_to_id (
            l4_addr_t kobjp,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 80 of file debugger.

14.69.2.4 query_log_name()

```
int L4::Debugger::query_log_name (
            unsigned idx,
            char * name,
            unsigned namelen,
```

```
                char * shortname,
                unsigned shortnamelen,
                l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 113 of file debugger.

14.69.2.5 query_log_typeid()

```
int L4::Debugger::query_log_typeid (
                const char * name,
                unsigned idx,
                l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 94 of file debugger.

14.69.2.6 set_object_name()

```
l4_msgtag_t L4::Debugger::set_object_name (
            const char * name,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 56 of file debugger.

14.69.2.7 switch_log()

```
l4_msgtag_t L4::Debugger::switch_log (
            const char * name,
            unsigned on_off,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 130 of file debugger.

The documentation for this class was generated from the following file:

- l4/sys/debugger

14.70 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

14.70.1 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

14.71 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

14.71.1 Detailed Description

Exception for a failed lookup (element not found).

Definition at line 232 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

14.72 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.

14.72.1 Detailed Description

Exception interface.

Definition at line 35 of file exception.

14.72.2 Member Typedef Documentation

14.72.2.1 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 53 of file exception.

The documentation for this class was generated from the following file:

- l4/sys/exception

14.73 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.

14.73.1 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 63 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

14.74 L4::Factory Class Reference

C++ Factory interface to create kernel objects.

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()) throw ()

Generic create call to the factory.

- template<typename OBJ >
  S create (Cap< OBJ > target, l4_utcb_t *utcb=l4_utcb()) throw ()

  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()) throw ()

  Create a new task.
- l4_msgtag_t create_thread (Cap< Thread > const &target_cap, l4_utcb_t *utcb=l4_utcb()) throw ()

  Create a new thread.
- l4_msgtag_t create_factory (Cap< Factory > const &target_cap, unsigned long limit, l4_utcb_t *utcb=l4_utcb()) throw ()

  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()) throw ()

  Create a new IPC gate.
- l4_msgtag_t create_irq (Cap< Irq >const &target_cap, l4_utcb_t *utcb=l4_utcb()) throw ()

  Create a new IRQ.
- l4_msgtag_t create_vm (Cap< Vm >const &target_cap, l4_utcb_t *utcb=l4_utcb()) throw ()

  Create a new virtual machine.

Additional Inherited Members

14.74.1 Detailed Description

C++ Factory interface to create kernel objects.

A factory is used to create all kinds of kernel objects:

- L4::Task
- L4::Thread
- L4::Factory
- L4::Ipc_gate
- L4::Irq
- L4::Vm

The factory is equipped with a limit that limits the amount of kernel memory available to that factory.

Note

The limit does not give any guarantee for the amount of available kernel memory.

Include File

```
#include <l4/sys/factory>
```

For the C interface refer to Factory.

Definition at line 55 of file factory.

14.74.2 Member Function Documentation

14.74.2.1 create() [1/2]

```
S L4::Factory::create (
            Cap< void > target,
            long obj,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 adding additional arguments to the create() call.

This method does currently not directly invoke the factory. It returns a stream that shall invoke the factory after adding all additional arguments.

Usage:

```
L4::Cap<L4Re::Namespace> ns = L4Re::Util::cap_alloc.
      alloc<L4Re::Namespace>();
factory->create(ns, L4Re::Namespace::Protocol) << "Argument text";
```

Definition at line 259 of file factory.

References L4::Kobject::cap().

Referenced by L4Re::Mem_alloc::alloc(), and create().

Here is the call graph for this function:

Here is the caller graph for this function:

14.74.2.2 create() [2/2]

```
template<typename OBJ >
S L4::Factory::create (
            Cap< OBJ > target,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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

Argument stream to call the factory.

See also

create

Definition at line 276 of file factory.

References L4::Kobject::cap(), create(), and L4_INLINE_RPC_NF.

Here is the call graph for this function:

14.74.2.3 create_factory()

```
l4_msgtag_t L4::Factory::create_factory (
            Cap< Factory > const & target_cap,
            unsigned long limit,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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

The limit of the new factory is subtracted from the available amount of the factory used for creation.

Definition at line 342 of file factory.

References L4::Cap_base::cap(), L4::Kobject::cap(), and l4_factory_create_factory_u().

Here is the call graph for this function:

14.74.2.4 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() ) throw )   [inline]
```

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 (is used if `thread_cap` is valid).
	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.
-L4_ENOENT	`thread_cap` is void or points to something that is not a thread.
-L4_EPERM	No write rights on `thread_cap`.

An unbound IPC gate can be bound to a thread using L4::lpc_gate::bind_thread().

See also

L4::Ipc_gate

Definition at line 372 of file factory.

References L4::Cap_base::cap(), L4::Kobject::cap(), and l4_factory_create_gate_u().

Here is the call graph for this function:

14.74.2.5 create_irq()

```
l4_msgtag_t L4::Factory::create_irq (
            Cap< Irq >const & target_cap,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

Create a new IRQ.

Parameters

out	*target_cap*	The kernel stores the new IRQ's capability into this slot.
	utcb	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Irq>` as argument instead.

See also

L4::Irq

Definition at line 390 of file factory.

References L4::Cap_base::cap(), L4::Kobject::cap(), and l4_factory_create_irq_u().

Here is the call graph for this function:

14.74.2.6 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() ) throw )   [inline]
```

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.

See also

L4::Task

Definition at line 306 of file factory.

References L4::Cap_base::cap(), L4::Kobject::cap(), and l4_factory_create_task_u().

Here is the call graph for this function:

14.74.2.7 create_thread()

```
l4_msgtag_t L4::Factory::create_thread (
            Cap< Thread > const & target_cap,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

Create a new thread.

Parameters

out	*target_cap*	The kernel stores the new thread's capability into this slot.
	utcb	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Thread>` as argument instead.

See also

L4::Thread

Definition at line 324 of file factory.

References L4::Cap_base::cap(), L4::Kobject::cap(), and l4_factory_create_thread_u().

Here is the call graph for this function:

14.74.2.8 create_vm()

```
l4_msgtag_t L4::Factory::create_vm (
            Cap< Vm >const & target_cap,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

Create a new virtual machine.

Parameters

out	*target_cap*	The kernel stores the new VM's capability into this slot.
	utcb	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Vm>` as argument instead.

See also

L4::Vm

Definition at line 408 of file factory.

References L4::Cap_base::cap(), L4::Kobject::cap(), and l4_factory_create_vm_u().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/factory

14.75 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, int len)

Data Fields

- char const ∗ s

  The character buffer.
- int len

  The number of characters in the buffer.

14.75.1 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 71 of file factory.

14.75.2 Constructor & Destructor Documentation

14.75.2.1 Lstr()

```
L4::Factory::Lstr::Lstr (
            char const * s,
            int len )  [inline]
```

Parameters

s	Pointer to the c-style string.
len	Length in number of characters of the string s.

Definition at line 87 of file factory.

The documentation for this struct was generated from the following file:

- l4/sys/factory

14.76 L4::Factory::Nil Struct Reference

Special type to add a void argument into the factory create stream.

Collaboration diagram for L4::Factory::Nil:

14.76.1 Detailed Description

Special type to add a void argument into the factory create stream.

Definition at line 64 of file factory.

The documentation for this struct was generated from the following file:

- l4/sys/factory

14.77 L4::Factory::S Class Reference

Stream class for the create() argument stream.

Collaboration diagram for L4::Factory::S:

Public Member Functions

- S (S const &o)

  Create a copy.
- S (l4_cap_idx_t f, long obj, L4::Cap< void > target, l4_utcb_t *utcb) throw ()

  Create a stream for a specific create() call.
- ~S ()

  Commit the operation in the destructor to have a cool syntax for create().
- operator l4_msgtag_t ()

  Explicitly commits the operation and returns the result.
- S & operator<< (l4_mword_t i)

  Put a single l4_mword_t as next argument.
- S & operator<< (l4_umword_t i)

  Put a single l4_umword_t as next argument.

- S & operator<< (char const *s)

  Add a zero-terminated string as next argument.
- S & operator<< (Lstr const &s)

  Add a pascal string as next argument.
- S & operator<< (Nil)

  Add an empty argument.
- S & operator<< (l4_fpage_t d)

  Add a flex page as next argument.

14.77.1 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 96 of file factory.

14.77.2 Constructor & Destructor Documentation

14.77.2.1 S() [1/2]

```
L4::Factory::S::S (
            S const & o )  [inline]
```

Create a copy.

Parameters

o	Instance of S to copy.

Definition at line 109 of file factory.

References l4_msgtag_t::raw.

14.77.2.2 S() [2/2]

```
L4::Factory::S::S (
            l4_cap_idx_t f,
            long obj,
            L4::Cap< void > target,
            l4_utcb_t * utcb ) throw )   [inline]
```

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 124 of file factory.

14.77.3 Member Function Documentation

14.77.3.1 operator l4_msgtag_t()

```
L4::Factory::S::operator l4_msgtag_t ( ) [inline]
```

Explicitly commits the operation and returns the result.

Returns

The result of the create() operation.

Definition at line 144 of file factory.

14.77.3.2 operator<<() [1/6]

```
S& L4::Factory::S::operator<< (
            l4_mword_t i ) [inline]
```

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 158 of file factory.

14.77.3.3 operator<<() [2/6]

```
S& L4::Factory::S::operator<< (
            l4_umword_t i )  [inline]
```

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 171 of file factory.

14.77.3.4 operator<<() [3/6]

```
S& L4::Factory::S::operator<< (
            char const * s )  [inline]
```

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 186 of file factory.

14.77.3.5 operator<<() [4/6]

```
S& L4::Factory::S::operator<< (
            Lstr const & s )  [inline]
```

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 203 of file factory.

14.77.3.6 operator<<() [5/6]

```
S& L4::Factory::S::operator<< (
            Nil  )  [inline]
```

Add an empty argument.

Returns

Reference to this stream.

Definition at line 214 of file factory.

14.77.3.7 operator<<() [6/6]

```
S& L4::Factory::S::operator<< (
            l4_fpage_t d )  [inline]
```

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 227 of file factory.

The documentation for this class was generated from the following file:

- l4/sys/factory

14.78 L4::Icu Class Reference

C++ Icu 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()) throw ()

  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()) throw ()

  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()) throw ()

  Get information about the capabilities of the ICU.
- 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()) throw ()

  Mask an IRQ line.
- l4_msgtag_t set_mode (unsigned irqnum, l4_umword_t mode, l4_utcb_t *utcb=l4_utcb()) throw ()

  Set interrupt mode.

Additional Inherited Members

14.78.1 Detailed Description

C++ Icu interface.

To setup an IRQ line the following steps are required:

1. set_mode() (optional if IRQ has a default mode)
2. Irq::attach() to attach the IRQ capability to a thread
3. bind()
4. unmask() to receive the first IRQ

Include File

```
#include <l4/sys/icu>
```

Definition at line 251 of file irq.

14.78.2 Member Function Documentation

14.78.2.1 bind()

```
l4_msgtag_t L4::Icu::bind (
            unsigned irqnum,
            L4::Cap< Triggerable > irq,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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.

Definition at line 299 of file irq.

References L4::Cap_base::cap(), L4::Kobject::cap(), l4_icu_bind_u(), L4_ICU_OP_BIND, and L4_RPC_NF_OP.

Referenced by L4Re::Util::Event_t< PAYLOAD >::init().

Here is the call graph for this function:

Here is the caller graph for this function:

14.78.2.2 info()

```
l4_msgtag_t L4::Icu::info (
            l4_icu_info_t * info,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

Get information about the capabilities of the ICU.

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 330 of file irq.

References L4::Kobject::cap(), l4_icu_info_u(), L4_ICU_OP_INFO, L4_ICU_OP_MSI_INFO, L4_INLINE_RPC_OP, L4_RPC_NF_OP, and l4_utcb().

Here is the call graph for this function:

14.78.2.3 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() ) throw )   [inline]
```

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 373 of file irq.

References L4::Kobject::cap(), l4_icu_mask_u(), L4_ICU_OP_MASK, L4_ICU_OP_UNMASK, L4_RPC_NF_OP, and L4::Irq_eoi::unmask().

Here is the call graph for this function:

14.78.2.4 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

14.78.2.5 set_mode()

```
l4_msgtag_t L4::Icu::set_mode (
            unsigned irqnum,
            l4_umword_t mode,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 395 of file irq.

References L4::Kobject::cap(), L4_ICU_OP_SET_MODE, l4_icu_set_mode_u(), and L4_RPC_NF_OP.

Here is the call graph for this function:

14.78.2.6 unbind()

```
l4_msgtag_t L4::Icu::unbind (
            unsigned irqnum,
            L4::Cap< Triggerable > irq,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 315 of file irq.

References L4::Cap_base::cap(), L4::Kobject::cap(), L4_ICU_OP_UNBIND, l4_icu_unbind_u(), and L4_RPC_N↩F_OP.

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/irq

14.79 L4::Icu::Info Class Reference

This class encapsulates information about an ICU.

Inheritance diagram for L4::Icu::Info:

Collaboration diagram for L4::Icu::Info:

Additional Inherited Members

14.79.1 Detailed Description

This class encapsulates information about an ICU.

Definition at line 278 of file irq.

The documentation for this class was generated from the following file:

- l4/sys/irq

14.80 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_obejct 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

14.80.1 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 246 of file exceptions.

14.80.2 Constructor & Destructor Documentation

14.80.2.1 Invalid_capability()

```
L4::Invalid_capability::Invalid_capability (
            Cap< void > const & o ) throw )   [inline], [explicit]
```

Create an Invalid_obejct exception for the Object o.

Parameters

o	The object that caused the server side error.

Definition at line 256 of file exceptions.

References L4::Cap_base::cap().

Here is the call graph for this function:

14.80.3 Member Function Documentation

14.80.3.1 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 265 of file exceptions.

The documentation for this class was generated from the following file:

- l4/cxx/exceptions

14.81 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::Opt< l4_mword_t &> result, L4::Ipc::Rcv_fpage rwin, L4::Ipc::Opt< L4::Ipc::Snd_fpage &> fp)

  IO page fault protocol message.

14.81.1 Detailed Description

Io_pager interface.

Definition at line 35 of file pager.

14.81.2 Member Function Documentation

14.81.2.1 io_page_fault()

```
l4_msgtag_t L4::Io_pager::io_page_fault (
            l4_fpage_t io_pfa,
            l4_umword_t pc,
            L4::Ipc::Opt< l4_mword_t &> result,
            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.
out	*result*	Optional: handling result value.
	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 by filling in `result` and / or `fp`.

The documentation for this class was generated from the following file:

- l4/sys/pager

14.82 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`.

14.82.1 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.

14.82.2 Member Function Documentation

14.82.2.1 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).

14.82.2.2 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

14.83 L4::IOModifier Class Reference

Modifier class for the IO stream.

Collaboration diagram for L4::IOModifier:

14.83.1 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

14.84 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.

14.84.1 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

14.85 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.

14.85.1 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

14.86 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 >:

14.86.1 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

14.87 L4::Ipc::As_value< T > Struct Template Reference

Pass the argument as plain data value.

Collaboration diagram for L4::Ipc::As_value< T >:

14.87.1 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

14.88 L4::Ipc::Buf_item Class Reference

RPC warpper for a receive item.

Collaboration diagram for L4::Ipc::Buf_item:

14.88.1 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

14.89 L4::Ipc::Call Struct Reference

RPC attribute for a standard RPC call.

Inheritance diagram for L4::Ipc::Call:

Collaboration diagram for L4::Ipc::Call:

14.89.1 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 equvalent to:

```
L4_RPC(long, send, (unsigned value));
```

Definition at line 215 of file ipc_iface.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_iface

14.90 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 >:

14.90.1 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 246 of file ipc_iface.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_iface

14.91 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:

14.91.1 Detailed Description

RPC attribute for an RPC call, with zero send timeout.

Definition at line 225 of file ipc_iface.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_iface

14.92 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)

  Make copy with conversion.
- Cap (L4::Cap< T > cap)

  Make a Cap from L4::Cap<T>, with minimal rights.
- template<typename O >
  Cap (L4::Cap< O > cap)

  Make IPC Cap from L4::Cap with conversion (and minimal rights).
- Cap ()

  Make an invalid cap.
- Cap (L4::Cap< T > cap, unsigned char rights)

  Make a Cap from L4::Cap<T> with the given rights.
- L4::Cap< T > cap () const

  Return the L4::Cap<T> of this Cap.
- unsigned rights () const

  Return the rights bits stored in this IPC cap.
- L4::Ipc::Snd_fpage fpage () const

  Return the send flexpage for this Cap (see l4_fpage_t)
- bool is_valid () const throw ()

  Return true if this Cap is valid.

Static Public Member Functions

- static Cap from_ci (l4_cap_idx_t c)

  Create an IPC capability from a C capability index plus rights.

14.92.1 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.

14.92.2 Member Enumeration Documentation

14.92.2.1 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.

14.92.3 Constructor & Destructor Documentation

14.92.3.1 Cap()

```
template<typename T>
L4::Ipc::Cap< T >::Cap (
            L4::Cap< T > cap,
            unsigned char rights )  [inline]
```

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.

14.92.4 Member Function Documentation

14.92.4.1 from_ci()

```
template<typename T>
static Cap L4::Ipc::Cap< T >::from_ci (
            l4_cap_idx_t c )  [inline], [static]
```

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.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_types

14.93 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

  Check if the capability is valid.
- bool cap_received () const

  Check if the capability has been mapped.
- bool id_received () const

  Check if a label was received instead of a mapping.
- bool local_id_received () const

  Check if a local capability id has been received.
- bool is_compound () const

  Check if the received item has the compound bit set.
- l4_umword_t data () const

  Return the raw flex page descriptor.
- l4_umword_t base_x () const

  Return the raw base descriptor.

14.93.1 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.

14.93.2 Member Function Documentation

14.93.2.1 cap_received()

```
template<typename T>
bool L4::Ipc::Gen_fpage< T >::cap_received ( ) const  [inline]
```

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.

Referenced by L4virtio::Svr::Device_t< Ds_data >::handle_mem_cmd_write(), and L4virtio::Svr::Device_t< Ds↩_data >::mem_info().

Here is the caller graph for this function:

14.93.2.2 id_received()

```
template<typename T>
bool L4::Ipc::Gen_fpage< T >::id_received ( ) const  [inline]
```

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.

Referenced by L4Re::Util::Dataspace_svr::is_static().

Here is the caller graph for this function:

14.93.2.3 is_compound()

```
template<typename T>
bool L4::Ipc::Gen_fpage< T >::is_compound ( ) const  [inline]
```

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.

14.93.2.4 local_id_received()

```
template<typename T>
bool L4::Ipc::Gen_fpage< T >::local_id_received ( ) const  [inline]
```

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

14.94 L4::Ipc::In_out< T > Struct Template Reference

Mark an argument as in-out argument.

Collaboration diagram for L4::Ipc::In_out< T >:

14.94.1 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

14.95 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 receiving to 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)

14.95.1 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.cc.

Definition at line 791 of file ipc_stream.

14.95.2 Constructor & Destructor Documentation

14.95.2.1 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 803 of file ipc_stream.

References L4::Ipc::Istream::tag(), and L4::Ipc::Istream::utcb().

Here is the call graph for this function:

14.95.3 Member Function Documentation

14.95.3.1 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 receiving to the input stream.

Parameters

dst	The destination L4 UID (thread) to call.
timeout	The IPC timeout for the call.
proto	The protocol value to use in the message tag.

Returns

The result dope 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 962 of file ipc_stream.

References l4_ipc_call(), L4_IPC_NEVER, and L4::Ipc::Istream::tag().

Here is the call graph for this function:

14.95.3.2 reply_and_wait() [1/2]

```
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 dope 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 876 of file ipc_stream.

References L4_IPC_SEND_TIMEOUT_0.

14.95.3.3 reply_and_wait() [2/2]

```
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 dope 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 977 of file ipc_stream.

References L4_INVALID_CAP, l4_ipc_reply_and_wait(), l4_ipc_send(), l4_ipc_send_and_wait(), L4_SYSF_REP↩LY, and L4::Ipc::Istream::tag().

Here is the call graph for this function:

14.95.3.4 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 817 of file ipc_stream.

References L4::Ipc::Istream::get(), L4::Ipc::Ostream::put(), 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

14.96 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)

14.96.1 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.

14.96.2 Constructor & Destructor Documentation

14.96.2.1 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.

14.96.3 Member Function Documentation

14.96.3.1 get() [1/3]

```
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.

See Istream::operator>>()

Definition at line 403 of file ipc_stream.

References L4_UNLIKELY.

Referenced by operator>>(), and L4::Ipc::Iostream::reset().

Here is the caller graph for this function:

14.96.3.2 get() [2/3]

```
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).

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 445 of file ipc_stream.

References L4_UNLIKELY.

14.96.3.3 get() [3/3]

```
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.

See Istream::operator>>()

Definition at line 467 of file ipc_stream.

References L4_UNLIKELY, and L4::Ipc::msg_ptr().

Here is the call graph for this function:

14.96.3.4 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 dope (l4_msgtag_t).

This is commonly known as 'closed wait'.

Definition at line 575 of file ipc_stream.

References L4_IPC_NEVER.

Referenced by wait().

Here is the caller graph for this function:

14.96.3.5 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:

14.96.3.6 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 422 of file ipc_stream.

References L4_UNLIKELY.

14.96.3.7 tag() [1/2]

```
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 508 of file ipc_stream.

Referenced by L4::Ipc::Iostream::call(), L4::Ipc::Iostream::Iostream(), operator>>(), and L4::Ipc::Iostream::reply↩_and_wait().

Here is the caller graph for this function:

14.96.3.8 tag() [2/2]

```
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 520 of file ipc_stream.

14.96.3.9 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 551 of file ipc_stream.

References L4_IPC_NEVER.

14.96.3.10 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 dope (l4_msgtag_t).

This wait is actually known as 'open wait'.

Definition at line 1009 of file ipc_stream.

References l4_ipc_receive(), l4_ipc_wait(), and receive().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/ipc_stream

14.97 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 >:

14.97.1 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

14.98 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:

14.98.1 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

14.99 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:

14.99.1 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

14.100 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:

14.100.1 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

14.101 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:

14.101.1 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

14.102 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:

14.102.1 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

14.103 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:

14.103.1 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

14.104 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:

14.104.1 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

14.105 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:

14.105.1 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

14.106 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:

14.106.1 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

14.107 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:

14.107.1 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

14.108 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.

14.108.1 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

14.109 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.

14.109.1 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

14.110 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.

14.110.1 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

14.111 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

14.111.1 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

14.112 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 >:

14.112.1 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

14.113 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 >:

14.113.1 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

14.114 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.

14.114.1 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 241 of file ipc_stream.

14.114.2 Constructor & Destructor Documentation

14.114.2.1 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 252 of file ipc_stream.

The documentation for this class was generated from the following file:

- l4/cxx/ipc_stream

14.115 L4::Ipc::Opt< T > Struct Template Reference

Attribute for defining an optional RPC argument.

Collaboration diagram for L4::lpc::Opt< T >:

Public Member Functions

- Opt ()

  Make an absent optional argument.
- Opt (T value)

  Make a present optional argument with the given value.
- Opt & operator= (T value)

  Assign a value to the optional argument (makes the argument present)
- void set_valid (bool valid=true)

  Set the argument to present or absent.
- T ∗ operator-> ()

  Get the pointer to the value.
- T const ∗ operator-> () const

  Get the const pointer to the value.
- T value () const

  Get the value.
- T & value ()

  Get the value.
- bool is_valid () const

  Get true if present, false if not.

Data Fields

- T _value

  The value.
- bool _valid

  True if the optional argument is present, false else.

14.115.1 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

14.116 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.

14.116.1 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 623 of file ipc_stream.

14.116.2 Member Function Documentation

14.116.2.1 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 660 of file ipc_stream.

Referenced by operator<<(), and L4::Ipc::Iostream::reset().

Here is the caller graph for this function:

14.116.2.2 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 678 of file ipc_stream.

14.116.2.3 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

Syscall return tag.

Definition at line 955 of file ipc_stream.

References L4_IPC_NEVER, l4_ipc_send(), and L4_MSGTAG_FLAGS.

Here is the call graph for this function:

14.116.2.4 tag() [1/2]

```
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 706 of file ipc_stream.

Referenced by operator<<().

Here is the caller graph for this function:

14.116.2.5 tag() [2/2]

```
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 713 of file ipc_stream.

The documentation for this class was generated from the following file:

- l4/cxx/ipc_stream

14.117 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 >:

14.117.1 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

14.118 L4::Ipc::Ret_array< T > Struct Template Reference

Dynamically sized output array of type T.

Collaboration diagram for L4::Ipc::Ret_array< T >:

14.118.1 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

14.119 L4::Ipc::Send_only Struct Reference

RPC attribute for a send-only RPC.

Collaboration diagram for L4::Ipc::Send_only:

14.119.1 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 263 of file ipc_iface.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_iface

14.120 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)

  Create a receive item from a C++ cap.
- Small_buf (l4_cap_idx_t cap, unsigned long flags=0)

  Create a receive item from a C cap.

14.120.1 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.

14.120.2 Constructor & Destructor Documentation

14.120.2.1 Small_buf() [1/2]

```
L4::Ipc::Small_buf::Small_buf (
            L4::Cap< void > cap,
            unsigned long flags = 0 )  [inline], [explicit]
```

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.

14.120.2.2 Small_buf() [2/2]

```
L4::Ipc::Small_buf::Small_buf (
            l4_cap_idx_t cap,
            unsigned long flags = 0 )  [inline], [explicit]
```

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

14.121 L4::Ipc::Snd_item Class Reference

RPC wrapper for a send item.

Collaboration diagram for L4::Ipc::Snd_item:

14.121.1 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

14.122 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.

14.122.1 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 191 of file ipc_stream.

14.122.2 Constructor & Destructor Documentation

14.122.2.1 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 202 of file ipc_stream.

The documentation for this class was generated from the following file:

- l4/cxx/ipc_stream

14.123 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.

14.123.1 Detailed Description

Variably sized RPC argument.

Definition at line 96 of file ipc_varg.

14.123.2 Member Function Documentation

14.123.2.1 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.

14.123.2.2 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.

14.123.2.3 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.

14.123.2.4 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.

14.123.2.5 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.

14.123.2.6 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.

Referenced by L4::Ipc::Varg_list_ref::next(), and L4::Ipc::Varg_list< MAX >::Varg_list().

Here is the caller graph for this function:

14.123.2.7 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.

Referenced by L4::Ipc::Varg_list_ref::next(), and L4::Ipc::Varg_list< MAX >::Varg_list().

Here is the caller graph for this function:

14.123.2.8 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.

14.123.2.9 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

14.124 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.

14.124.1 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 239 of file ipc_varg.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_varg

14.125 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:

Public Member Functions

- Varg_list_ref ()

  Create an empty parameter list.
- Varg_list_ref (void const ∗start, void const ∗end)

  Create a parameter list over a given memory region.
- Varg next ()

  Get the next parameter in the list.

14.125.1 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.

14.125.2 Constructor & Destructor Documentation

14.125.2.1 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 269 of file ipc_varg.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_varg

14.126 L4::Ipc_gate Class Reference

The C++ IPC gate 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

14.126.1 Detailed Description

The C++ IPC gate interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the L4::Factory interface. L4::Ipc_gate::bind_thread() binds an L4::Thread as the receiver of all messages to an IPC gate.

The bind_thread() call allows to assign each IPC gate a kernel protected, machine-word sized payload called a *label*. It securely identifies the gate. The two least significant bits of the *label* are ORed with the L4_CAP_FPA↩GE_S and L4_CAP_FPAGE_W bits stored in the capability when transfered to the receiver. This means the *label* should usually have its two least significant bits set to zero. The *label* is only visible in the L4::Task which is running the thread the IPC gate was bound to and cannot be altered by the sender.

Include File

```
#include <l4/sys/ipc_gate>
```

For the C interface refer to the C IPC-Gate API.

Definition at line 56 of file ipc_gate.

14.126.2 Member Function Documentation

14.126.2.1 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

14.127 L4::Ipc_svr::Br_manager_no_buffers Class Reference

Empty implementation of Server_iface.

Inheritance diagram for L4::lpc_svr::Br_manager_no_buffers:

Collaboration diagram for L4::lpc_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::lpc_svr::Reply_mode)

  Setup wait function for the server loop (Server<>).

Additional Inherited Members

14.127.1 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 216 of file ipc_server_loop.

14.127.2 Member Function Documentation

14.127.2.1 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 223 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

14.128 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:

14.128.1 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

14.129 L4::lpc_svr::Default_loop_hooks Struct Reference

Default LOOP_HOOKS.

Inheritance diagram for L4::lpc_svr::Default_loop_hooks:

Collaboration diagram for L4::Ipc_svr::Default_loop_hooks:

Additional Inherited Members

14.129.1 Detailed Description

Default LOOP_HOOKS.

Combination of Ignore_errors, Default_timeout, Compound_reply, and Br_manager_no_buffers.

Definition at line 268 of file ipc_server_loop.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

14.130 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:

14.130.1 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

14.131 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:

14.131.1 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

14.132 L4::Ipc_svr::Direct_dispatch< R > Struct Template Reference

Direct disptach helper, for forwarding dispatch calls 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

14.132.1 Detailed Description

template<typename R>
struct L4::Ipc_svr::Direct_dispatch< R >

Direct disptach helper, for forwarding dispatch calls 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 139 of file ipc_server_loop.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

14.133 L4::Ipc_svr::Direct_dispatch< R * > Struct Template Reference

Direct disptach 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

14.133.1 Detailed Description

template<typename R>
struct L4::Ipc_svr::Direct_dispatch< R * >

Direct disptach 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 160 of file ipc_server_loop.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

14.134 L4::lpc_svr::Exc_dispatch< R, Exc > Struct Template Reference

Dispatch helper wrapping try {} catch {} around the dispatch call.

Inheritance diagram for L4::lpc_svr::Exc_dispatch< R, Exc >:

Collaboration diagram for L4::lpc_svr::Exc_dispatch< R, Exc >:

Public Member Functions

- Exc_dispatch (R r)

  Make en 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.

14.134.1 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 184 of file ipc_server_loop.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

14.135 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:

14.135.1 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

14.136 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.

14.136.1 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.

14.136.2 Member Function Documentation

14.136.2.1 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::Br_manager_no_buffers, L4::Ipc_svr::Timeout_queue_hooks$<$ HOOKS, BR_MAN $>$, L4::Ipc_svr::Timeout_queue_hooks$<$ Br_manager_timeout_hooks, Br_manager $>$, and L4Re::Util::Br_manager.

Referenced by Server_iface().

Here is the caller graph for this function:

14.136.2.2 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.

Referenced by L4Re::Util::Object_registry::Object_registry(), and Server_iface().

Here is the caller graph for this function:

14.136.2.3 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(), and Server_iface().

Here is the caller graph for this function:

14.136.2.4 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 L4::cap_cast(), and get_rcv_cap().

Here is the call graph for this function:

14.136.2.5 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 L4::Cap_base::cap(), get_rcv_cap(), L4_CAP_MASK, and Server_iface().

Here is the call graph for this function:

14.136.2.6 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.

Referenced by Server_iface().

Here is the caller graph for this function:

14.136.2.7 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::Br_manager_no_buffers, L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >, L4::Ipc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >, and L4Re::Util::Br_manager.

Referenced by Server_iface().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_epiface

14.137 L4::Ipc_svr::Timed_work< HOOKS > Class Template Reference

DEPRECATED.

Inherits HOOKS.

Collaboration diagram for L4::Ipc_svr::Timed_work< HOOKS >:

14.137.1 Detailed Description

template<typename HOOKS>
class L4::Ipc_svr::Timed_work< HOOKS >

DEPRECATED.

Deprecated Use L4::Ipc_svr::Timeout_queue_hooks

Definition at line 97 of file ipc_server_loop.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_server_loop

14.138 L4::Ipc_svr::Timeout Class Reference

Callback interface for Timeout_queue.

Inheritance diagram for L4::lpc_svr::Timeout:

Collaboration diagram for L4::lpc_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.

14.138.1 Detailed Description

Callback interface for Timeout_queue.

Definition at line 28 of file ipc_timeout_queue.

14.138.2 Member Function Documentation

14.138.2.1 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 again within the expired() function.

Referenced by L4::Ipc_svr::Timeout_queue::handle_expired_timeouts(), and Timeout().

Here is the caller graph for this function:

14.138.2.2 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 51 of file ipc_timeout_queue.

Referenced by L4::Ipc_svr::Timeout_queue::remove().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/cxx/ipc_timeout_queue

14.139 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.

14.139.1 Detailed Description

Timeout queue to be used in l4re server loop.

Definition at line 64 of file ipc_timeout_queue.

14.139.2 Member Function Documentation

14.139.2.1 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 120 of file ipc_timeout_queue.

14.139.2.2 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 100 of file ipc_timeout_queue.

References L4::Ipc_svr::Timeout::expired().

Here is the call graph for this function:

14.139.2.3 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 74 of file ipc_timeout_queue.

14.139.2.4 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 135 of file ipc_timeout_queue.

References L4::Ipc_svr::Timeout::timeout().

Here is the call graph for this function:

14.139.2.5 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 90 of file ipc_timeout_queue.

The documentation for this class was generated from the following file:

- l4/cxx/ipc_timeout_queue

14.140 L4::lpc_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::lpc_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

14.140.1 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 159 of file ipc_timeout_queue.

14.140.2 Member Function Documentation

14.140.2.1 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 211 of file ipc_timeout_queue.

14.140.2.2 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 224 of file ipc_timeout_queue.

The documentation for this class was generated from the following file:

- l4/cxx/ipc_timeout_queue

14.141 L4::Irq Class Reference

C++ Irq interface.

Inheritance diagram for L4::Irq:

Collaboration diagram for L4::Irq:

Public Member Functions

- l4_msgtag_t attach (l4_umword_t label, Cap< Thread > const &thread=Cap< Thread >::Invalid, l4_utcb_t *utcb=l4_utcb()) throw ()

  Attach a thread to this interrupt.

- l4_msgtag_t detach (l4_utcb_t *utcb=l4_utcb()) throw ()

  Detach from this interrupt.

- l4_msgtag_t receive (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) throw ()

  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()) throw ()

  Unmask IRQ and (open) wait for any message.
- l4_msgtag_t unmask (l4_utcb_t *utcb=l4_utcb()) throw ()

  Unmask IRQ.

Additional Inherited Members

14.141.1 Detailed Description

C++ Irq interface.

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)

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 112 of file irq.

14.141.2 Member Function Documentation

14.141.2.1 attach()

```
l4_msgtag_t L4::Irq::attach (
            l4_umword_t label,
            Cap< Thread > const & thread = Cap<Thread>::Invalid,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

Attach a thread to this interrupt.

Parameters

label	Identifier of the IRQ (*protected label* used for messages)
thread	Capability of the thread to attach the IRQ to.
utcb	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

The *protected label* is stored in the kernel and sent to the attached thread with the IRQ-triggered notification. It allows the receiver thread to securely identify the IRQ.

Deprecated Use bind_thread().

Definition at line 133 of file irq.

References L4::Kobject::cap(), and l4_irq_attach_u().

Here is the call graph for this function:

14.141.2.2 detach()

```
l4_msgtag_t L4::Irq::detach (
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 151 of file irq.

References L4::Kobject::cap(), and l4_irq_detach_u().

Here is the call graph for this function:

14.141.2.3 receive()

```
l4_msgtag_t L4::Irq::receive (
            l4_timeout_t timeout = L4_IPC_NEVER,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 166 of file irq.

References L4::Kobject::cap(), and l4_irq_receive_u().

Here is the call graph for this function:

14.141.2.4 unmask()

```
l4_msgtag_t L4::Irq::unmask (
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 202 of file irq.

References L4_IPC_NEVER, and L4::Irq_eoi::unmask().

Here is the call graph for this function:

14.141.2.5 wait()

```
l4_msgtag_t L4::Irq::wait (
            l4_umword_t * label,
            l4_timeout_t timeout = L4_IPC_NEVER,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 179 of file irq.

References L4::Irq_eoi::unmask().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/irq

14.142 L4::Irq_eoi Class Reference

Interface for sending an acknowledge 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()) throw ()

  Acknowledge the given interrupt line.

14.142.1 Detailed Description

Interface for sending an acknowledge message to an object.

The object is usually an ICU or an IRQ.

Definition at line 41 of file irq.

14.142.2 Member Function Documentation

14.142.2.1 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() ) throw )   [inline]
```

Acknowledge the given interrupt line.

Parameters

	irqnum	The interrupt line that shall be acknowledged.
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 63 of file irq.

Referenced by L4::Icu::mask(), L4::Irq::unmask(), and L4::Irq::wait().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/irq

14.143 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

14.143.1 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 173 of file ipc_server.

The documentation for this struct was generated from the following file:

- l4/cxx/ipc_server

14.144 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()) throw ()

  Attach an IRQ to this multiplexer.

Additional Inherited Members

14.144.1 Detailed Description

IRQ multiplexer for shared IRQs.

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 acknowledged.

Definition at line 219 of file irq.

14.144.2 Member Function Documentation

14.144.2.1 chain()

```
l4_msgtag_t L4::Irq_mux::chain (
            Cap< Triggerable > const & slave,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 234 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

14.145 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) throw ()

  Initialize memory descriptor.
- unsigned long start () const throw ()

  Return start address of memory descriptor.
- unsigned long end () const throw ()

  Return end address of memory descriptor.
- unsigned long size () const throw ()

  Return size of region described by the memory descriptor.
- Mem_type type () const throw ()

  Return type of the memory descriptor.
- unsigned char sub_type () const throw ()

  Return sub-type of the memory descriptor.
- unsigned is_virtual () const throw ()

  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) throw ()

  Set values of a memory descriptor.

Static Public Member Functions

- static Mem_desc ∗ first (void ∗kip) throw ()

  Get first memory descriptor.
- static unsigned long count (void const ∗kip) throw ()

  Return number of memory descriptors stored in the kernel info page.
- static void count (void ∗kip, unsigned count) throw ()

  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.

14.145.1 Detailed Description

Memory descriptors stored in the kernel interface page.

Include File

```
#include <l4/sys/kip>
```

Definition at line 53 of file kip.

14.145.2 Member Enumeration Documentation

14.145.2.1 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.

14.145.2.2 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.

14.145.3 Constructor & Destructor Documentation

14.145.3.1 Mem_desc()

```
L4::Kip::Mem_desc::Mem_desc (
            unsigned long start,
            unsigned long end,
```

```
            Mem_type t,
            unsigned char st = 0,
            bool virt = false ) throw )   [inline]
```

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.

14.145.4 Member Function Documentation

14.145.4.1 all() [1/2]

```
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:

14.145.4.2 all() [2/2]

```
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:

14.145.4.3 count() [1/2]

```
static unsigned long L4::Kip::Mem_desc::count (
            void const * kip ) throw )   [inline], [static]
```

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:

14.145.4.4 count() [2/2]

```
static void L4::Kip::Mem_desc::count (
            void * kip,
            unsigned count ) throw )   [inline], [static]
```

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:

14.145.4.5 end()

```
unsigned long L4::Kip::Mem_desc::end ( ) const throw )   [inline]
```

Return end address of memory descriptor.

Returns

End address of memory descriptor

Definition at line 184 of file kip.

Referenced by size().

Here is the caller graph for this function:

14.145.4.6 first()

```
static Mem_desc* L4::Kip::Mem_desc::first (
            void * kip ) throw )   [inline], [static]
```

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:

14.145.4.7 is_virtual()

```
unsigned L4::Kip::Mem_desc::is_virtual ( ) const throw )   [inline]
```

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.

14.145.4.8 set()

```
void L4::Kip::Mem_desc::set (
            unsigned long start,
            unsigned long end,
            Mem_type t,
            unsigned char st = 0,
            bool virt = false ) throw )   [inline]
```

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.

14.145.4.9 size()

```
unsigned long L4::Kip::Mem_desc::size ( ) const throw )   [inline]
```

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:

14.145.4.10 start()

```
unsigned long L4::Kip::Mem_desc::start ( ) const throw )   [inline]
```

Return start address of memory descriptor.

Returns

Start address of memory descriptor

Definition at line 177 of file kip.

Referenced by size().

Here is the caller graph for this function:

14.145.4.11 sub_type()

```
unsigned char L4::Kip::Mem_desc::sub_type ( ) const throw )   [inline]
```

Return sub-type of the memory descriptor.

Returns

Sub-type of the memory descriptor

Definition at line 205 of file kip.

14.145.4.12 type()

```
Mem_type L4::Kip::Mem_desc::type ( ) const throw )   [inline]
```

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

14.146 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 throw ()

  Return capability selector.

14.146.1 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.

14.146.2 Member Function Documentation

14.146.2.1 cap()

```
l4_cap_idx_t L4::Kobject::cap ( ) const throw )   [inline], [protected]
```

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.

References L4_CAP_MASK.

Referenced by L4::Irq::attach(), L4::Icu::bind(), L4::Irq_mux::chain(), L4::Factory::create(), L4::Factory::create_factory(), L4::Factory::create_gate(), L4::Factory::create_irq(), L4::Factory::create_task(), L4::Factory::create_thread(), L4::Factory::create_vm(), L4::Irq::detach(), L4::Semaphore::down(), L4::Thread::ex_regs(), L4::Vcon::get_attr(), L4::Icu::info(), L4::Icu::mask(), L4::Thread::modify_senders(), L4::Vcon::read(), L4::Irq::receive(), 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:

14.146.2.2 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.

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.

Definition at line 104 of file kobject.

The documentation for this class was generated from the following file:

- l4/sys/kobject

14.147 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

  Get the capability to ourselves.

Static Protected Member Functions

- static void __check_protocols__ ()

14.147.1 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.

14.147.2 Member Typedef Documentation

14.147.2.1 __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.

14.147.2.2 __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, PRO↩
TO, S_DEMAND >::__Iface_list  [protected]
```

The list of all RPC interfaces provided directly or through inheritance.

Definition at line 849 of file __typeinfo.h.

14.147.2.3 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.

14.147.3 Member Function Documentation

14.147.3.1 __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]
```

Definition at line 852 of file __typeinfo.h.

14.147.3.2 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]
```

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

14.148 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

  Get the capability to ourselves.

Static Protected Member Functions

- static void __check_protocols__ ()

14.148.1 Detailed Description

template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_ANY, typename S_D↩EMAND = 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 935 of file __typeinfo.h.

14.148.2 Member Typedef Documentation

14.148.2.1 __Iface

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROT↩
O_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 941 of file __typeinfo.h.

14.148.2.2 __Iface_list

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROT↩
O_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 952 of file __typeinfo.h.

14.148.2.3 Class

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROT↩
O_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 939 of file __typeinfo.h.

14.148.3 Member Function Documentation

14.148.3.1 __check_protocols__()

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROT↩
O_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]
```

Definition at line 955 of file __typeinfo.h.

14.148.3.2 c()

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROT↩
O_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]
```

Get the capability to ourselves.

Definition at line 983 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

14.149 L4::Kobject_demand< T > Struct Template Reference

Get the combined server-side resource requirements for all type T...

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Kobject_demand< T >:

14.149.1 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 1033 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

14.150 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

  Get the capability to ourselves.

Static Protected Member Functions

- static void __check_protocols__ ()

  Helper to check for protocol conflicts.

14.150.1 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

14.151 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 ()

  Get a pointer to teh Kobject type information of T.
- static Type_info::Demand demand ()

  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.

14.151.1 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.

14.151.2 Member Typedef Documentation

14.151.2.1 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.

14.151.3 Member Function Documentation

14.151.3.1 demand()

```
template<typename T>
static Type_info::Demand L4::Kobject_typeid< T >::demand ( )  [inline], [static]
```

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.

14.151.3.2 id()

```
template<typename T>
static Type_info const* L4::Kobject_typeid< T >::id ( )  [inline], [static]
```

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:

14.151.3.3 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

T↩ HIS	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

Parameters

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< Kobject >::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

14.152 L4::Kobject_typeid< void > Struct Template Reference

Minimalistic ID for `void` interface.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Kobject_typeid< void >:

14.152.1 Detailed Description

template<>
struct L4::Kobject_typeid< void >

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

14.153 L4::Kobject_x< Derived, ARGS > Struct Template Reference

Generic Kobject inheritance template.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject_x< Derived, ARGS >:

Collaboration diagram for L4::Kobject_x< Derived, ARGS >:

14.153.1 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 1195 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

14.154 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

14.154.1 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.

14.154.2 Member Function Documentation

14.154.2.1 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.

14.154.2.2 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.

14.154.2.3 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.

Referenced by L4::cap_dynamic_cast().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/meta

14.155 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

14.155.1 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

14.156 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::Opt< l4_mword_t &> result, L4::Ipc::↩ Rcv_fpage rwin, L4::Ipc::Opt< L4::Ipc::Snd_fpage &> fp)

  Page-fault protocol message.

Additional Inherited Members

14.156.1 Detailed Description

Pager interface including the Io_pager interface.

Definition at line 67 of file pager.

14.156.2 Member Function Documentation

14.156.2.1 page_fault()

```
l4_msgtag_t L4::Pager::page_fault (
            l4_umword_t pfa,
            l4_umword_t pc,
            L4::Ipc::Opt< l4_mword_t &> result,
            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.
out	*result*	Optional: handling result value.
	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 funtion that fills in `result` and / or `fp` for the reply.

`pfa` encoding is as shown:

[63/31 .. 3]	2	1	0
PFA	X	W	T

- **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` is set to 1 for a page fault during instruction fetch.
- **W** Bit 1 of `pfa` is set to 1 for a page fault due to a write operation.
- **T** Bit 0 of `pfa` is set for translation faults (no mapping was present).

The documentation for this class was generated from the following file:

- l4/sys/pager

14.157 L4::Platform_control Class Reference

L4 C++ interface for controlling platform-wide properties.

Inheritance diagram for L4::Platform_control:

Collaboration diagram for L4::Platform_control:

Public Types

- enum Opcode { Suspend = L4_PLATFORM_CTL_SYS_SUSPEND_OP, Shutdown = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP, Cpu_enable = L4_PLATFORM_CTL_CPU_ENABLE_OP, Cpu_disable = L4_PLATFORM_CTL_CPU_DISABLE_OP }

  Opcodes for platform-control object.

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_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

14.157.1 Detailed Description

L4 C++ interface for controlling platform-wide properties.

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 46 of file platform_control.

14.157.2 Member Enumeration Documentation

14.157.2.1 Opcode

```
enum L4::Platform_control::Opcode
```

Opcodes for platform-control object.

Enumerator

Suspend	Opcode for suspend to RAM.
Shutdown	Opcode for shutdown / reboot.
Cpu_enable	Opcode to enable a CPU.
Cpu_disable	Opcode to disable a CPU.

Definition at line 51 of file platform_control.

14.157.3 Member Function Documentation

14.157.3.1 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

14.157.3.2 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

14.157.3.3 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

14.157.3.4 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.

The documentation for this class was generated from the following file:

- l4/sys/platform_control

14.158 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.

14.158.1 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 37 of file poll_timeout_kipclock.

14.158.2 Constructor & Destructor Documentation

14.158.2.1 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 44 of file poll_timeout_kipclock.

14.158.3 Member Function Documentation

14.158.3.1 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 53 of file poll_timeout_kipclock.

References l4_kip_clock(), and l4re_kip().

Here is the call graph for this function:

14.158.3.2 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 67 of file poll_timeout_kipclock.

References l4_kip_clock(), and l4re_kip().

Here is the call graph for this function:

14.158.3.3 timed_out()

```
bool L4::Poll_timeout_kipclock::timed_out ( ) const  [inline]
```

Query whether timeout has expired.

Returns

Expiry state of timeout

Definition at line 79 of file poll_timeout_kipclock.

The documentation for this class was generated from the following file:

- l4/re/util/poll_timeout_kipclock

14.159 L4::Proto_t< P > Struct Template Reference

Data type for defining protocol numbers.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Proto_t< P >:

14.159.1 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 1179 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

- l4/sys/__typeinfo.h

14.160 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 (Ipc::Opt< Ipc::Cap< Thread > > t, l4_umword_t label)

  Bind a thread to an IPC receive endpoint.

Additional Inherited Members

14.160.1 Detailed Description

Interface for kernel objects that allow to receive IPC from them.

Such an object is for example an Ipc_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.

14.160.2 Member Function Documentation

14.160.2.1 bind_thread()

```
l4_msgtag_t L4::Rcv_endpoint::bind_thread (
            Ipc::Opt< 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	`t` is missing L4_CAP_FPAGE_S right.

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

14.161 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.

14.161.1 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 305 of file ipc_epiface.

14.161.2 Member Function Documentation

14.161.2.1 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.

Referenced by L4Re::Util::Object_registry::register_irq_obj(), and L4virtio::Svr::Block_dev< Ds_data >::register_obj().

Here is the caller graph for this function:

14.161.2.2 register_obj() [1/3]

```
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.

Referenced by L4virtio::Svr::Block_dev< Ds_data >::register_obj().

Here is the caller graph for this function:

14.161.2.3 register_obj() [2/3]

```
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.

14.161.2.4 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.

14.161.2.5 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

14.162 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

14.162.1 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.

14.162.2 Constructor & Destructor Documentation

14.162.2.1 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.

14.162.3 Member Function Documentation

14.162.3.1 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.

14.162.3.2 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

14.163 L4::Scheduler Class Reference

C++ interface of the Scheduler kernel object.

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_utcb_t *utcb=l4_utcb()) const throw ()

  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 throw ()

  Query if a CPU is online.

Additional Inherited Members

14.163.1 Detailed Description

C++ interface of the Scheduler kernel object.

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.

Include File

```
#include <l4/sys/scheduler>
```

Definition at line 42 of file scheduler.

14.163.2 Member Function Documentation

14.163.2.1 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.

14.163.2.2 info()

```
l4_msgtag_t L4::Scheduler::info (
            l4_umword_t * cpu_max,
            l4_sched_cpu_set_t * cpus,
            l4_utcb_t * utcb = l4_utcb() ) const throw )   [inline]
```

Get scheduler information.

Parameters

out	*cpu_max*	Maximum number of CPUs ever available.
in,out	*cpus*	*cpus.offset* is first CPU of interest. *cpus.granularity* (see l4_sched_cpu_set_t). *cpus.map* Bitmap of online CPUs.
	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 66 of file scheduler.

References l4_sched_cpu_set_t::gran_offset, L4_INLINE_RPC_OP, L4_SCHEDULER_IDLE_TIME_OP, L4_SCHEDULER_RUN_THREAD_OP, l4_sched_cpu_set_t::map, and cxx::max().

Here is the call graph for this function:

14.163.2.3 is_online()

```
bool L4::Scheduler::is_online (
            l4_umword_t cpu,
            l4_utcb_t * utcb = l4_utcb() ) const throw )   [inline]
```

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 139 of file scheduler.

14.163.2.4 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

14.164 L4::Semaphore Struct Reference

Kernel-provided semaphore object.

Inheritance diagram for L4::Semaphore:

Collaboration diagram for L4::Semaphore:

Public Member Functions

- l4_msgtag_t up (l4_utcb_t *utcb=l4_utcb()) throw ()

  Semaphore up operation (wrapper for trigger()).
- l4_msgtag_t down (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) throw ()

  Semaphore down operation.

Additional Inherited Members

14.164.1 Detailed Description

Kernel-provided semaphore object.

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 sucessfully, 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 51 of file semaphore.

14.164.2 Member Function Documentation

14.164.2.1 down()

```
l4_msgtag_t L4::Semaphore::down (
            l4_timeout_t timeout = L4_IPC_NEVER,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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

IPC message return tag. Use l4_ipc_error() to check for a timeout or a cancel condition.

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 84 of file semaphore.

References L4::Kobject::cap().

Here is the call graph for this function:

14.164.2.2 up()

```
l4_msgtag_t L4::Semaphore::up (
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 65 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

14.165 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 *utcb)

  Initializes the server loop.
- template<typename DISPATCH >
  L4_NORETURN void internal_loop (DISPATCH dispatch)

  The server loop.
- template<typename R >
  L4_NORETURN void loop_noexc (R r)

  Server loop without exception handling.
- template<typename EXC , typename R >
  L4_NORETURN void loop (R r)

  Server loop with internal exception handling.

Protected Member Functions

- l4_msgtag_t reply_n_wait (l4_msgtag_t reply, l4_umword_t *p)

  Internal implementation for reply and wait.

14.165.1 Detailed Description

template<typename LOOP_HOOKS = Ipc_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 Ipc_svr::Default_loop_hooks, Ipc_svr::Ignore_errors, Ipc_svr::Default_timeout, Ipc_svr::Compound_reply, and Ipc_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 290 of file ipc_server_loop.

14.165.2 Constructor & Destructor Documentation

14.165.2.1 Server()

```
template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
L4::Server< LOOP_HOOKS >::Server (
            l4_utcb_t * utcb )  [inline], [explicit]
```

Initializes the server loop.

Parameters

utcb	The UTCB of the thread running the server loop.

Definition at line 298 of file ipc_server_loop.

References L4_NORETURN.

14.165.3 Member Function Documentation

14.165.3.1 internal_loop()

```
template<typename L >
template<typename DISPATCH >
L4_NORETURN void L4::Server< L >::internal_loop (
            DISPATCH dispatch )  [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 360 of file ipc_server_loop.

References l4_msgtag_t::has_error(), L4_ENOREPLY, and l4_msgtag().

Here is the call graph for this function:

14.165.3.2 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 )  [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 323 of file ipc_server_loop.

The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_server_loop

14.166 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.

14.166.1 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 50 of file ipc_server.

14.166.2 Member Function Documentation

14.166.2.1 dispatch()

```
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 >.

The documentation for this class was generated from the following file:

- l4/cxx/ipc_server

14.167 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.

14.167.1 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 92 of file ipc_server.

14.167.2 Member Function Documentation

14.167.2.1 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 111 of file ipc_server.

14.167.2.2 get_buffer_demand()

```
template<typename IFACE, typename BASE = L4::Server_object>
BASE::Demand L4::Server_object_t< IFACE, BASE >::get_buffer_demand ( ) const  [inline]
```

Returns

the server-side buffer demand based in *IFACE*.

Definition at line 98 of file ipc_server.

14.167.2.3 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 126 of file ipc_server.

Referenced by L4::Server_object_x< Derived, IFACE, BASE >::dispatch().

Here is the caller graph for this function:

The documentation for this struct was generated from the following file:

- l4/cxx/ipc_server

14.168 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

14.168.1 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 155 of file ipc_server.

The documentation for this struct was generated from the following file:

- l4/cxx/ipc_server

14.169 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) throw ()

  Internal constructor, use to generate a capability from a `this` *pointer.*

- Cap< T > operator-> () const throw ()

  Member access of a `T`*.*

Additional Inherited Members

14.169.1 Detailed Description

template<typename T, typename SMART>
class L4::Smart_cap< T, SMART >

Smart capability class.

Definition at line 36 of file smart_capability.

14.169.2 Constructor & Destructor Documentation

14.169.2.1 Smart_cap()

```
template<typename T, typename SMART>
template<typename O >
L4::Smart_cap< T, SMART >::Smart_cap (
            Cap< O > const & p ) throw )   [inline]
```

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.

References L4::Cap_base::_c, L4::Cap_base::cap(), and L4::Cap_base::Cap_base().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/sys/smart_capability

14.170 L4::String Class Reference

A null-terminated string container class.

```
#include <string.h>
```

Collaboration diagram for L4::String:

14.170.1 Detailed Description

A null-terminated string container class.

Definition at line 35 of file string.h.

The documentation for this class was generated from the following file:

- l4/cxx/string.h

14.171 L4::Task Class Reference

C++ interface of the Task kernel object.

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_addr_t snd_base, l4_utcb_t *utcb=l4_utcb()) throw ()

  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()) throw ()

  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()) throw ()

  Revoke rights from a task.
- l4_msgtag_t delete_obj (L4::Cap< void > obj, l4_utcb_t *utcb=l4_utcb()) throw ()

  Release capability and delete object.

- l4_msgtag_t release_cap (L4::Cap< void > cap, l4_utcb_t *utcb=l4_utcb()) throw ()

  Release capability.
- l4_msgtag_t cap_valid (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb()) throw ()

  Check whether a capability is present (refers to an object).
- l4_msgtag_t cap_has_child (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb()) throw ()

  Test whether a capability has child mappings (in another task).
- l4_msgtag_t cap_equal (Cap< void > const &cap_a, Cap< void > const &cap_b, l4_utcb_t *utcb=l4_utcb()) throw ()

  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()) throw ()

  Add kernel-user memory.

Additional Inherited Members

14.171.1 Detailed Description

C++ interface of the Task kernel object.

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 43 of file task.

14.171.2 Member Function Documentation

14.171.2.1 add_ku_mem()

```
l4_msgtag_t L4::Task::add_ku_mem (
            l4_fpage_t const & fpage,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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.

14.171.2.2 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() ) throw )   [inline]
```

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.

14.171.2.3 cap_has_child()

```
l4_msgtag_t L4::Task::cap_has_child (
            Cap< void > const & cap,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

Test whether a capability has child mappings (in another task).

Parameters

cap	Capability selector to look up in the destination task.
utcb	UTCB pointer of the calling thread.

Return values

tag.label()	= 1: The capability has at least on child mapping in another task.
tag.label()	= 0: No child mappings.

Deprecated Do not use. Undetermined future, might be removed.

Definition at line 179 of file task.

14.171.2.4 cap_valid()

```
l4_msgtag_t L4::Task::cap_valid (
            Cap< void > const & cap,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

Check whether a capability is present (refers to an object).

Parameters

cap	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.

Definition at line 163 of file task.

14.171.2.5 delete_obj()

```
l4_msgtag_t L4::Task::delete_obj (
            L4::Cap< void > obj,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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

The object will be deleted if the `obj` has sufficient rights. No error will be reported if the rights are insufficient, however, the capability is removed in all cases.

Definition at line 133 of file task.

14.171.2.6 map()

```
l4_msgtag_t L4::Task::map (
            Cap< Task > const & src_task,
            l4_fpage_t const & snd_fpage,
            l4_addr_t snd_base,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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.
utcb	UTCB pointer of the calling thread.

Returns

Syscall return tag.

This method allows for asynchronous rights delegation from one task to another. It can be used to share memory as well as to delegate access to objects. The destination task is the task referenced by the capability invoking map and the receive window is the whole address space of said task.

Definition at line 67 of file task.

14.171.2.7 release_cap()

```
l4_msgtag_t L4::Task::release_cap (
            L4::Cap< void > cap,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

Release capability.

Parameters

cap	Capability selector to release.
utcb	UTCB pointer of the calling thread.

Returns

Syscall return tag.

This operation unmaps the capability from `this` task.

Definition at line 147 of file task.

14.171.2.8 unmap()

```
l4_msgtag_t L4::Task::unmap (
            l4_fpage_t const & fpage,
            l4_umword_t map_mask,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

Revoke rights from the task.

Parameters

fpage	Flexpage that describes an area in the address space or object 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 and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Note

If the capability possesses delete rights or if it is the last capability pointing to the object, calling this function might destroy the object itself.

Definition at line 90 of file task.

Referenced by L4Re::Rm::attach().

Here is the caller graph for this function:

14.171.2.9 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() ) throw )   [inline]
```

Revoke rights from a task.

Parameters

fpages	An array of flexpages. Each item describes an area in the address or object 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.

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Precondition

The caller needs to take care that `num_fpages` is not bigger than L4_UTCB_GENERIC_DATA_SIZE - 2.

Note

If the capability possesses delete rights or if it is the last capability pointing to the object, calling this function might destroy the object itself.

Definition at line 115 of file task.

The documentation for this class was generated from the following file:

- l4/sys/task

14.172 L4::Thread Class Reference

C++ L4 kernel thread interface.

Inheritance diagram for L4::Thread:

Collaboration diagram for L4::Thread:

Data Structures

- class Attr

  Thread attributes used for control_commit().
- class Modify_senders

  Wrapper class for modifying senders.

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()) throw ()

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()) throw ()

  Exchange basic thread registers and return previous values.
- l4_msgtag_t control (Attr const &attr) throw ()

  Commit the given thread-attributes object.
- l4_msgtag_t switch_to (l4_utcb_t *utcb=l4_utcb()) throw ()

  Switch execution to this thread.
- l4_msgtag_t stats_time (l4_kernel_clock_t *us, l4_utcb_t *utcb=l4_utcb()) throw ()

  Get consumed time of thread in us.
- l4_msgtag_t vcpu_resume_start (l4_utcb_t *utcb=l4_utcb()) throw ()

  vCPU resume, start.
- l4_msgtag_t vcpu_resume_commit (l4_msgtag_t tag, l4_utcb_t *utcb=l4_utcb()) throw ()

  vCPU resume, commit.
- l4_msgtag_t vcpu_control (l4_addr_t vcpu_state, l4_utcb_t *utcb=l4_utcb()) throw ()

  Enable or disable the vCPU feature for the thread.
- l4_msgtag_t vcpu_control_ext (l4_addr_t ext_vcpu_state, l4_utcb_t *utcb=l4_utcb()) throw ()

  Enable or disable the extended vCPU feature for the thread.
- l4_msgtag_t register_del_irq (Cap< Irq > irq, l4_utcb_t *u=l4_utcb()) throw ()

  Register an IRQ that will trigger upon deletion events.
- l4_msgtag_t modify_senders (Modify_senders const &todo) throw ()

  Apply sender modifiction rules.

Additional Inherited Members

14.172.1 Detailed Description

C++ L4 kernel thread 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.

14.172.2 Member Function Documentation

14.172.2.1 control()

```
l4_msgtag_t L4::Thread::control (
            Attr const & attr ) throw )   [inline]
```

Commit the given thread-attributes object.

Parameters

attr	the attribute object to commit to the thread.

Definition at line 215 of file thread.

14.172.2.2 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() ) throw )   [inline]
```

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 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`).

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 85 of file thread.

References L4::Kobject::cap(), and l4_thread_ex_regs_u().

Here is the call graph for this function:

14.172.2.3 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() ) throw )   [inline]
```

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`).

Definition at line 111 of file thread.

References L4::Kobject::cap(), and l4_thread_ex_regs_ret_u().

Here is the call graph for this function:

14.172.2.4 modify_senders()

```
l4_msgtag_t L4::Thread::modify_senders (
            Modify_senders const & todo ) throw )   [inline]
```

Apply sender modifiction rules.

Parameters

todo	Prepared sender modification rules.

Returns

System call return tag.

Definition at line 373 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:

14.172.2.5 register_del_irq()

```
l4_msgtag_t L4::Thread::register_del_irq (
            Cap< Irq > irq,
            l4_utcb_t * u = l4_utcb() ) throw )   [inline]
```

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.

An example of a deletion event is the removal of an IPC gate that is bound to this thread.

See also

l4_thread_register_del_irq

Definition at line 313 of file thread.

14.172.2.6 stats_time()

```
l4_msgtag_t L4::Thread::stats_time (
            l4_kernel_clock_t * us,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 236 of file thread.

14.172.2.7 switch_to()

```
l4_msgtag_t L4::Thread::switch_to (
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

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 225 of file thread.

14.172.2.8 vcpu_control()

```
l4_msgtag_t L4::Thread::vcpu_control (
            l4_addr_t vcpu_state,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

Enable or disable the vCPU feature for the thread.

Parameters

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.

This function enables the vCPU feature of `this` thread if `vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `vcpu_state` is 0. (Disable: optional, currently unsupported.)

Definition at line 272 of file thread.

References L4::Kobject::cap(), and l4_thread_vcpu_control_u().

Here is the call graph for this function:

14.172.2.9 vcpu_control_ext()

```
l4_msgtag_t L4::Thread::vcpu_control_ext (
            l4_addr_t ext_vcpu_state,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

Enable or disable 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 if `ext_vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `ext_vcpu_state` is 0.

Note

The extended vCPU mode includes the normal vCPU mode.

Definition at line 296 of file thread.

References L4::Kobject::cap(), and l4_thread_vcpu_control_ext_u().

Here is the call graph for this function:

14.172.2.10 vcpu_resume_commit()

```
l4_msgtag_t L4::Thread::vcpu_resume_commit (
            l4_msgtag_t tag,
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

vCPU resume, commit.

See also

l4_thread_vcpu_resume_commit

Definition at line 253 of file thread.

14.172.2.11 vcpu_resume_start()

```
l4_msgtag_t L4::Thread::vcpu_resume_start (
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

vCPU resume, start.

See also

l4_thread_vcpu_resume_start

Definition at line 245 of file thread.

The documentation for this class was generated from the following file:

- l4/sys/thread

14.173 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()) throw ()

  Create a thread-attribute object with the given UTCB.
- void pager (Cap< void > const &pager) throw ()

  Set the pager capability selector.
- Cap< void > pager () throw ()

  Get the capability selector used for page-fault messages.
- void exc_handler (Cap< void > const &exc_handler) throw ()

  Set the exception-handler capability selector.
- Cap< void > exc_handler () throw ()

  Get the capability selector used for exception messages.
- void bind (l4_utcb_t *thread_utcb, Cap< Task > const &task) throw ()

  Bind the thread to a task.
- void alien (int on) throw ()

  Set the thread to alien mode.
- void ux_host_syscall (int on) throw ()

  Allow host system calls on Fiasco-UX.

Friends

- class **L4::Thread**

14.173.1 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.

See also

Thread control for some more details.

Definition at line 125 of file thread.

14.173.2 Constructor & Destructor Documentation

14.173.2.1 Attr()

```
L4::Thread::Attr::Attr (
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline], [explicit]
```

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 138 of file thread.

14.173.3 Member Function Documentation

14.173.3.1 bind()

```
void L4::Thread::Attr::bind (
            l4_utcb_t * thread_utcb,
            Cap< Task > const & task ) throw )   [inline]
```

Bind the thread to a task.

Parameters

thread_utcb	The UTCB address of the thread within the task specified by `task`.
task	The capability selector for the task the thread shall be bound to.

Binding a thread to a task means that the thread shall afterwards execute in the given task. To actually start execution you need to use L4::Thread::ex_regs().

Definition at line 191 of file thread.

14.173.3.2 exc_handler() [1/2]

```
void L4::Thread::Attr::exc_handler (
            Cap< void > const & exc_handler ) throw )   [inline]
```

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 167 of file thread.

14.173.3.3 exc_handler() [2/2]

```
Cap<void> L4::Thread::Attr::exc_handler ( ) throw )   [inline]
```

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 176 of file thread.

14.173.3.4 pager() [1/2]

```
void L4::Thread::Attr::pager (
            Cap< void > const & pager ) throw )   [inline]
```

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 148 of file thread.

14.173.3.5 pager() [2/2]

```
Cap<void> L4::Thread::Attr::pager ( ) throw )   [inline]
```

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 157 of file thread.

14.173.3.6 ux_host_syscall()

```
void L4::Thread::Attr::ux_host_syscall (
            int on ) throw )   [inline]
```

Allow host system calls on Fiasco-UX.

Precondition

Running on Fiasco-UX.

Definition at line 205 of file thread.

The documentation for this class was generated from the following file:

- l4/sys/thread

14.174 L4::Thread::Modify_senders Class Reference

Wrapper class for modifying senders.

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) throw ()

  Add a rule.

14.174.1 Detailed Description

Wrapper class for modifying senders.

Use the add() function to add modification rules, and use modify_senders() to commit. Do not use the UTCB inbetween as it is used by add() and modify_senders().

Definition at line 323 of file thread.

14.174.2 Member Function Documentation

14.174.2.1 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 ) throw )   [inline]
```

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 sucess, <0 on error

Only the first match is applied.

See also

l4_thread_modify_sender_add()

Definition at line 352 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

14.175 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()) throw ()

  Trigger.

Additional Inherited Members

14.175.1 Detailed Description

Interface that allows an object to be triggered by some source.

This interface is usually used in conjunction with L4::Icu.

Definition at line 76 of file irq.

14.175.2 Member Function Documentation

14.175.2.1 trigger()

```
l4_msgtag_t L4::Triggerable::trigger (
            l4_utcb_t * utcb = l4_utcb() ) throw )   [inline]
```

Trigger.

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 91 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

14.176 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.

14.176.1 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

14.177 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)

  Make Demand object.
- bool no_demand () const
- Demand operator| (Demand const &rhs) const

  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.

14.177.1 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.

14.177.2 Constructor & Destructor Documentation

14.177.2.1 Demand()

```
L4::Type_info::Demand::Demand (
            unsigned char caps = 0,
            unsigned char flags = 0,
            unsigned char mem = 0,
            unsigned char ports = 0 )  [inline], [explicit]
```

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.

14.177.3 Member Function Documentation

14.177.3.1 no_demand()

```
bool L4::Type_info::Demand::no_demand ( ) const  [inline]
```

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

14.178 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

14.178.1 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.

14.178.2 Member Enumeration Documentation

14.178.2.1 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

14.179 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

14.179.1 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

14.180 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 >:

14.180.1 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

14.181 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 >:

14.181.1 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

14.182 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.

14.182.1 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

14.183 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 >:

14.183.1 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

14.184 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:

14.184.1 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

14.185 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 >:

14.185.1 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

14.186 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 >:

14.186.1 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

14.187 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 >:

14.187.1 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

14.188 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 >:

14.188.1 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

14.189 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

14.189.1 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

14.190 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 >:

14.190.1 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

14.191 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 >:

14.191.1 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

14.192 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.

14.192.1 Detailed Description

template<bool V>
struct L4::Types::Bool< V >

Boolean meta type.

Template Parameters

V	The boolean value

Definition at line 165 of file types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/types

14.193 L4::Types::False Struct Reference

False meta value.

Inheritance diagram for L4::Types::False:

Collaboration diagram for L4::Types::False:

Additional Inherited Members

14.193.1 Detailed Description

False meta value.

Definition at line 173 of file types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/types

14.194 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.

14.194.1 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_E↩NUM 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.

14.194.2 Member Enumeration Documentation

14.194.2.1 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.

14.194.3 Constructor & Destructor Documentation

14.194.3.1 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.

14.194.3.2 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.

14.194.4 Member Function Documentation

14.194.4.1 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:

14.194.4.2 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

14.195 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

14.195.1 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 189 of file types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/types

14.196 L4::Types::True Struct Reference

True meta value.

Inheritance diagram for L4::Types::True:

Collaboration diagram for L4::Types::True:

Additional Inherited Members

14.196.1 Detailed Description

True meta value.

Definition at line 177 of file types.

The documentation for this struct was generated from the following file:

- l4/sys/cxx/types

14.197 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

14.197.1 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

14.198 L4::Vcon Class Reference

C++ L4 Vcon 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 throw ()

  Send data to `this` virtual console.
- long write (char const ∗buf, unsigned size, l4_utcb_t ∗utcb=l4_utcb()) const throw ()

  Write data to `this` virtual console.
- int read (char ∗buf, unsigned size, l4_utcb_t ∗utcb=l4_utcb()) const throw ()

Read data from `this` virtual console.

- int read_with_flags (char *buf, unsigned size, l4_utcb_t *utcb=l4_utcb()) const throw ()

  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 throw ()

  Set the attributes of `this` virtual console.

- l4_msgtag_t get_attr (l4_vcon_attr_t *attr, l4_utcb_t *utcb=l4_utcb()) const throw ()

  Get attributes of `this` virtual console.

Additional Inherited Members

14.198.1 Detailed Description

C++ L4 Vcon 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.

Include File

```
#include <l4/sys/vcon>
```

See the Virtual Console for the C interface.

Definition at line 43 of file vcon.

14.198.2 Member Function Documentation

14.198.2.1 get_attr()

```
l4_msgtag_t L4::Vcon::get_attr (
            l4_vcon_attr_t * attr,
            l4_utcb_t * utcb = l4_utcb() ) const throw )   [inline]
```

Get attributes of `this` virtual console.

Parameters

out	*attr*	Attribute structure. Contains the attributes after a successfull call of this function.
	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_get_attr_u().

Here is the call graph for this function:

14.198.2.2 read()

```
int L4::Vcon::read (
            char * buf,
            unsigned size,
            l4_utcb_t * utcb = l4_utcb() ) const throw )   [inline]
```

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 94 of file vcon.

References L4::Kobject::cap(), and l4_vcon_read_u().

Here is the call graph for this function:

14.198.2.3 read_with_flags()

```
int L4::Vcon::read_with_flags (
            char * buf,
            unsigned size,
            l4_utcb_t * utcb = l4_utcb() ) const throw )   [inline]
```

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 119 of file vcon.

14.198.2.4 send()

```
l4_msgtag_t L4::Vcon::send (
            char const * buf,
            unsigned size,
            l4_utcb_t * utcb = l4_utcb() ) const throw )   [inline]
```

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 63 of file vcon.

References L4::Kobject::cap(), and l4_vcon_send_u().

Here is the call graph for this function:

14.198.2.5 set_attr()

```
l4_msgtag_t L4::Vcon::set_attr (
            l4_vcon_attr_t const * attr,
            l4_utcb_t * utcb = l4_utcb() ) const throw )   [inline]
```

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 132 of file vcon.

References L4::Kobject::cap(), and l4_vcon_set_attr_u().

Here is the call graph for this function:

14.198.2.6 write()

```
long L4::Vcon::write (
            char const * buf,
            unsigned size,
            l4_utcb_t * utcb = l4_utcb() ) const throw )   [inline]
```

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 77 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

14.199 L4::Vm Class Reference

Virtual machine.

Inheritance diagram for L4::Vm:

Collaboration diagram for L4::Vm:

Additional Inherited Members

14.199.1 Detailed Description

Virtual machine.

Definition at line 35 of file vm.

The documentation for this class was generated from the following file:

- l4/sys/vm

14.200 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.

14.200.1 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

14.201 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 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

14.201.1 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.

14.201.2 Field Documentation

14.201.2.1 flags

```
l4_umword_t l4_exc_regs_t::flags
```

rflags

eflags

Definition at line 80 of file utcb.h.

14.201.2.2 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

14.202 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.

14.202.1 Detailed Description

L4 flexpage type.

Definition at line 81 of file __l4_fpage.h.

The documentation for this union was generated from the following file:

- l4/sys/__l4_fpage.h

14.203 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,.

14.203.1 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 159 of file icu.h.

14.203.2 Field Documentation

14.203.2.1 features

```
unsigned l4_icu_info_t::features
```

Feature flags.

If L4_ICU_FLAG_MSI is set the ICU supports MSIs.

Definition at line 166 of file icu.h.

The documentation for this struct was generated from the following file:

- l4/sys/icu.h

14.204 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.

14.204.1 Detailed Description

Info to use for a specific MSI.

Definition at line 180 of file icu.h.

14.204.2 Field Documentation

14.204.2.1 msi_data

```
l4_uint32_t l4_icu_msi_info_t::msi_data
```

Value to use as data written to msi_addr, when sending this MSI.

Definition at line 185 of file icu.h.

The documentation for this struct was generated from the following file:

- l4/sys/icu.h

14.205 l4_kernel_info_mem_desc_t Struct Reference

Memory descriptor data structure.

```
#include <memdesc.h>
```

Collaboration diagram for l4_kernel_info_mem_desc_t:

14.205.1 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

14.206 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

14.206.1 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

14.207 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.

14.207.1 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

14.208 l4_msgtag_t Struct Reference

Message tag data structure.

```
#include <types.h>
```

Collaboration diagram for l4_msgtag_t:

Public Member Functions

- long label () const throw ()

  Get the protocol value.
- void label (long v) throw ()

  Set the protocol value.
- unsigned words () const throw ()

  Get the number of untyped words.
- unsigned items () const throw ()

  Get the number of typed items.
- unsigned flags () const throw ()

  Get the flags value.
- bool is_page_fault () const throw ()

  Test if protocol indicates page-fault protocol.
- bool is_preemption () const throw ()

  Test if protocol indicates preemption protocol.
- bool is_sys_exception () const throw ()

  Test if protocol indicates system-exception protocol.
- bool is_exception () const throw ()

  Test if protocol indicates exception protocol.
- bool is_sigma0 () const throw ()

  Test if protocol indicates sigma0 protocol.
- bool is_io_page_fault () const throw ()

  Test if protocol indicates IO-page-fault protocol.
- unsigned has_error () const throw ()

  Test if flags indicate an error.

Data Fields

- l4_mword_t raw

  raw value

14.208.1 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 159 of file types.h.

14.208.2 Member Function Documentation

14.208.2.1 flags()

```
unsigned l4_msgtag_t::flags ( ) const throw )   [inline]
```

Get the flags value.

The flags are a combination of the flags defined by l4_msgtag_flags.

Definition at line 177 of file types.h.

Referenced by has_error().

Here is the caller graph for this function:

The documentation for this struct was generated from the following file:

- l4/sys/types.h

14.209 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.

14.209.1 Detailed Description

CPU sets.

Examples:

examples/sys/migrate/thread_migrate.cc.

Definition at line 44 of file scheduler.h.

14.209.2 Member Function Documentation

14.209.2.1 granularity()

```
unsigned char l4_sched_cpu_set_t::granularity ( ) const  [inline]
```

Returns

Get granularity value

Definition at line 67 of file scheduler.h.

Referenced by l4_sched_cpu_set(), and set().

Here is the caller graph for this function:

14.209.2.2 offset()

```
unsigned l4_sched_cpu_set_t::offset ( ) const  [inline]
```

Returns

Get offset value

Definition at line 69 of file scheduler.h.

Referenced by l4_sched_cpu_set(), and set().

Here is the caller graph for this function:

14.209.3 Field Documentation

14.209.3.1 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 58 of file scheduler.h.

Referenced by L4::Scheduler::info(), and l4_sched_cpu_set().

The documentation for this struct was generated from the following file:

- l4/sys/scheduler.h

14.210 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.

14.210.1 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 120 of file scheduler.h.

The documentation for this struct was generated from the following file:

- l4/sys/scheduler.h

14.211 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.

14.211.1 Detailed Description

Send-flex-page types.

Definition at line 98 of file __l4_fpage.h.

The documentation for this struct was generated from the following file:

- l4/sys/__l4_fpage.h

14.212 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)

14.212.1 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

14.213 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

14.213.1 Detailed Description

Basic timeout specification.

Basically a floating point number with 10 bits mantissa and 5 bits exponent (t = m∗2^e).

The timeout can also specify an absolute point in time (bit 16 == 1).

Definition at line 45 of file __timeout.h.

The documentation for this struct was generated from the following file:

- l4/sys/__timeout.h

14.214 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

14.214.1 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 57 of file __timeout.h.

The documentation for this union was generated from the following file:

- l4/sys/__timeout.h

14.215 l4_tracebuffer_status_t Struct Reference

Trace-buffer status.

```
#include <ktrace.h>
```

Collaboration diagram for l4_tracebuffer_status_t:

Data Fields

- volatile l4_tracebuffer_entry_t * current_entry

  Address of the most current event in trace-buffer.
- l4_uint32_t logevents [LOG_EVENT_MAX_EVENTS]

  Available LOG events.
- l4_uint32_t scaler_tsc_to_ns

  Scaler used for translation of CPU cycles to nano seconds.
- l4_uint32_t scaler_tsc_to_us

  Scaler used for translation of CPU cycles to micro seconds.
- l4_uint32_t scaler_ns_to_tsc

  Scaler used for translation of nano seconds to CPU cycles.

- volatile l4_uint32_t cnt_context_switch

  Number of context switches (intra AS or inter AS)
- volatile l4_uint32_t cnt_addr_space_switch

  Number of inter AS context switches.
- volatile l4_uint32_t cnt_shortcut_failed

  How often was the IPC shortcut taken.
- volatile l4_uint32_t cnt_shortcut_success

  How often was the IPC shortcut not taken.
- volatile l4_uint32_t cnt_irq

  Number of hardware interrupts (without kernel scheduling interrupt)
- volatile l4_uint32_t cnt_ipc_long

  Number of long IPCs.
- volatile l4_uint32_t cnt_page_fault

  Number of page faults.
- volatile l4_uint32_t cnt_io_fault

  Number of faults (application runs at IOPL 0 and tries to execute cli, sti, in, or out but does not have a sufficient right in the I/O bitmap)
- volatile l4_uint32_t cnt_task_create

  Number of tasks created.
- volatile l4_uint32_t cnt_schedule

  Number of reschedules.
- volatile l4_uint32_t cnt_iobmap_tlb_flush

  Number of flushes of the I/O bitmap.

14.215.1 Detailed Description

Trace-buffer status.

Definition at line 67 of file ktrace.h.

14.215.2 Field Documentation

14.215.2.1 cnt_iobmap_tlb_flush

```
volatile l4_uint32_t l4_tracebuffer_status_t::cnt_iobmap_tlb_flush
```

Number of flushes of the I/O bitmap.

Increases on context switches between two small address spaces if at least one of the spaces has an I/O bitmap allocated.

Definition at line 106 of file ktrace.h.

The documentation for this struct was generated from the following file:

- l4/sys/ktrace.h

14.216 l4_tracebuffer_status_window_t Struct Reference

Trace-buffer status window descriptor.

```
#include <ktrace.h>
```

Collaboration diagram for l4_tracebuffer_status_window_t:

Data Fields

- l4_tracebuffer_entry_t * tracebuffer

  Address of trace-buffer.
- l4_umword_t size

  Size of trace-buffer.
- volatile l4_uint64_t version

  Version number of trace-buffer (incremented if trace-buffer overruns)

14.216.1 Detailed Description

Trace-buffer status window descriptor.

Definition at line 52 of file ktrace.h.

The documentation for this struct was generated from the following file:

- l4/sys/ktrace.h

14.217 l4_vcon_attr_t Struct Reference

Vcon attribute structure.

```
#include <vcon.h>
```

Collaboration diagram for l4_vcon_attr_t:

Data Fields

- l4_umword_t i_flags

  input flags
- l4_umword_t o_flags

  output flags
- l4_umword_t l_flags

  local flags

14.217.1 Detailed Description

Vcon attribute structure.

Definition at line 178 of file vcon.h.

The documentation for this struct was generated from the following file:

- l4/sys/vcon.h

14.218 l4_vcpu_ipc_regs_t Struct Reference

vCPU message registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4_vcpu_ipc_regs_t:

14.218.1 Detailed Description

vCPU message registers.

Definition at line 63 of file __vcpu-arch.h.

The documentation for this struct was generated from the following file:

- arm/l4/sys/__vcpu-arch.h

14.219 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 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

14.219.1 Detailed Description

vCPU registers.

Definition at line 39 of file __vcpu-arch.h.

14.219.2 Field Documentation

14.219.2.1 ax

```
l4_umword_t l4_vcpu_regs_t::ax
```

rax register

eax register

Definition at line 68 of file __vcpu-arch.h.

14.219.2.2 bp

```
l4_umword_t l4_vcpu_regs_t::bp
```

rbp register

ebp register

Definition at line 63 of file __vcpu-arch.h.

14.219.2.3 bx

```
l4_umword_t l4_vcpu_regs_t::bx
```

rbx register

ebx register

Definition at line 65 of file __vcpu-arch.h.

14.219.2.4 cx

```
l4_umword_t l4_vcpu_regs_t::cx
```

rcx register

ecx register

Definition at line 67 of file __vcpu-arch.h.

14.219.2.5 di

```
l4_umword_t l4_vcpu_regs_t::di
```

rdi register

edi register

Definition at line 61 of file __vcpu-arch.h.

14.219.2.6 dx

```
l4_umword_t l4_vcpu_regs_t::dx
```

rdx register

edx register

Definition at line 66 of file __vcpu-arch.h.

14.219.2.7 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

14.220 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.

- l4_uint16_t saved_state

  Saved vCPU state.

- l4_uint16_t sticky_flags

  Pending 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.

14.220.1 Detailed Description

State of a vCPU.

Definition at line 34 of file vcpu.h.

The documentation for this struct was generated from the following file:

- l4/sys/vcpu.h

14.221 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.

14.221.1 Detailed Description

Virtual hardware devices description.

Examples:

examples/sys/ux-vhw/main.c.

Definition at line 70 of file vhw.h.

14.221.2 Field Documentation

14.221.2.1 count

```
l4_uint8_t l4_vhw_descriptor::count
```

Number of entries.

Examples:

examples/sys/ux-vhw/main.c.

Definition at line 73 of file vhw.h.

14.221.2.2 descs

```
struct l4_vhw_entry l4_vhw_descriptor::descs[]
```

Array of device descriptions.

Definition at line 77 of file vhw.h.

14.221.2.3 magic

```
l4_uint32_t l4_vhw_descriptor::magic
```

Magic.

Examples:

examples/sys/ux-vhw/main.c.

Definition at line 71 of file vhw.h.

14.221.2.4 version

```
l4_uint8_t l4_vhw_descriptor::version
```

Version of the descriptor.

Examples:

examples/sys/ux-vhw/main.c.

Definition at line 72 of file vhw.h.

The documentation for this struct was generated from the following file:

- l4/sys/vhw.h

14.222 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.

14.222.1 Detailed Description

Description of a device.

Examples:

examples/sys/ux-vhw/main.c.

Definition at line 55 of file vhw.h.

14.222.2 Field Documentation

14.222.2.1 fd

```
l4_uint32_t l4_vhw_entry::fd
```

File descriptor.

Definition at line 63 of file vhw.h.

14.222.2.2 irq_no

```
l4_uint32_t l4_vhw_entry::irq_no
```

IRQ number.

Examples:

examples/sys/ux-vhw/main.c.

Definition at line 62 of file vhw.h.

14.222.2.3 mem_size

```
l4_addr_t l4_vhw_entry::mem_size
```

Size of memory region.

Examples:

examples/sys/ux-vhw/main.c.

Definition at line 60 of file vhw.h.

14.222.2.4 mem_start

```
l4_addr_t l4_vhw_entry::mem_start
```

Start of memory region.

Examples:

examples/sys/ux-vhw/main.c.

Definition at line 59 of file vhw.h.

14.222.2.5 provider_pid

```
l4_uint32_t l4_vhw_entry::provider_pid
```

Host PID of the VHW provider.

Examples:

examples/sys/ux-vhw/main.c.

Definition at line 57 of file vhw.h.

14.222.2.6 type

```
enum l4_vhw_entry_type l4_vhw_entry::type
```

Type of virtual hardware.

Examples:

examples/sys/ux-vhw/main.c.

Definition at line 56 of file vhw.h.

The documentation for this struct was generated from the following file:

- l4/sys/vhw.h

14.223 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:

14.223.1 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

14.224 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:

14.224.1 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

14.225 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:

14.225.1 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

14.226 l4_vm_svm_vmcb_t Struct Reference

Control structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4_vm_svm_vmcb_t:

14.226.1 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

14.227 l4_vm_tz_state Struct Reference

state structure for TrustZone VMs

```
#include <vm.h>
```

Collaboration diagram for l4_vm_tz_state:

14.227.1 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

14.228 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 ()=0 throw ()

  Allocate a capability.
- template<typename T >
  L4::Cap< T > alloc () throw ()

  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)=0 throw ()

  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.

14.228.1 Detailed Description

Capability allocator interface.

Definition at line 41 of file cap_alloc.

14.228.2 Member Function Documentation

14.228.2.1 alloc() [1/2]

```
virtual L4::Cap<void> L4Re::Cap_alloc::alloc ( ) throw )   [pure virtual]
```

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:

14.228.2.2 alloc() [2/2]

```
template<typename T >
L4::Cap<T> L4Re::Cap_alloc::alloc ( ) throw )   [inline]
```

Allocate a capability.

Returns

Capability of type T

Definition at line 64 of file cap_alloc.

References alloc(), L4::cap_cast(), free(), L4_FP_ALL_SPACES, L4_INVALID_CAP, and ~Cap_alloc().

Here is the call graph for this function:

14.228.2.3 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 ) throw )   [pure virtual]
```

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 alloc(), and get_cap_alloc().

Here is the caller graph for this function:

14.228.2.4 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

14.229 L4Re::Console Class Reference

Console class.

Inheritance diagram for L4Re::Console:

Collaboration diagram for L4Re::Console:

Additional Inherited Members

14.229.1 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

14.230 L4Re::Dataspace Class Reference

Interface for memory-like objects.

Inheritance diagram for L4Re::Dataspace:

Collaboration diagram for L4Re::Dataspace:

Data Structures

- struct Stats

  Information about the dataspace.

Public Types

- enum Map_flags {
  Map_ro = 0, Map_rw = 1, Map_normal = 0x00, Map_cacheable = Map_normal,
  Map_bufferable = 0x10, Map_uncacheable = 0x20, Map_caching_mask = 0x30, Map_caching_shift = 4 }

  Flags for map operations.

Public Member Functions

- long map (l4_addr_t offset, unsigned long flags, l4_addr_t local_addr, l4_addr_t min_addr, l4_addr_t max_addr) const throw ()

  Request a flex-page mapping from the dataspace.
- long map_region (l4_addr_t offset, unsigned long flags, l4_addr_t min_addr, l4_addr_t max_addr) const throw ()

  Map a part of a dataspace completely.
- long clear (l4_addr_t offset, unsigned long size)

  Clear parts of a dataspace.
- long allocate (l4_addr_t offset, l4_size_t size)

  Allocate a range in the dataspace.
- long copy_in (l4_addr_t dst_offs, L4::Ipc::Cap< Dataspace > src, l4_addr_t src_offs, unsigned long size)

  Copy contents from another dataspace.
- long phys (l4_addr_t offset, l4_addr_t &phys_addr, l4_size_t &phys_size)

  Get the physical addresses of a dataspace.
- unsigned long size () const throw ()

  Get size of a dataspace.
- long flags () const throw ()

  Get flags of the dataspace.
- long info (Stats *stats)

  Get information on the dataspace.

Additional Inherited Members

14.230.1 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_ds/ds_srv.cc.

Definition at line 59 of file dataspace.

14.230.2 Member Enumeration Documentation

14.230.2.1 Map_flags

```
enum L4Re::Dataspace::Map_flags
```

Flags for map operations.

Enumerator

Map_ro	Request read-only mapping.
Map_rw	Request writable mapping.
Map_normal	request normal memory mapping
Map_cacheable	request normal memory mapping
Map_bufferable	request bufferable (write buffered) mappings
Map_uncacheable	request uncacheable memory mappings
Map_caching_mask	mask for caching flags
Map_caching_shift	shift value for caching flags

Definition at line 68 of file dataspace.

14.230.3 Member Function Documentation

14.230.3.1 allocate()

```
long L4Re::Dataspace::allocate (
            l4_addr_t offset,
            l4_size_t 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.

14.230.3.2 clear()

```
long L4Re::Dataspace::clear (
            l4_addr_t offset,
            unsigned long 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.

14.230.3.3 copy_in()

```
long L4Re::Dataspace::copy_in (
            l4_addr_t dst_offs,
            L4::Ipc::Cap< Dataspace > src,
            l4_addr_t src_offs,
            unsigned long 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.

14.230.3.4 flags()

```
long L4Re::Dataspace::flags ( ) const throw )
```

Get flags of the dataspace.

Return values

>=0	Flags of the dataspace
<0	IPC errors

See also

L4Re::Dataspace::Map_flags

Definition at line 116 of file dataspace_impl.h.

14.230.3.5 info()

```
long L4Re::Dataspace::info (
            Stats * stats )
```

Get information on the dataspace.

Parameters

out	*stats*	Dataspace information

Return values

0	Success
<0	IPC errors

14.230.3.6 map()

```
long L4Re::Dataspace::map (
            l4_addr_t offset,
            unsigned long flags,
            l4_addr_t local_addr,
            l4_addr_t min_addr,
            l4_addr_t max_addr ) const throw )
```

Request a flex-page mapping from the dataspace.

Parameters

offset	Offset to start within dataspace
flags	map flags, see Map_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 92 of file dataspace_impl.h.

References l4_fpage_max_order(), L4_LOG2_PAGESIZE, l4_round_page(), and l4_trunc_page().

Here is the call graph for this function:

14.230.3.7 map_region()

```
long L4Re::Dataspace::map_region (
            l4_addr_t offset,
            unsigned long flags,
            l4_addr_t min_addr,
            l4_addr_t max_addr ) const throw )
```

Map a part of a dataspace completely.

Parameters

offset	Offset to start within dataspace
flags	map flags, see Map_flags.
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

Definition at line 55 of file dataspace_impl.h.

References l4_fpage_max_order(), L4_LOG2_PAGESIZE, l4_round_page(), l4_round_size(), l4_trunc_page(), l4_trunc_size(), and L4_UNLIKELY.

Referenced by L4Re::Rm::attach().

Here is the call graph for this function:

Here is the caller graph for this function:

14.230.3.8 phys()

```
long L4Re::Dataspace::phys (
            l4_addr_t offset,
            l4_addr_t & phys_addr,
            l4_size_t & phys_size )
```

Get the physical addresses of a dataspace.

Parameters

offset	Offset in dataspace

Return values

phys_addr	Physical address.
phys_size	Size of largest physically contiguous region in the dataspace after `phys_addr` (in bytes).

Return values

L4_EOK	Success
-L4_EINVAL	Dataspace is not pinned.
<0	IPC errors

This call will only succeed on pinned memory dataspaces.

Deprecated Use L4Re::Dma_space instead. This function will be removed in future releases.

14.230.3.9 size()

```
unsigned long L4Re::Dataspace::size ( ) const throw )
```

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 106 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

14.231 L4Re::Dataspace::Stats Struct Reference

Information about the dataspace.

Collaboration diagram for L4Re::Dataspace::Stats:

Data Fields

- unsigned long size

  size
- unsigned long flags

  flags

14.231.1 Detailed Description

Information about the dataspace.

Definition at line 85 of file dataspace.

The documentation for this struct was generated from the following file:

- l4/re/dataspace

14.232 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

14.232.1 Detailed Description

Debug interface.

See also

Debugging API .

Definition at line 51 of file debug.

14.232.2 Member Function Documentation

14.232.2.1 debug()

```
long L4Re::Debug_obj::debug (
            unsigned long function )
```

Debug call.

Parameters

function	Function to call.

Returns

- L4_EOK
  - IPC errors

The documentation for this class was generated from the following file:

- l4/re/debug

14.233 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, l4_addr_t 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.

14.233.1 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 61 of file dma_space.

14.233.2 Member Typedef Documentation

14.233.2.1 Attributes

```
typedef L4::Types::Flags<Attribute> L4Re::Dma_space::Attributes
```

Attributes for DMA mappings.

See also

Attribute

Definition at line 106 of file dma_space.

14.233.3 Member Enumeration Documentation

14.233.3.1 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 85 of file dma_space.

14.233.3.2 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 73 of file dma_space.

14.233.3.3 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 113 of file dma_space.

14.233.4 Member Function Documentation

14.233.4.1 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 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 Space_attrib.

14.233.4.2 disassociate()

```
long L4Re::Dma_space::disassociate ( )
```

Disassociate the DMA task from this Dma_space.

Precondition

requires capability rights: {RW}

14.233.4.3 map()

```
long L4Re::Dma_space::map (
            L4::Ipc::Cap< L4Re::Dataspace > src,
            l4_addr_t 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 of the region to be mapped to for DMA, after successful mapping the size returned is the size mapped for DMA as 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.

14.233.4.4 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

14.234 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 throw ()

  Object-capability to the parent.
- L4::Cap< Mem_alloc > mem_alloc () const throw ()

  Object-capability to the memory allocator.
- L4::Cap< L4::Factory > user_factory () const throw ()

  Object-capability to the user-level object factory.
- L4::Cap< Rm > rm () const throw ()

  Object-capability to the region map.
- L4::Cap< Log > log () const throw ()

  Object-capability to the logging service.
- L4::Cap< L4::Thread > main_thread () const throw ()

  Object-capability of the first user thread.
- L4::Cap< L4::Task > task () const throw ()

  Object-capability of the user task.
- L4::Cap< L4::Factory > factory () const throw ()

  Object-capability to the factory object available to the task.
- l4_cap_idx_t first_free_cap () const throw ()

  First available capability selector.
- l4_fpage_t utcb_area () const throw ()

  UTCB area of the task.
- l4_addr_t first_free_utcb () const throw ()

  First free UTCB.
- Cap_entry const * initial_caps () const throw ()

  Get a pointer to the first entry in the initial objects array.
- Cap_entry const * get (char const *name, unsigned l) const throw ()

  Get the Cap_entry for the object named name.
- template<typename T >
  L4::Cap< T > get_cap (char const *name, unsigned l) const throw ()

  Get the capability selector for the object named name.
- template<typename T >
  L4::Cap< T > get_cap (char const *name) const throw ()

  Get the capability selector for the object named name.
- void parent (L4::Cap< Parent > const &c) throw ()

  Set parent object-capability.
- void mem_alloc (L4::Cap< Mem_alloc > const &c) throw ()

  Set memory allocator object-capability.
- void rm (L4::Cap< Rm > const &c) throw ()

  Set region map object-capability.
- void log (L4::Cap< Log > const &c) throw ()

  Set log object-capability.
- void main_thread (L4::Cap< L4::Thread > const &c) throw ()

  Set object-capability of first user thread.
- void factory (L4::Cap< L4::Factory > const &c) throw ()

Set factory object-capability.

- void first_free_cap (l4_cap_idx_t c) throw ()

  Set first available capability selector.
- void utcb_area (l4_fpage_t utcbs) throw ()

  Set UTCB area of the task.
- void first_free_utcb (l4_addr_t u) throw ()

  Set first free UTCB.
- L4::Cap< L4::Scheduler > scheduler () const throw ()

  Get the scheduler capability for the task.
- void scheduler (L4::Cap< L4::Scheduler > const &c) throw ()

  Set the scheduler capability.
- void initial_caps (Cap_entry *first) throw ()

  Set the pointer to the first Cap_entry in the initial objects array.

Static Public Member Functions

- static Env const * env () throw ()

  Returns the initial environment for the current task.

14.234.1 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 the C interface refer to Initial Environment.

Definition at line 82 of file env.

14.234.2 Member Function Documentation

14.234.2.1 env()

```
static Env const* L4Re::Env::env ( ) throw )   [inline], [static]
```

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.cc.

Definition at line 100 of file env.

Referenced by L4virtio::Svr::Dev_config::Dev_config(), and L4vcpu::Vcpu::entry_ip().

Here is the caller graph for this function:

14.234.2.2 factory() [1/2]

```
L4::Cap<L4::Factory> L4Re::Env::factory ( ) const throw )   [inline]
```

Object-capability to the factory object available to the task.

Returns

Factory object-capability

Definition at line 148 of file env.

References l4re_env_t::factory.

14.234.2.3 factory() [2/2]

```
void L4Re::Env::factory (
            L4::Cap< L4::Factory > const & c ) throw )   [inline]
```

Set factory object-capability.

Parameters

c	Factory object-capability

Definition at line 253 of file env.

References l4re_env_t::factory.

14.234.2.4 first_free_cap() [1/2]

```
l4_cap_idx_t L4Re::Env::first_free_cap ( ) const throw )   [inline]
```

First available capability selector.

Returns

First capability selector.

First capability selector available for use for in the application.

Definition at line 156 of file env.

References l4re_env_t::first_free_cap.

14.234.2.5 first_free_cap() [2/2]

```
void L4Re::Env::first_free_cap (
            l4_cap_idx_t c ) throw )   [inline]
```

Set first available capability selector.

Parameters

c	First capability selector available to the application.

Definition at line 259 of file env.

References l4re_env_t::first_free_cap.

14.234.2.6 first_free_utcb() [1/2]

```
l4_addr_t L4Re::Env::first_free_utcb ( ) const throw )   [inline]
```

First free UTCB.

Returns

object-capability

First free UTCB within the UTCB area available for the application to use.

Definition at line 171 of file env.

References l4re_env_t::first_free_utcb.

14.234.2.7 first_free_utcb() [2/2]

```
void L4Re::Env::first_free_utcb (
            l4_addr_t u ) throw )   [inline]
```

Set first free UTCB.

Parameters

u	First UTCB available for the application to use.

Definition at line 271 of file env.

References l4re_env_t::first_free_utcb.

14.234.2.8 get()

```
Cap_entry const* L4Re::Env::get (
            char const * name,
            unsigned l ) const throw )   [inline]
```

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 189 of file env.

References l4re_env_get_cap_l().

Here is the call graph for this function:

14.234.2.9 get_cap() [1/2]

```
template<typename T >
L4::Cap<T> L4Re::Env::get_cap (
            char const * name,
            unsigned l ) const throw )   [inline]
```

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 201 of file env.

References L4_ENOENT.

14.234.2.10 get_cap() [2/2]

```
template<typename T >
L4::Cap<T> L4Re::Env::get_cap (
            char const * name ) const throw )   [inline]
```

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 216 of file env.

14.234.2.11 initial_caps() [1/2]

```
Cap_entry const* L4Re::Env::initial_caps ( ) const throw )   [inline]
```

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 178 of file env.

14.234.2.12 initial_caps() [2/2]

```
void L4Re::Env::initial_caps (
            Cap_entry * first ) throw )   [inline]
```

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 293 of file env.

14.234.2.13 log() [1/2]

```
L4::Cap<Log> L4Re::Env::log ( ) const throw )   [inline]
```

Object-capability to the logging service.

Returns

Log object-capability

Definition at line 130 of file env.

References l4re_env_t::log.

14.234.2.14 log() [2/2]

```
void L4Re::Env::log (
            L4::Cap< Log > const & c ) throw )   [inline]
```

Set log object-capability.

Parameters

c	Log object-capability

Definition at line 241 of file env.

References l4re_env_t::log.

14.234.2.15 main_thread() [1/2]

```
L4::Cap<L4::Thread> L4Re::Env::main_thread ( ) const throw )   [inline]
```

Object-capability of the first user thread.

Returns

Object-capability of the first user thread.

Definition at line 136 of file env.

References l4re_env_t::main_thread.

14.234.2.16 main_thread() [2/2]

```
void L4Re::Env::main_thread (
            L4::Cap< L4::Thread > const & c ) throw )   [inline]
```

Set object-capability of first user thread.

Parameters

c	First thread's object-capability

Definition at line 247 of file env.

References l4re_env_t::main_thread.

14.234.2.17 mem_alloc() [1/2]

```
L4::Cap<Mem_alloc> L4Re::Env::mem_alloc ( ) const throw )   [inline]
```

Object-capability to the memory allocator.

Returns

Memory allocator object-capability

Examples:

examples/libs/l4re/c++/shared_ds/ds_srv.cc.

Definition at line 113 of file env.

References l4re_env_t::mem_alloc.

14.234.2.18 mem_alloc() [2/2]

```
void L4Re::Env::mem_alloc (
            L4::Cap< Mem_alloc > const & c ) throw )   [inline]
```

Set memory allocator object-capability.

Parameters

c	Memory allocator object-capability

Definition at line 229 of file env.

References l4re_env_t::mem_alloc.

14.234.2.19 parent() [1/2]

```
L4::Cap<Parent> L4Re::Env::parent ( ) const throw )   [inline]
```

Object-capability to the parent.

Returns

Parent object-capability

Definition at line 107 of file env.

References l4re_env_t::parent.

14.234.2.20 parent() [2/2]

```
void L4Re::Env::parent (
            L4::Cap< Parent > const & c ) throw )   [inline]
```

Set parent object-capability.

Parameters

c	Parent object-capability

Definition at line 223 of file env.

References l4re_env_t::parent.

14.234.2.21 rm() [1/2]

```
L4::Cap<Rm> L4Re::Env::rm ( ) const throw )   [inline]
```

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 124 of file env.

References l4re_env_t::rm.

14.234.2.22 rm() [2/2]

```
void L4Re::Env::rm (
            L4::Cap< Rm > const & c ) throw )   [inline]
```

Set region map object-capability.

Parameters

c	Region map object-capability

Definition at line 235 of file env.

References l4re_env_t::rm.

14.234.2.23 scheduler() [1/2]

```
L4::Cap<L4::Scheduler> L4Re::Env::scheduler ( ) const throw )   [inline]
```

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 279 of file env.

References l4re_env_t::scheduler.

14.234.2.24 scheduler() [2/2]

```
void L4Re::Env::scheduler (
            L4::Cap< L4::Scheduler > const & c ) throw )   [inline]
```

Set the scheduler capability.

Parameters

c	is the capability to be set as scheduler.

Definition at line 286 of file env.

References l4re_env_t::scheduler.

14.234.2.25 task()

```
L4::Cap<L4::Task> L4Re::Env::task ( ) const throw )   [inline]
```

Object-capability of the user task.

Returns

Object-capability of the user task.

Definition at line 142 of file env.

14.234.2.26 utcb_area() [1/2]

```
l4_fpage_t L4Re::Env::utcb_area ( ) const throw )   [inline]
```

UTCB area of the task.

Returns

UTCB area

Definition at line 162 of file env.

References l4re_env_t::utcb_area.

14.234.2.27 utcb_area() [2/2]

```
void L4Re::Env::utcb_area (
            l4_fpage_t utcbs ) throw )   [inline]
```

Set UTCB area of the task.

Parameters

utcbs	UTCB area

Definition at line 265 of file env.

References l4re_env_t::utcb_area.

The documentation for this class was generated from the following file:

- l4/re/env

14.235 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

14.235.1 Detailed Description

Event class.

Definition at line 133 of file event.

14.235.2 Member Function Documentation

14.235.2.1 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.

Referenced by L4Re::Util::Event_t< PAYLOAD >::init(), and L4Re::Util::Event_t< PAYLOAD >::init_poll().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/re/event

14.236 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 () throw ()

  Next event in buffer.
- bool put (Event const &ev) throw ()

  Put event into buffer at current position.

14.236.1 Detailed Description

template<typename PAYLOAD = Default_event_payload>
class L4Re::Event_buffer_t< PAYLOAD >

Event buffer class.

Definition at line 187 of file event.

14.236.2 Constructor & Destructor Documentation

14.236.2.1 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 234 of file event.

14.236.3 Member Function Documentation

14.236.3.1 next()

```
template<typename PAYLOAD = Default_event_payload>
Event* L4Re::Event_buffer_t< PAYLOAD >::next ( ) throw )   [inline]
```

Next event in buffer.

Returns

0 if no event available, event otherwise.

Definition at line 244 of file event.

References L4Re::Event_buffer_t< PAYLOAD >::Event::time.

14.236.3.2 put()

```
template<typename PAYLOAD = Default_event_payload>
bool L4Re::Event_buffer_t< PAYLOAD >::put (
            Event const & ev ) throw )   [inline]
```

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 261 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

14.237 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 () throw ()

  Free the entry.

Data Fields

- long long time

  Event time stamp.

14.237.1 Detailed Description

template<typename PAYLOAD = Default_event_payload>
struct L4Re::Event_buffer_t< PAYLOAD >::Event

Event structure used in buffer.

Definition at line 194 of file event.

The documentation for this struct was generated from the following file:

- l4/re/event

14.238 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

14.238.1 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.

14.238.2 Member Enumeration Documentation

14.238.2.1 anonymous enum

```
anonymous enum
```

Enumerator

Name_max	The maximum length of a lock's name.

Definition at line 44 of file inhibitor.

14.238.3 Member Function Documentation

14.238.3.1 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.

14.238.3.2 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(), and L4_INLINE_RPC_NF.

Here is the call graph for this function:

14.238.3.3 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

14.239 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 throw ()

  Print string with length len, NULL characters don't matter.
- void print (char const ∗string) const throw ()

  Print NULL-terminated string.

Additional Inherited Members

14.239.1 Detailed Description

Log interface class.

Definition at line 44 of file log.

14.239.2 Member Function Documentation

14.239.2.1 print()

```
void L4Re::Log::print (
            char const * string ) const throw )
```

Print NULL-terminated string.

Parameters

string	string to print

14.239.2.2 printn()

```
void L4Re::Log::printn (
            char const * string,
            int len ) const throw )
```

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

14.240 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 throw ()

  Allocate anonymous memory.
- long free (L4::Cap< Dataspace > mem) const throw ()

  Free dataspace.

Additional Inherited Members

14.240.1 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.

14.240.2 Member Enumeration Documentation

14.240.2.1 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.

14.240.3 Member Function Documentation

14.240.3.1 alloc()

```
long L4Re::Mem_alloc::alloc (
            long size,
            L4::Cap< Dataspace > mem,
            unsigned long flags = 0,
            unsigned long align = 0 ) const throw )
```

Allocate anonymous memory.

Parameters

	size	Size in bytes to be requested (granularity is (super)pages and the size is rounded up to this granularity). 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:

14.240.3.2 free()

```
long L4Re::Mem_alloc::free (
            L4::Cap< Dataspace > mem ) const throw )
```

Free dataspace.

Parameters

mem	Dataspace to be released.

Return values

0	

Deprecated This function is an empty stub which remains here for backward compatibility only. Use `L4::Task::unmap(mem, L4_FP_DELETE_OBJ)` or other similar means to remove a dataspace.

Definition at line 47 of file mem_alloc_impl.h.

The documentation for this class was generated from the following files:

- l4/re/mem_alloc
- l4/re/impl/mem_alloc_impl.h

14.241 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

14.241.1 Detailed Description

Interface for memory-like address space accessible via IPC.

Include File

```
#include <l4/re/mmio_space>
```

Definition at line 25 of file mmio_space.

14.241.2 Member Enumeration Documentation

14.241.2.1 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 29 of file mmio_space.

14.241.3 Member Function Documentation

14.241.3.1 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.

14.241.3.2 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

14.242 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_FP↩AGE_RWS,
  Strong = L4_CAP_FPAGE_S, Trusted = 0x008 , 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 throw ()

  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 throw ()

Query the name space for a named object.

- long register_obj (char const *name, L4::Ipc::Cap< void > obj, unsigned flags=Rw) const throw ()

  Register an object with a name.

- long unlink (char const *name)

  Remove an entry from the name space.

Additional Inherited Members

14.242.1 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.

14.242.2 Member Enumeration Documentation

14.242.2.1 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.

14.242.2.2 Register_flags

`enum L4Re::Namespace::Register_flags`

Flags for registering name spaces.

Enumerator

Ro	Read-only.

Enumerator

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.

14.242.3 Member Function Documentation

14.242.3.1 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 throw )
```

Query the name space for a named object.

Parameters

in	*name*	String to query (without any leading slashes).
in	*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 118 of file namespace_impl.h.

14.242.3.2 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 throw )
```

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).
in	*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:

14.242.3.3 register_obj()

```
long L4Re::Namespace::register_obj (
            char const * name,
            L4::Ipc::Cap< void > obj,
            unsigned flags = Rw ) const throw )   [inline]
```

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.

References L4_RPC_NF_OP.

14.242.3.4 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

14.243 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.

14.243.1 Detailed Description

Direct control interface for Ned.

Definition at line 20 of file cmd_control.

14.243.2 Member Function Documentation

14.243.2.1 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.

14.243.2.2 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

14.244 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

14.244.1 Detailed Description

Parent interface.

See also

Parent API for more details about the purpose.

Definition at line 51 of file parent.

14.244.2 Member Function Documentation

14.244.2.1 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

14.245 L4Re::Rm Class Reference

Region map.

```
#include <l4/re/rm>
```

Inheritance diagram for L4Re::Rm:

Collaboration diagram for L4Re::Rm:

Public Types

- enum Detach_result { Detached_ds = 0, Kept_ds = 1, Split_ds = 2 , Detach_again = 4 }

  Result values for detach operation.
- enum Region_flags {
  Read_only = 0x01, Detach_free = 0x02, Pager = 0x04, Reserved = 0x08,
  Caching_shift = 8, Caching_ds_shift = Caching_shift - Dataspace::Map_caching_shift, Caching =

Dataspace::Map_caching_mask << Caching_ds_shift, Cache_normal = Dataspace::Map_normal << Caching_ds_shift,
Cache_buffered = Dataspace::Map_bufferable << Caching_ds_shift, Cache_uncached = Dataspace:↩ :Map_uncacheable << Caching_ds_shift, Region_flags = Caching | 0x0f }

Flags for regions.

- enum Attach_flags { Search_addr = 0x20, In_area = 0x40, Eager_map = 0x80, Attach_flags = 0xf0 }

  Flags for attach operation.
- enum Detach_flags { Detach_exact = 1, Detach_overlap = 2, Detach_keep = 4 }

  Flags for detach operation.

Public Member Functions

- long reserve_area (l4_addr_t *start, unsigned long size, unsigned flags=0, unsigned char align=L4_PAGE↩ SHIFT) const throw ()

  Reserve the given area in the region map.
- template<typename T >
  long reserve_area (T **start, unsigned long size, unsigned flags=0, unsigned char align=L4_PAGESHIFT) const throw ()

  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, unsigned long flags, L4::Ipc::Cap< Dataspace > mem, l4_addr_t offs=0, unsigned char align=L4_PAGESHIFT) const throw ()

  Attach a data space to a region.
- template<typename T >
  long attach (T **start, unsigned long size, unsigned long flags, L4::Ipc::Cap< Dataspace > mem, l4_addr_t offs=0, unsigned char align=L4_PAGESHIFT) const throw ()

  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 throw ()

  Detach a region from the address space.
- int detach (void *addr, L4::Cap< Dataspace > *mem, L4::Cap< L4::Task > const &task=This_task) const throw ()

  Detach 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 throw ()

  Detach all regions of the specified interval.
- int find (l4_addr_t *addr, unsigned long *size, l4_addr_t *offset, unsigned *flags, L4::Cap< Dataspace > *m) throw ()

  Find a region given an address and size.

Additional Inherited Members

14.245.1 Detailed Description

Region map.

Definition at line 69 of file rm.

14.245.2 Member Enumeration Documentation

14.245.2.1 Attach_flags

`enum L4Re::Rm::Attach_flags`

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_flags	Mask of all attach flags.

Definition at line 111 of file rm.

14.245.2.2 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 Detach_free.

Definition at line 121 of file rm.

14.245.2.3 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 75 of file rm.

14.245.2.4 Region_flags

```
enum L4Re::Rm::Region_flags
```

Flags for regions.

Enumerator

Read_only	Region is read-only.
Detach_free	Free the portion of the data space after detach.
Pager	Region has a pager.
Reserved	Region is reserved (blocked)
Caching_shift	Start of Rm cache bits.
Caching_ds_shift	Shift value for Dataspace to Rm cache bits.
Caching	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.
Region_flags	Mask of all region flags.

Definition at line 86 of file rm.

14.245.3 Member Function Documentation

14.245.3.1 attach() [1/2]

```
long L4Re::Rm::attach (
            l4_addr_t * start,
            unsigned long size,
            unsigned long flags,
            L4::Ipc::Cap< Dataspace > mem,
            l4_addr_t offs = 0,
            unsigned char align = L4_PAGESHIFT ) const throw )
```

Attach a data space to a region.

Parameters

in,out	*start*	Virtual start address where the region manager shall attach the data space. If L4Re::Rm::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::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.

Parameters

	size	Size of the data space to attach (in bytes)
	flags	Flags, see L4Re::Rm::Attach_flags and L4Re::Rm::Region_flags. If the `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::Search_addr flag is used.

Return values

0	Success
-L4_ENOENT	No area could be found (see L4Re::Rm::In_area)
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAV↩AIL	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.

References L4::Ipc::Cap< T >::cap(), detach(), L4::Cap_base::is_valid(), L4_FP_ALL_SPACES, l4_fpage(), L4↩_FPAGE_RWX, L4_INVALID_CAP, L4_LOG2_PAGESIZE, l4_round_page(), L4_UNLIKELY, L4Re::Dataspace↩::map_region(), L4Re::Dataspace::Map_ro, L4Re::Dataspace::Map_rw, and L4::Task::unmap().

Here is the call graph for this function:

14.245.3.2 attach() [2/2]

```
template<typename T >
long L4Re::Rm::attach (
            T ** start,
            unsigned long size,
            unsigned long flags,
            L4::Ipc::Cap< Dataspace > mem,
            l4_addr_t offs = 0,
            unsigned char align = L4_PAGESHIFT ) const throw )   [inline]
```

Attach a data space to a region.

Parameters

in,out	*start*	Virtual start address where the region manager shall attach the data space. If L4Re::Rm::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::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)
	flags	Flags, see L4Re::Rm::Attach_flags and L4Re::Rm::Region_flags. If the `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::Search_addr flag is used.

Return values

0	Success
-L4_ENOENT	No area could be found (see L4Re::Rm::In_area)
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAV↩AIL	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 345 of file rm.

References L4::Cap_base::is_valid(), and L4_PAGESHIFT.

Here is the call graph for this function:

14.245.3.3 detach() [1/3]

```
int L4Re::Rm::detach (
            l4_addr_t addr,
            L4::Cap< Dataspace > * mem,
            L4::Cap< L4::Task > const & task = This_task ) const throw )   [inline]
```

Detach 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

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 589 of file rm.

Referenced by attach().

Here is the caller graph for this function:

14.245.3.4 detach() [2/3]

```
int L4Re::Rm::detach (
            void * addr,
            L4::Cap< Dataspace > * mem,
            L4::Cap< L4::Task > const & task = This_task ) const throw )   [inline]
```

Detach 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

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 594 of file rm.

14.245.3.5 detach() [3/3]

```
int L4Re::Rm::detach (
            l4_addr_t start,
            unsigned long size,
            L4::Cap< Dataspace > * mem,
            L4::Cap< L4::Task > const & task ) const throw )   [inline]
```

Detach all regions of 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

Detach_result	On success.
-L4_ENOENT	No region found.
<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 599 of file rm.

14.245.3.6 find()

```
int L4Re::Rm::find (
            l4_addr_t * addr,
            unsigned long * size,
            l4_addr_t * offset,
            unsigned * flags,
            L4::Cap< Dataspace > * m ) throw )   [inline]
```

Find a region given an address and size.

Parameters

	addr	Address to look for
	size	Size of the area to look for (in bytes).
out	*addr*	Start address of the found region.
out	*size*	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 Region_flags (and 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 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 553 of file rm.

References L4_RPC, and L4_RPC_NF.

14.245.3.7 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.

14.245.3.8 reserve_area() [1/2]

```
long L4Re::Rm::reserve_area (
            l4_addr_t * start,
```

```
            unsigned long size,
            unsigned flags = 0,
            unsigned char align = L4_PAGESHIFT ) const throw )   [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::Region_flags and L4Re::Rm::Attach_flags).
	align	Alignment of area if searched as bits (log2 value).

Return values

0	Success
-L4_EADDRNOTAV↩AIL	The given area cannot be reserved.
<0	IPC errors

This function reserves an area within the virtual address space implemented by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = Reserved), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the 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* = Search_addr), the space between *start* and the end of the virtual address space is searched.

Definition at line 239 of file rm.

References L4_RPC_NF.

14.245.3.9 reserve_area() [2/2]

```
template<typename T >
long L4Re::Rm::reserve_area (
            T ** start,
            unsigned long size,
            unsigned flags = 0,
            unsigned char align = L4_PAGESHIFT ) const throw )   [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 Region_flags and Attach_flags).
	align	Alignment of area if searched as bits (log2 value).

Return values

0	Success
-L4_EADDRNOTAV↩ AIL	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 265 of file rm.

References L4_PAGESHIFT, L4_RPC, and L4_RPC_NF.

The documentation for this class was generated from the following files:

- l4/re/rm
- l4/re/impl/rm_impl.h

14.246 L4Re::Smart_cap_auto< Unmap_flags > Class Template Reference

Helper for Auto_cap and Auto_del_cap.

Collaboration diagram for L4Re::Smart_cap_auto< Unmap_flags >:

14.246.1 Detailed Description

template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Smart_cap_auto< Unmap_flags >

Helper for Auto_cap and Auto_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

14.247 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) throw ()

  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) throw ()

  Invalidate operation for L4::Smart_cap.

14.247.1 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

14.248 L4Re::Util::Auto_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap of the capability selector.

Collaboration diagram for L4Re::Util::Auto_cap< T >:

14.248.1 Detailed Description

template<typename T>
struct L4Re::Util::Auto_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.

Deprecated Use L4Re::Util::Unique_cap.

This kind of automatic capability is useful for capabilities that shall have a lifetime that is strictly coupled to one C++ scope.

Usage:

```
{
  L4Re::Util::Auto_cap<L4Re::Dataspace>::Cap
    ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());

  // use the dataspace cap
  L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

  ...

  // At the end of the scope ds_cap is unmapped and the capability selector
  // is freed.
}
```

Definition at line 163 of file cap_alloc.

The documentation for this struct was generated from the following file:

- l4/re/util/cap_alloc

14.249 L4Re::Util::Auto_del_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector.

Collaboration diagram for L4Re::Util::Auto_del_cap< T >:

14.249.1 Detailed Description

template<typename T>
struct L4Re::Util::Auto_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.

Deprecated Use L4Re::Util::Unique_cap.

This kind of automatic capability is useful for capabilities with that shall have a lifetime that is strictly coupled to one C++ scope. The main difference to Auto_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::Auto_del_cap<L4Re::Dataspace>::Cap
    ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());

  // use the dataspace cap
  L4Re::chksys(mem_alloc->alloc(4096, 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 be
  // also deleted (even if there are other references to this data space).
}
```

Definition at line 200 of file cap_alloc.

The documentation for this struct was generated from the following file:

- l4/re/util/cap_alloc

14.250 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

14.250.1 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.

14.250.2 Member Function Documentation

14.250.2.1 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 50 of file br_manager.

14.250.2.2 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 81 of file br_manager.

References L4_CAP_MASK.

14.250.2.3 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 89 of file br_manager.

14.250.2.4 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 113 of file br_manager.

References l4_assert.

The documentation for this class was generated from the following file:

- l4/re/util/br_manager

14.251 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

14.251.1 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 160 of file br_manager.

The documentation for this struct was generated from the following file:

- l4/re/util/br_manager

14.252 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

14.252.1 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 174 of file br_manager.

The documentation for this struct was generated from the following file:

- l4/re/util/br_manager

14.253 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 () throw ()

  Allocate a capability slot.
- template<typename T >
  void free (L4::Cap< T > const &cap, l4_cap_idx_t task=-1UL, l4_umword_t unmap_flags=L4_FP_ALL_S↩PACES) throw ()

  Free a capability slot.

14.253.1 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

14.254 L4Re::Util::Counter< COUNTER > Struct Template Reference

Counter for Counting_cap_alloc with variable data width.

Collaboration diagram for L4Re::Util::Counter< COUNTER >:

14.254.1 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

14.255 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 () throw ()

  Allocate a new capability slot.
- template<typename T >
  L4::Cap< T > alloc () throw ()

  Allocate a new capability slot.
- void take (L4::Cap< void > cap) throw ()

  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_AL↩L_SPACES) throw ()

  Free the capability.
- bool release (L4::Cap< void > cap, l4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_flags=L4_FP↩_ALL_SPACES) throw ()

  Decrease the reference counter for a capability.
- long last () throw ()

  Return highest capability id managed by this allocator.

Protected Member Functions

- Counting_cap_alloc () throw ()

  Create a new, empty allocator.
- void setup (void *m, long capacity, long bias) throw ()

  Set up the backing memory for the allocator and the area of managed capability slots.

14.255.1 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.

14.255.2 Constructor & Destructor Documentation

14.255.2.1 Counting_cap_alloc()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::Counting_cap_alloc ( ) throw )   [inline],
[protected]
```

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.

14.255.3 Member Function Documentation

14.255.3.1 alloc() [1/2]

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
L4::Cap<void> L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc ( ) throw )   [inline]
```

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_ds/ds_srv.cc.

Definition at line 135 of file counting_cap_alloc.

References L4::Cap_base::Invalid, and L4_CAP_SHIFT.

Referenced by L4Re::Util::make_shared_cap(), L4Re::Util::make_shared_del_cap(), L4Re::Util::make_unique_cap(), and L4Re::Util::make_unique_del_cap().

Here is the caller graph for this function:

14.255.3.2 alloc() [2/2]

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
template<typename T >
L4::Cap<T> L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc ( ) throw )   [inline]
```

Allocate a new capability slot.

Returns

The newly allocated capability slot, invalid if the allocator was exhausted.

Definition at line 156 of file counting_cap_alloc.

References L4::cap_cast().

Here is the call graph for this function:

14.255.3.3 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 ) throw )   [inline]
```

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/ds_srv.cc, and examples/libs/l4re/streammap/client.cc.

Definition at line 198 of file counting_cap_alloc.

References l4_assert, L4_CAP_SHIFT, L4_INVALID_CAP, and l4_task_unmap().

Referenced by L4Re::Util::Object_registry::unregister_obj().

Here is the call graph for this function:

Here is the caller graph for this function:

14.255.3.4 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 ) throw )   [inline]
```

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 241 of file counting_cap_alloc.

References l4_assert, L4_CAP_SHIFT, L4_INVALID_CAP, and l4_task_unmap().

Here is the call graph for this function:

14.255.3.5 setup()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::setup (
            void * m,
            long capacity,
            long bias ) throw )   [inline], [protected]
```

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.

14.255.3.6 take()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::take (
            L4::Cap< void > cap ) throw )   [inline]
```

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 171 of file counting_cap_alloc.

References L4_CAP_SHIFT.

The documentation for this class was generated from the following file:

- l4/re/util/counting_cap_alloc

14.256 L4Re::Util::Dataspace_svr Class Reference

Dataspace server class.

Collaboration diagram for L4Re::Util::Dataspace_svr:

Public Member Functions

- int map (l4_addr_t offset, l4_addr_t local_addr, unsigned long flags, l4_addr_t min_addr, l4_addr_t max_addr, L4::Ipc::Snd_fpage &memory)

  Map a region of the dataspace.
- virtual int map_hook (l4_addr_t offs, unsigned long flags, l4_addr_t min, l4_addr_t max)

  A hook that is called as the first operation in each map request.
- virtual int phys (l4_addr_t offset, l4_addr_t &phys_addr, l4_size_t &phys_size) throw ()

  Return physical address for a virtual address.
- virtual void take () throw ()

  Take a reference to this dataspace.
- virtual unsigned long release () throw ()

  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) throw ()

  Copy from src dataspace to this destination dataspace.
- virtual long clear (unsigned long offs, unsigned long size) const throw ()

  Clear a region in the dataspace.
- virtual long allocate (l4_addr_t offset, l4_size_t size, unsigned access) throw ()

  Allocate a region within a dataspace.

- virtual unsigned long page_shift () const throw ()

  Define the size of the flexpage to map.
- virtual bool is_static () const throw ()

  Return whether the dataspace is static.

14.256.1 Detailed Description

Dataspace server class.

The default implementation of the interface provides a continuously mapped dataspace.

Definition at line 40 of file dataspace_svr.

14.256.2 Member Function Documentation

14.256.2.1 allocate()

```
virtual long L4Re::Util::Dataspace_svr::allocate (
            l4_addr_t offset,
            l4_size_t size,
            unsigned access ) throw )   [inline], [virtual]
```

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 168 of file dataspace_svr.

References L4_ENODEV.

Referenced by is_static().

Here is the caller graph for this function:

14.256.2.2 clear()

```
virtual long L4Re::Util::Dataspace_svr::clear (
            unsigned long offs,
            unsigned long size ) const throw )   [virtual]
```

Clear a region in the dataspace.

Parameters

offs	Start of the region
size	Size of the region

Return values

0	Success
<0	Error

Referenced by copy(), and is_static().

Here is the caller graph for this function:

14.256.2.3 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 ) throw )   [inline], [virtual]
```

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 139 of file dataspace_svr.

References clear(), and L4_ENODEV.

Referenced by is_static().

Here is the call graph for this function:

Here is the caller graph for this function:

14.256.2.4 is_static()

```
virtual bool L4Re::Util::Dataspace_svr::is_static ( ) const throw )   [inline], [virtual]
```

Return whether the dataspace is static.

Returns

True if dataspace is static

Definition at line 184 of file dataspace_svr.

References allocate(), clear(), copy(), L4::Ipc::Gen_fpage< T >::data(), L4Re::Dataspace::Stats::flags, L4::Ipc↩ ::Gen_fpage< T >::id_received(), L4_CAP_FPAGE_W, L4_EACCESS, L4_EINVAL, L4_EOK, L4_EPERM, l4_↩ round_size(), map(), page_shift(), phys(), release(), L4Re::Dataspace::Stats::size, and take().

Here is the call graph for this function:

14.256.2.5 map()

```
int L4Re::Util::Dataspace_svr::map (
            l4_addr_t offset,
            l4_addr_t local_addr,
            unsigned long flags,
            l4_addr_t min_addr,
            l4_addr_t 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	Map flags, see L4Re::Dataspace::Map_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

Referenced by is_static().

Here is the caller graph for this function:

14.256.2.6 map_hook()

```
virtual int L4Re::Util::Dataspace_svr::map_hook (
            l4_addr_t offs,
            unsigned long flags,
            l4_addr_t min,
            l4_addr_t 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 90 of file dataspace_svr.

References phys().

Here is the call graph for this function:

14.256.2.7 page_shift()

```
virtual unsigned long L4Re::Util::Dataspace_svr::page_shift ( ) const throw )   [inline],
[virtual]
```

Define the size of the flexpage to map.

Returns

flexpage size

Definition at line 176 of file dataspace_svr.

References L4_LOG2_PAGESIZE.

Referenced by is_static().

Here is the caller graph for this function:

14.256.2.8 phys()

```
virtual int L4Re::Util::Dataspace_svr::phys (
            l4_addr_t offset,
            l4_addr_t & phys_addr,
            l4_size_t & phys_size ) throw )   [virtual]
```

Return physical address for a virtual address.

Parameters

	offset	Offset into the dataspace
out	*phys_addr*	Physical address
out	*phys_size*	Size of continious physical region

Return values

0	Success
<0	Error

Referenced by is_static(), and map_hook().

Here is the caller graph for this function:

14.256.2.9 release()

```
virtual unsigned long L4Re::Util::Dataspace_svr::release ( ) throw )   [inline], [virtual]
```

Release a reference to this dataspace.

Returns

Number of references to the dataspace

Default does nothing and returns always zero.

Definition at line 124 of file dataspace_svr.

Referenced by is_static().

Here is the caller graph for this function:

14.256.2.10 take()

```
virtual void L4Re::Util::Dataspace_svr::take ( ) throw )   [inline], [virtual]
```

Take a reference to this dataspace.

Default does nothing.

Definition at line 114 of file dataspace_svr.

Referenced by is_static().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/re/util/dataspace_svr

14.257 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.

14.257.1 Detailed Description

template<typename PAYLOAD>
class L4Re::Util::Event_buffer_consumer_t< PAYLOAD >

An event buffer consumer.

Definition at line 92 of file event_buffer.

14.257.2 Member Function Documentation

14.257.2.1 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 103 of file event_buffer.

References L4Re::Event_buffer_t< PAYLOAD >::Event::free().

Here is the call graph for this function:

14.257.2.2 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 124 of file event_buffer.

References L4::Rcv_endpoint::bind_thread(), L4::Cap_base::cap(), 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

14.258 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 throw ()

  Return the buffer.
- long attach (L4::Cap< L4Re::Dataspace > ds, L4::Cap< L4Re::Rm > rm) throw ()

  Attach event buffer from address space.
- long detach (L4::Cap< L4Re::Rm > rm) throw ()

  Detach event buffer from address space.

14.258.1 Detailed Description

template<typename PAYLOAD>
class L4Re::Util::Event_buffer_t< PAYLOAD >

Event_buffer utility class.

Definition at line 36 of file event_buffer.

14.258.2 Member Function Documentation

14.258.2.1 attach()

```
template<typename PAYLOAD >
long L4Re::Util::Event_buffer_t< PAYLOAD >::attach (
            L4::Cap< L4Re::Dataspace > ds,
            L4::Cap< L4Re::Rm > rm ) throw )   [inline]
```

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(), and L4Re::Rm::Search_addr.

Here is the call graph for this function:

14.258.2.2 buf()

```
template<typename PAYLOAD >
void* L4Re::Util::Event_buffer_t< PAYLOAD >::buf ( ) const throw )   [inline]
```

Return the buffer.

Returns

Pointer to the event buffer.

Definition at line 46 of file event_buffer.

14.258.2.3 detach()

```
template<typename PAYLOAD >
long L4Re::Util::Event_buffer_t< PAYLOAD >::detach (
            L4::Cap< L4Re::Rm > rm ) throw )   [inline]
```

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 77 of file event_buffer.

The documentation for this class was generated from the following file:

- l4/re/util/event_buffer

14.259 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.

14.259.1 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.

14.259.2 Member Enumeration Documentation

14.259.2.1 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.

14.259.3 Member Function Documentation

14.259.3.1 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 157 of file event.

14.259.3.2 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::Icu::bind(), L4Re::Event::get_buffer(), L4_ENOMEM, l4_error(), L4::Ipc::make_cap_rw(), and L4↩Re::Rm::Search_addr.

Here is the call graph for this function:

14.259.3.3 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 122 of file event.

References L4Re::Event::get_buffer(), L4_ENOMEM, L4::Ipc::make_cap_rw(), and L4Re::Rm::Search_addr.

Here is the call graph for this function:

14.259.3.4 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 164 of file event.

The documentation for this class was generated from the following file:

- l4/re/util/event

14.260 L4Re::Util::Item_alloc_base Class Reference

Item allocator.

Inherited by L4Re::Util::Item_alloc< Bits >.

Collaboration diagram for L4Re::Util::Item_alloc_base:

14.260.1 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

14.261 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

14.261.1 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

14.262 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().

14.262.1 Detailed Description

Abstract server-side implementation of the L4::Namespace interface.

Definition at line 184 of file name_space_svr.

14.262.2 Member Function Documentation

14.262.2.1 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.

14.262.2.2 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.

14.262.2.3 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.

14.262.2.4 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.

14.262.2.5 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.

14.262.2.6 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

14.263 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

14.263.1 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.

14.263.2 Constructor & Destructor Documentation

14.263.2.1 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.

14.263.2.2 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.

References L4::Ipc_svr::Server_iface::alloc_buffer_demand().

Here is the call graph for this function:

14.263.3 Member Function Documentation

14.263.3.1 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.

Returns

The capability the server object was registered with.

The IRQ will be newly allocated using the registry's factory object.

Implements L4::Registry_iface.

Definition at line 229 of file object_registry.

References L4::Registry_iface::register_irq_obj().

Here is the call graph for this function:

14.263.3.2 register_obj() [1/3]

```
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.

Returns

The capability the server object was registered with.

Implements L4::Registry_iface.

Definition at line 201 of file object_registry.

14.263.3.3 register_obj() [2/3]

```
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.

Returns

The capability the server object was registered with.

The IPC gate will be allocated using the registry's factory.

Implements L4::Registry_iface.

Definition at line 215 of file object_registry.

14.263.3.4 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.

Returns

The capability the server object was registered with.

Implements L4::Registry_iface.

Definition at line 247 of file object_registry.

14.263.3.5 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 263 of file object_registry.

References L4::Thread::Modify_senders::add(), L4Re::Util::cap_alloc, L4Re::Util::Counting_cap_alloc< COUNT↩ ERTYPE >::free(), L4_FP_ALL_SPACES, and L4::Thread::modify_senders().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- l4/re/util/object_registry

14.264 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 >:

14.264.1 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 235 of file cap_alloc.

The documentation for this struct was generated from the following file:

- l4/re/util/cap_alloc

14.265 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 >:

14.265.1 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 276 of file cap_alloc.

The documentation for this struct was generated from the following file:

- l4/re/util/cap_alloc

14.266 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 *utcb, 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 ()

  Start the server loop.

Additional Inherited Members

14.266.1 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 293 of file object_registry.

14.266.2 Constructor & Destructor Documentation

14.266.2.1 Registry_server() [1/2]

```
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 305 of file object_registry.

14.266.2.2 Registry_server() [2/2]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (
            l4_utcb_t * utcb,
            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.

Definition at line 315 of file object_registry.

14.266.3 Member Function Documentation

14.266.3.1 registry() [1/2]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
Object_registry const* L4Re::Util::Registry_server< LOOP_HOOKS >::registry ( ) const  [inline]
```

Return registry of this server loop.

Definition at line 321 of file object_registry.

14.266.3.2 registry() [2/2]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
Object_registry* L4Re::Util::Registry_server< LOOP_HOOKS >::registry ( )  [inline]
```

Return registry of this server loop.

Definition at line 323 of file object_registry.

The documentation for this class was generated from the following file:

- l4/re/util/object_registry

14.267 L4Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference

Helper for Auto_cap and Auto_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.

14.267.1 Detailed Description

template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Util::Smart_cap_auto< Unmap_flags >

Helper for Auto_cap and Auto_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

14.268 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) throw ()

  Free operation for L4::Smart_cap (decrement ref count and delete if 0).
- static void invalidate (L4::Cap_base &c) throw ()

  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).

14.268.1 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

14.269 L4Re::Util::Vcon_svr< SVR > Class Template Reference

Console server template class.

Collaboration diagram for L4Re::Util::Vcon_svr< SVR >:

14.269.1 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_R↩EAD_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

14.270 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.

14.270.1 Detailed Description

Goos server class.

Definition at line 36 of file goos_svr.

14.270.2 Member Function Documentation

14.270.2.1 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.

14.270.2.2 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 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_info, 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.

14.270.2.3 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.

Referenced by ~Goos_svr().

Here is the caller graph for this function:

14.270.2.4 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.

14.270.2.5 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

14.271 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 utime (const struct utimbuf ∗) throw ()

  Default backend for POSIX utime.
- int utimes (const struct timeval [2]) throw ()

  Default backend for POSIX utimes.
- 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.

14.271.1 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.

Examples:

tmpfs/lib/src/fs.cc.

Definition at line 40 of file backend.

14.271.2 Member Function Documentation

14.271.2.1 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.

14.271.2.2 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.

14.271.2.3 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

14.272 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.

14.272.1 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.

Examples:

tmpfs/lib/src/fs.cc.

Definition at line 300 of file backend.

14.272.2 Constructor & Destructor Documentation

14.272.2.1 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 314 of file backend.

14.272.2.2 ~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 326 of file backend.

14.272.3 Member Function Documentation

14.272.3.1 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 336 of file backend.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/backend

14.273 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.

14.273.1 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 139 of file vfs.h.

14.273.2 Member Function Documentation

14.273.2.1 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.

14.273.2.2 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.

14.273.2.3 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.

14.273.2.4 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.

14.273.2.5 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.

14.273.2.6 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.

14.273.2.7 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

14.274 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

14.274.1 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 428 of file vfs.h.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

14.275 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.

14.275.1 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 815 of file vfs.h.

14.275.2 Member Function Documentation

14.275.2.1 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:

14.275.2.2 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

14.276 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::Ref_ptr< File > 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.

14.276.1 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 867 of file vfs.h.

14.276.2 Member Function Documentation

14.276.2.1 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.

14.276.2.2 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.

14.276.2.3 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.

14.276.2.4 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.

14.276.2.5 set_fd()

```
virtual cxx::Ref_ptr<File> 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 pointer to the file object that was previously assigned to fd.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

14.277 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).

14.277.1 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 61 of file vfs.h.

14.277.2 Member Function Documentation

14.277.2.1 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.

14.277.2.2 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.

14.277.2.3 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.

14.277.2.4 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.

14.277.2.5 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

14.278 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.

14.278.1 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 738 of file vfs.h.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

14.279 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

14.279.1 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 988 of file vfs.h.

The documentation for this class was generated from the following file:

- l4/l4re_vfs/vfs.h

14.280 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.

14.280.1 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 260 of file vfs.h.

14.280.2 Member Function Documentation

14.280.2.1 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.

14.280.2.2 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.

14.280.2.3 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.

14.280.2.4 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.

14.280.2.5 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.

14.280.2.6 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.

14.280.2.7 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.

14.280.2.8 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.

14.280.2.9 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

14.281 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.

14.281.1 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 393 of file vfs.h.

14.281.2 Member Function Documentation

14.281.2.1 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

14.282 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.

14.282.1 Detailed Description

A color component.

Definition at line 31 of file colors.

14.282.2 Constructor & Destructor Documentation

14.282.2.1 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 46 of file colors.

14.282.3 Member Function Documentation

14.282.3.1 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 93 of file colors.

14.282.3.2 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 73 of file colors.

14.282.3.3 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 65 of file colors.

14.282.3.4 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 84 of file colors.

14.282.3.5 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 59 of file colors.

14.282.3.6 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 53 of file colors.

Referenced by L4Re::Video::Pixel_info::bits_per_pixel(), and L4Re::Video::Pixel_info::has_alpha().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/re/video/colors

14.283 L4Re::Video::Goos Class Reference

A goos.

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 throw ()

  Create a view.
- int delete_view (View const &v, l4_utcb_t *utcb=l4_utcb()) const throw ()

  Delete a view.
- View view (unsigned index) const throw ()

  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

14.283.1 Detailed Description

A goos.

Definition at line 207 of file goos.

14.283.2 Member Enumeration Documentation

14.283.2.1 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 212 of file goos.

14.283.3 Member Function Documentation

14.283.3.1 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

14.283.3.2 create_view()

```
int L4Re::Video::Goos::create_view (
            View * view,
            l4_utcb_t * utcb = l4_utcb() ) const throw )   [inline]
```

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 296 of file goos.

References L4_INLINE_RPC_NF.

14.283.3.3 delete_buffer()

```
long L4Re::Video::Goos::delete_buffer (
            unsigned idx )
```

Delete a buffer.

Parameters

idx	Buffer to delete.

Return values

0	Success
<0	Error

14.283.3.4 delete_view()

```
int L4Re::Video::Goos::delete_view (
            View const & v,
            l4_utcb_t * utcb = l4_utcb() ) const throw )   [inline]
```

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 316 of file goos.

References L4_INLINE_RPC.

14.283.3.5 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

14.283.3.6 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

14.283.3.7 view()

```
View L4Re::Video::Goos::view (
            unsigned index ) const throw )   [inline]
```

Return a view.

Parameters

index	Index of the view to return.

Returns

The view.

Definition at line 347 of file goos.

The documentation for this class was generated from the following file:

- l4/re/video/goos

14.284 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.

14.284.1 Detailed Description

Information structure of a goos.

Definition at line 221 of file goos.

14.284.2 Member Function Documentation

14.284.2.1 auto_refresh()

```
bool L4Re::Video::Goos::Info::auto_refresh ( ) const  [inline]
```

Return whether this goos does auto refreshing or the view refresh functions must be used to make changes visible.

Definition at line 232 of file goos.

The documentation for this struct was generated from the following file:

- l4/re/video/goos

14.285 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.
- 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 ()

  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.

14.285.1 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 106 of file colors.

14.285.2 Constructor & Destructor Documentation

14.285.2.1 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 203 of file colors.

14.285.2.2 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 215 of file colors.

14.285.3 Member Function Documentation

14.285.3.1 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 135 of file colors.

14.285.3.2 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 178 of file colors.

14.285.3.3 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 129 of file colors.

14.285.3.4 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 172 of file colors.

14.285.3.5 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 147 of file colors.

References L4Re::Video::Color_component::size().

Here is the call graph for this function:

14.285.3.6 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 141 of file colors.

14.285.3.7 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 184 of file colors.

14.285.3.8 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 238 of file colors.

Referenced by L4Re::Video::View::Info::dump().

Here is the caller graph for this function:

14.285.3.9 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 123 of file colors.

14.285.3.10 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 166 of file colors.

14.285.3.11 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 154 of file colors.

References L4Re::Video::Color_component::size().

Here is the call graph for this function:

14.285.3.12 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 227 of file colors.

14.285.3.13 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 117 of file colors.

14.285.3.14 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 160 of file colors.

The documentation for this class was generated from the following file:

- l4/re/video/colors

14.286 L4Re::Video::View Class Reference

View.

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 throw ()

  Return the view information of the view.
- int set_info (Info const &info) const throw ()

  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 throw ()

  Set the position of the view in the goos.
- int stack (View const &pivot, bool behind=true) const throw ()

  Move this view in the view stack.
- int push_top () const throw ()

  Make this view the top-most view.
- int push_bottom () const throw ()

  Push this view the back.
- int refresh (int x, int y, int w, int h) const throw ()

  Refresh/Redraw the view.
- bool valid () const

  Return whether this view is valid.

14.286.1 Detailed Description

View.

Definition at line 34 of file goos.

14.286.2 Member Enumeration Documentation

14.286.2.1 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
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 54 of file goos.

14.286.2.2 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 77 of file goos.

14.286.3 Member Function Documentation

14.286.3.1 info()

```
int L4Re::Video::View::info (
            Info * info ) const throw )   [inline]
```

Return the view information of the view.

Parameters

out	*info*	Information structure pointer.

Return values

0	Success
<0	Error

Definition at line 351 of file goos.

14.286.3.2 refresh()

```
int L4Re::Video::View::refresh (
            int x,
            int y,
            int w,
            int h ) const throw )   [inline]
```

Refresh/Redraw the view.

Parameters

x	X position.
y	Y position.
w	Width.
h	Height.

Return values

0	Success
<0	Error

Definition at line 363 of file goos.

14.286.3.3 set_info()

```
int L4Re::Video::View::set_info (
            Info const & info ) const throw )   [inline]
```

Set the information structure for this view.

Parameters

info	Information structure.

Return values

0	Success
<0	Error

The function will also set the view port according to the values given in the information structure.

Definition at line 355 of file goos.

14.286.3.4 set_viewport()

```
int L4Re::Video::View::set_viewport (
            int scr_x,
            int scr_y,
            int w,
            int h,
            unsigned long buf_offset ) const throw )   [inline]
```

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 367 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.

14.286.3.5 stack()

```
int L4Re::Video::View::stack (
            View const & pivot,
            bool behind = true ) const throw )   [inline]
```

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 359 of file goos.

The documentation for this class was generated from the following file:

- l4/re/video/goos

14.287 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.

14.287.1 Detailed Description

Information structure of a view.

Definition at line 86 of file goos.

The documentation for this struct was generated from the following file:

- l4/re/video/goos

14.288 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.

14.288.1 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

14.289 l4re_ds_stats_t Struct Reference

Information about the data space.

```
#include <dataspace.h>
```

Collaboration diagram for l4re_ds_stats_t:

Data Fields

- unsigned long size

  size
- unsigned long flags

  flags

14.289.1 Detailed Description

Information about the data space.

Definition at line 45 of file dataspace.h.

The documentation for this struct was generated from the following file:

- l4/re/c/dataspace.h

14.290 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:

14.290.1 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

14.291 l4re_elf_aux_t Struct Reference

Generic header for each auxiliary vector element.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re_elf_aux_t:

14.291.1 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

14.292 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:

14.292.1 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

14.293 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 ()

  Create an invalid entry.
- l4re_env_cap_entry_t (char const *n, l4_cap_idx_t c, l4_umword_t f=0)

  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.

14.293.1 Detailed Description

Entry in the L4Re environment array for the named inital objects.

Definition at line 46 of file env.h.

14.293.2 Constructor & Destructor Documentation

14.293.2.1 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 77 of file env.h.

References l4re_env_cap_entry_t(), and name.

Here is the call graph for this function:

14.293.3 Field Documentation

14.293.3.1 flags

```
l4_umword_t l4re_env_cap_entry_t::flags
```

Some flags for the object.

Note

Currently unused.

Definition at line 57 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

14.294 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.

14.294.1 Detailed Description

Initial environment data structure.

See also

Initial environment

Definition at line 105 of file env.h.

The documentation for this struct was generated from the following file:

- l4/re/env.h

14.295 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.

14.295.1 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

14.296 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

14.296.1 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

14.297 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.

14.297.1 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

14.298 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.

14.298.1 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

14.299 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.

14.299.1 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

14.300 l4re_video_view_t Struct Reference

C representation of a goos view.

```
#include <view.h>
```

Collaboration diagram for l4re_video_view_t:

14.300.1 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

14.301 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

14.301.1 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

14.302 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)

14.302.1 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

14.303 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

14.303.1 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 47 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

14.304 l4util_mb_apm_t Struct Reference

APM BIOS info.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_apm_t:

14.304.1 Detailed Description

APM BIOS info.

Definition at line 95 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

14.305 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

14.305.1 Detailed Description

Drive Info structure.

Definition at line 78 of file mb_info.h.

14.305.2 Field Documentation

14.305.2.1 drive_cylinders

```
l4_uint16_t l4util_mb_drive_t::drive_cylinders
```

<The access mode (see below).

Definition at line 83 of file mb_info.h.

14.305.2.2 drive_mode

```
l4_uint8_t l4util_mb_drive_t::drive_mode
```

<The BIOS drive number.

Definition at line 82 of file mb_info.h.

14.305.2.3 drive_number

```
l4_uint8_t l4util_mb_drive_t::drive_number
```

<The size of this structure.

Definition at line 81 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

14.306 l4util_mb_info_t Struct Reference

```
#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

14.306.1 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 207 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

14.307 l4util_mb_mod_t Struct Reference

```
#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

14.307.1 Detailed Description

The structure type "mod_list" is used by the multiboot_info structure.

Definition at line 31 of file mb_info.h.

14.307.2 Field Documentation

14.307.2.1 mod_end

```
l4_uint32_t l4util_mb_mod_t::mod_end
```

End address of module in memory.

Definition at line 34 of file mb_info.h.

14.307.2.2 mod_start

```
l4_uint32_t l4util_mb_mod_t::mod_start
```

Starting address of module in memory.

Definition at line 33 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

14.308 l4util_mb_vbe_ctrl_t Struct Reference

VBE controller information.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_vbe_ctrl_t:

14.308.1 Detailed Description

VBE controller information.

Definition at line 109 of file mb_info.h.

The documentation for this struct was generated from the following file:

- l4/util/mb_info.h

14.309 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**
- l4_uint8_t **win_a_attributes**
- l4_uint8_t **win_b_attributes**
- l4_uint16_t **win_granularity**

- l4_uint16_t **win_size**
- l4_uint16_t **win_a_segment**
- l4_uint16_t **win_b_segment**
- l4_uint32_t **win_func**
- l4_uint16_t **bytes_per_scanline**

>= VESA version 1.2

- l4_uint16_t **x_resolution**
- l4_uint16_t **y_resolution**
- l4_uint8_t **x_char_size**
- l4_uint8_t **y_char_size**
- l4_uint8_t **number_of_planes**
- l4_uint8_t **bits_per_pixel**
- l4_uint8_t **number_of_banks**
- l4_uint8_t **memory_model**
- l4_uint8_t **bank_size**
- l4_uint8_t **number_of_image_pages**
- l4_uint8_t **reserved0**

direct color

- l4_uint8_t **red_mask_size**
- l4_uint8_t **red_field_position**
- l4_uint8_t **green_mask_size**
- l4_uint8_t **green_field_position**
- l4_uint8_t **blue_mask_size**
- l4_uint8_t **blue_field_position**
- l4_uint8_t **reserved_mask_size**
- l4_uint8_t **reserved_field_position**
- l4_uint8_t **direct_color_mode_info**

>= VESA version 2.0

- l4_uint32_t **phys_base**
- l4_uint32_t **reserved1**
- l4_uint16_t **reversed2**

>= VESA version 3.0

- l4_uint16_t **linear_bytes_per_scanline**
- l4_uint8_t **banked_number_of_image_pages**
- l4_uint8_t **linear_number_of_image_pages**
- l4_uint8_t **linear_red_mask_size**
- l4_uint8_t **linear_red_field_position**
- l4_uint8_t **linear_green_mask_size**
- l4_uint8_t **linear_green_field_position**
- l4_uint8_t **linear_blue_mask_size**
- l4_uint8_t **linear_blue_field_position**
- l4_uint8_t **linear_reserved_mask_size**
- l4_uint8_t **linear_reserved_field_position**
- l4_uint32_t **max_pixel_clock**
- l4_uint8_t **reserved3** [189+1]

14.309.1 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

14.310 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 human 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.

14.310.1 Detailed Description

Device on a L4vbus::Vbus.

Definition at line 71 of file vbus.

14.310.2 Member Function Documentation

14.310.2.1 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 83 of file vbus.

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:

14.310.2.2 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 92 of file vbus.

14.310.2.3 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 after a successfull call.

Return values

0	Success.
-L4_ENODEV	No device with the given device handle `dev` could be found.

Definition at line 164 of file vbus.

References l4vbus_get_device().

Here is the call graph for this function:

14.310.2.4 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 human 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 124 of file vbus.

14.310.2.5 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 184 of file vbus.

References l4vbus_get_resource().

Here is the call graph for this function:

14.310.2.6 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 198 of file vbus.

References l4vbus_is_compatible().

Here is the call graph for this function:

14.310.2.7 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 found. To start from the beginning `child` must be initialized using the default (L4VBUS_NULL).
	depth	Depth to look for
`out`	*devinfo*	device information (might be NULL)

Returns

0 on success, else failure

Definition at line 145 of file vbus.

14.310.2.8 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 214 of file vbus.

References _bus, and _dev.

14.310.2.9 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 205 of file vbus.

References _bus, and _dev.

14.310.3 Field Documentation

14.310.3.1 _bus

```
L4::Cap<Vbus> L4vbus::Device::_bus  [protected]
```

The Vbus capability (where this device is located on).

Definition at line 220 of file vbus.

Referenced by L4vbus::Gpio_pin::config_get(), L4vbus::Gpio_pin::config_pad(), L4vbus::Gpio_module::config_pad(), L4vbus::Gpio_pin::config_pull(), L4vbus::Gpio_pin::get(), L4vbus::Gpio_module::get(), operator!=(), operator==(), L4vbus::Gpio_pin::set(), L4vbus::Gpio_module::set(), L4vbus::Gpio_pin::setup(), L4vbus::Gpio_module::setup(), and L4vbus::Gpio_pin::to_irq().

The documentation for this class was generated from the following file:

- l4/vbus/vbus

14.311 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

14.311.1 Detailed Description

A Gpio_module groups multiple GPIO pins together.

Definition at line 129 of file vbus_gpio.

14.311.2 Member Function Documentation

14.311.2.1 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 181 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:

14.311.2.2 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 197 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_multi_get().

Here is the call graph for this function:

14.311.2.3 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 225 of file vbus_gpio.

14.311.2.4 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 213 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_multi_set().

Here is the call graph for this function:

14.311.2.5 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 162 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

14.312 L4vbus::Gpio_module::Pin_slice Struct Reference

A slice of the pins provided by this module.

Collaboration diagram for L4vbus::Gpio_module::Pin_slice:

14.312.1 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 142 of file vbus_gpio.

The documentation for this struct was generated from the following file:

- l4/vbus/vbus_gpio

14.313 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

14.313.1 Detailed Description

A GPIO pin.

Definition at line 22 of file vbus_gpio.

14.313.2 Member Function Documentation

14.313.2.1 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 98 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_config_get().

Here is the call graph for this function:

14.313.2.2 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 85 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_config_pad().

Here is the call graph for this function:

14.313.2.3 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 71 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_config_pull().

Here is the call graph for this function:

14.313.2.4 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 34 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_get().

Here is the call graph for this function:

14.313.2.5 pin()

```
unsigned L4vbus::Gpio_pin::pin ( ) const  [inline]
```

Get pin number.

Returns

GPIO pin number

Definition at line 118 of file vbus_gpio.

14.313.2.6 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 45 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_set().

Here is the call graph for this function:

14.313.2.7 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 60 of file vbus_gpio.

References L4vbus::Device::_bus, L4vbus::Device::_dev, and l4vbus_gpio_setup().

Here is the call graph for this function:

14.313.2.8 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 108 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

14.314 L4vbus::Icu Class Reference

Vbus Interrupt controller API.

Inheritance diagram for L4vbus::Icu:

Collaboration diagram for L4vbus::Icu:

Public Member Functions

- int vicu (L4::Cap< L4::Icu > icu) const

  Request the L4::Icu capability for this Vbus ICU.

Additional Inherited Members

14.314.1 Detailed Description

Vbus Interrupt controller API.

Allows to access the underlying L4::Icu capability managing IRQs for the L4vbus::Vbus.

See also

L4::Icu

Definition at line 234 of file vbus.

The documentation for this class was generated from the following file:

- l4/vbus/vbus

14.315 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

14.315.1 Detailed Description

A PCI device.

Definition at line 87 of file vbus_pci.

14.315.2 Member Function Documentation

14.315.2.1 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 99 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:

14.315.2.2 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 115 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:

14.315.2.3 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 131 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

14.316 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

14.316.1 Detailed Description

A Pci host bridge.

Definition at line 20 of file vbus_pci.

14.316.2 Member Function Documentation

14.316.2.1 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 34 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:

14.316.2.2 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 53 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:

14.316.2.3 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 74 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

14.317 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.

14.317.1 Detailed Description

template<typename DEC>
class L4vbus::Pm< DEC >

Power-management API mixin.

Definition at line 47 of file vbus.

The documentation for this class was generated from the following file:

- l4/vbus/vbus

14.318 L4vbus::Vbus Class Reference

The virtual bus (Vbus) interface.

Inheritance diagram for L4vbus::Vbus:

Collaboration diagram for L4vbus::Vbus:

Public Member Functions

- int request_resource (l4vbus_resource_t *res, int flags=0) const

  Request the given 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

14.318.1 Detailed Description

The virtual bus (Vbus) interface.

See also

L4Re Vbus API

Definition at line 253 of file vbus.

14.318.2 Member Function Documentation

14.318.2.1 assign_dma_domain() [1/2]

```
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.
-L4_EBUSY	DMA domain is already active, this means another DMA space is already assigned.

Definition at line 303 of file vbus.

References L4::Cap_base::cap(), l4vbus_assign_dma_domain(), L4VBUS_DMAD_KERNEL_DMA_SPACE, and L4VBUS_DMAD_L4RE_DMA_SPACE.

Here is the call graph for this function:

14.318.2.2 assign_dma_domain() [2/2]

```
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.
-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 328 of file vbus.

References L4::Cap_base::cap(), l4vbus_assign_dma_domain(), L4VBUS_DMAD_KERNEL_DMA_SPACE, and L4VBUS_DMAD_L4RE_DMA_SPACE.

Here is the call graph for this function:

14.318.2.3 release_resource()

```
int L4vbus::Vbus::release_resource (
            l4vbus_resource_t * res ) const  [inline]
```

Release the given resource from the bus.

Parameters

res	The resource that shall be requested from the bus.

Returns

>=0 on success, <0 on error.

Definition at line 273 of file vbus.

References l4vbus_release_resource().

Here is the call graph for this function:

14.318.2.4 request_resource()

```
int L4vbus::Vbus::request_resource (
            l4vbus_resource_t * res,
            int flags = 0 ) const [inline]
```

Request the given resource from the bus.

Parameters

res	The resource that shall be requested from the bus.
flags	The flags for the request.

Returns

>=0 on success, <0 on error.

Definition at line 263 of file vbus.

References l4vbus_request_resource().

Here is the call graph for this function:

14.318.2.5 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 282 of file vbus.

References L4VBUS_ROOT_BUS.

The documentation for this class was generated from the following file:

- l4/vbus/vbus

14.319 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.

14.319.1 Detailed Description

Detailed information about a vbus device.

Definition at line 56 of file vbus_types.h.

The documentation for this struct was generated from the following file:

- l4/vbus/vbus_types.h

14.320 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.

14.320.1 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

14.321 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.

14.321.1 Detailed Description

C++ implementation of state word in the vCPU area.

Definition at line 31 of file vcpu.

14.321.2 Constructor & Destructor Documentation

14.321.2.1 State()

```
L4vcpu::State::State (
            unsigned v )  [inline], [explicit]
```

Initialize state.

Parameters

v	Initial state.

Definition at line 41 of file vcpu.

14.321.3 Member Function Documentation

14.321.3.1 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.

14.321.3.2 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.

14.321.3.3 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

14.322 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.

14.322.1 Detailed Description

C++ implementation of the vCPU save state area.

Definition at line 72 of file vcpu.

14.322.2 Member Function Documentation

14.322.2.1 cast() [1/2]

```
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.

14.322.2.2 cast() [2/2]

```
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.

14.322.2.3 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, L4Re::Env::env(), and L4_CV.

Here is the call graph for this function:

14.322.2.4 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.

14.322.2.5 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

14.322.2.6 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.

14.322.2.7 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.

14.322.2.8 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:

14.322.2.9 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:

14.322.2.10 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:

14.322.2.11 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:

14.322.2.12 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:

14.322.2.13 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.

14.322.2.14 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.

14.322.2.15 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.

14.322.2.16 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.

14.322.2.17 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.

14.322.2.18 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.

14.322.2.19 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.

14.322.2.20 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

14.323 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

14.323.1 Detailed Description

Driver-side implementation of a Virtqueue.

Adds function for managing the descriptor list, enqueueing new and dequeueing finished requests.

Definition at line 471 of file virtqueue.

14.323.2 Member Function Documentation

14.323.2.1 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 559 of file virtqueue.

14.323.2.2 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 593 of file virtqueue.

14.323.2.3 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 576 of file virtqueue.

14.323.2.4 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 612 of file virtqueue.

14.323.2.5 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 635 of file virtqueue.

14.323.2.6 init_queue() [1/2]

```
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 523 of file virtqueue.

14.323.2.7 init_queue() [2/2]

```
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 538 of file virtqueue.

14.323.2.8 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 495 of file virtqueue.

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

14.324 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

14.324.1 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.

14.324.2 Member Enumeration Documentation

14.324.2.1 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.

14.324.3 Member Function Documentation

14.324.3.1 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< Ds_data >::load_desc(), and L4virtio::Svr::Driver_mem_region_t< Ds_data >::local().

Here is the caller graph for this function:

14.324.3.2 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

14.325 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.

14.325.1 Detailed Description

Exception used by Queue to indicate descriptor errors.

Definition at line 332 of file virtio.

14.325.2 Member Enumeration Documentation

14.325.2.1 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 335 of file virtio.

14.325.3 Constructor & Destructor Documentation

14.325.3.1 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 355 of file virtio.

14.325.4 Member Function Documentation

14.325.4.1 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 364 of file virtio.

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

14.326 L4virtio::Svr::Block_dev< Ds_data > Class Template Reference

Base class for virtio block devices.

Inheritance diagram for L4virtio::Svr::Block_dev< Ds_data >:

Collaboration diagram for L4virtio::Svr::Block_dev< Ds_data >:

Public Member Functions

- Block_dev (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 reset_client ()

  The client requests reinitialisation of the connection.

- 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)

  callback for client queue-config request

- L4::Cap< void > register_obj (L4::Registry_iface *registry, char const *service=0)

  Attach device to an object registry.

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.

Additional Inherited Members

14.326.1 Detailed Description

template<typename Ds_data>
class L4virtio::Svr::Block_dev< Ds_data >

Base class for virtio block devices.

Use this class as a base to implement your own specific block device.

Definition at line 23 of file virtio-block.

14.326.2 Constructor & Destructor Documentation

14.326.2.1 Block_dev()

```
template<typename Ds_data>
L4virtio::Svr::Block_dev< Ds_data >::Block_dev (
            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 378 of file virtio-block.

14.326.3 Member Function Documentation

14.326.3.1 finalize_request()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< 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 441 of file virtio-block.

14.326.3.2 process_request()

```
template<typename Ds_data>
virtual bool L4virtio::Svr::Block_dev< 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 needs to be set accordingly and the request immediately finished with finish_request() if the client is to be answered.

14.326.3.3 register_obj()

```
template<typename Ds_data>
L4::Cap<void> L4virtio::Svr::Block_dev< Ds_data >::register_obj (
            L4::Registry_iface * registry,
            char const * service = 0 )  [inline]
```

Attach device to an object registry.

Parameters

registry	Object registry that will be responsible for dispatching requests.
service	Name of an existing capability the device should use.

This functions registers the general virtio interface as well as the interrupt handler which is used for receiving client notifications.

Definition at line 475 of file virtio-block.

References L4::cap_cast(), L4Re::chkcap(), L4Re::chksys(), L4virtio::Virtqueue::disable(), L4_EINVAL, L4virtio::Svr::Virtqueue::next_avail(), L4virtio::Virtqueue::ready(), L4::Registry_iface::register_irq_obj(), and L4::Registry_iface::register_obj().

Here is the call graph for this function:

14.326.3.4 reset_client()

```
template<typename Ds_data>
virtual bool L4virtio::Svr::Block_dev< Ds_data >::reset_client ( ) [inline], [virtual]
```

The client requests reinitialisation of the connection.

Returns

False if reinitialisation is not supported.
True if reinitialisation is supported and the client has been reinitialised successfully.

Definition at line 423 of file virtio-block.

14.326.3.5 set_blk_size()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< 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 337 of file virtio-block.

14.326.3.6 set_size_max()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< 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 incomming 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 295 of file virtio-block.

14.326.3.7 set_topology()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< 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 353 of file virtio-block.

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio-block

14.327 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.

14.327.1 Detailed Description

template<typename Ds_data>
class L4virtio::Svr::Block_request< Ds_data >

A request to read or write data.

Definition at line 29 of file virtio-block.

14.327.2 Member Function Documentation

14.327.2.1 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 64 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:

14.327.2.2 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 114 of file virtio-block.

References L4virtio::Svr::Bad_descriptor::Bad_size, and L4_EEXIST.

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio-block

14.328 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.

14.328.1 Detailed Description

Abstract data buffer.

Definition at line 246 of file virtio.

14.328.2 Constructor & Destructor Documentation

14.328.2.1 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 262 of file virtio.

14.328.3 Member Function Documentation

14.328.3.1 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 291 of file virtio.

References left, cxx::max(), cxx::min(), and pos.

Here is the call graph for this function:

14.328.3.2 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 323 of file virtio.

14.328.3.3 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 275 of file virtio.

14.328.3.4 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 311 of file virtio.

References cxx::min().

Here is the call graph for this function:

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

14.329 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_CON↩ FIG >.

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)

  Setup/Create a L4-Virtio config data space.
- l4_uint32_t guest_features (unsigned idx) const

  Return a specific set of guest 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_failed ()

  Set device status failed bit.
- 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.

14.329.1 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 56 of file l4virtio.

14.329.2 Constructor & Destructor Documentation

14.329.2.1 Dev_config()

```
L4virtio::Svr::Dev_config::Dev_config (
            l4_uint32_t vendor,
            l4_uint32_t device,
            unsigned cfg_size,
            l4_uint32_t num_queues = 0 )  [inline]
```

Setup/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 96 of file l4virtio.

References L4Re::chkcap(), L4Re::chksys(), L4Re::Env::env(), L4_PAGESIZE, l4virtio_clear_feature(), l4virtio_get_feature(), l4virtio_set_feature(), L4::Ipc::make_cap_rw(), reset_hdr(), and L4Re::Rm::Search_addr.

Referenced by ds().

Here is the call graph for this function:

Here is the caller graph for this function:

14.329.3 Member Function Documentation

14.329.3.1 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 219 of file l4virtio.

References L4_PAGESIZE, and reset_hdr().

Here is the call graph for this function:

14.329.3.2 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 308 of file l4virtio.

References Dev_config().

Referenced by L4virtio::Svr::Driver_mem_list_t< Ds_data >::add(), L4virtio::Svr::Device_t< Ds_data >::handle↩_mem_cmd_write(), and L4virtio::Svr::Device_t< Ds_data >::mem_info().

Here is the call graph for this function:

Here is the caller graph for this function:

14.329.3.3 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 174 of file l4virtio.

References l4virtio_config_hdr_t::cmd, and hdr().

Referenced by L4virtio::Svr::Device_t< Ds_data >::handle_mem_cmd_write().

Here is the call graph for this function:

Here is the caller graph for this function:

14.329.3.4 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 156 of file l4virtio.

14.329.3.5 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 301 of file l4virtio.

Referenced by get_cmd(), reset_cmd(), set_failed(), and set_status().

Here is the caller graph for this function:

14.329.3.6 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 236 of file l4virtio.

References L4_UNLIKELY.

Referenced by reset_queue(), and L4virtio::Svr::Device_t< Ds_data >::setup_queue().

Here is the caller graph for this function:

14.329.3.7 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 185 of file l4virtio.

References l4virtio_config_hdr_t::cmd, and hdr().

Referenced by L4virtio::Svr::Device_t< Ds_data >::handle_mem_cmd_write().

Here is the call graph for this function:

Here is the caller graph for this function:

14.329.3.8 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 descriptor 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 278 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< Ds_data >::reset_queue_config().

Here is the call graph for this function:

Here is the caller graph for this function:

14.329.3.9 set_failed()

```
void L4virtio::Svr::Dev_config::set_failed ( ) [inline]
```

Set device status failed bit.

This function sets the internal status register and also the status register in the shared memory to *status*.

Definition at line 209 of file l4virtio.

References L4virtio::Svr::Dev_status::failed(), hdr(), L4virtio::Svr::Dev_status::raw, and l4virtio_config_hdr_t↩ ::status.

Referenced by L4virtio::Svr::Device_t< Ds_data >::device_error().

Here is the call graph for this function:

Here is the caller graph for this function:

14.329.3.10 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 197 of file l4virtio.

References hdr(), L4virtio::Svr::Dev_status::raw, l4virtio_config_hdr_t::status, and status().

Referenced by L4virtio::Svr::Device_t< Ds_data >::handle_mem_cmd_write().

Here is the call graph for this function:

Here is the caller graph for this function:

14.329.3.11 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 166 of file l4virtio.

Referenced by L4virtio::Svr::Device_t< Ds_data >::handle_mem_cmd_write(), L4virtio::Svr::Device_t< Ds_data >::mem_info(), 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

14.330 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 .

14.330.1 Detailed Description

Type for device feature bitmap.

Definition at line 75 of file virtio.

14.330.2 Member Typedef Documentation

14.330.2.1 ring_event_idx_bfm_t

```
typedef cxx::Bitfield<decltype( raw ), 29 , 29 > L4virtio::Svr::Dev_features::ring_event_idx↩
_bfm_t
```

Type to access the *ring_event_idx* bits (29 to 29) of *raw* .

Definition at line 83 of file virtio.

14.330.2.2 ring_indirect_desc_bfm_t

```
typedef cxx::Bitfield<decltype( raw ), 28 , 28 > L4virtio::Svr::Dev_features::ring_indirect_↩
desc_bfm_t
```

Type to access the *ring_indirect_desc* bits (28 to 28) of *raw* .

Definition at line 83 of file virtio.

14.330.3 Member Function Documentation

14.330.3.1 ring_event_idx() [1/2]

```
ring_event_idx_bfm_t::Val L4virtio::Svr::Dev_features::ring_event_idx ( ) const  [inline]
```

Get the *ring_event_idx* bits (29 to 29) of *raw* .

Definition at line 84 of file virtio.

14.330.3.2 ring_event_idx() [2/2]

```
ring_event_idx_bfm_t::Ref L4virtio::Svr::Dev_features::ring_event_idx ( )  [inline]
```

Get a reference to the *ring_event_idx* bits (29 to 29) of *raw* .

Definition at line 84 of file virtio.

14.330.3.3 ring_indirect_desc() [1/2]

```
ring_indirect_desc_bfm_t::Val L4virtio::Svr::Dev_features::ring_indirect_desc ( ) const  [inline]
```

Get the *ring_indirect_desc* bits (28 to 28) of *raw* .

Definition at line 83 of file virtio.

14.330.3.4 ring_indirect_desc() [2/2]

```
ring_indirect_desc_bfm_t::Ref L4virtio::Svr::Dev_features::ring_indirect_desc ( )  [inline]
```

Get a reference to the *ring_indirect_desc* bits (28 to 28) of *raw* .

Definition at line 83 of file virtio.

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

14.331 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), 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 .

14.331.1 Detailed Description

Type of the device status register.

Definition at line 44 of file virtio.

14.331.2 Member Typedef Documentation

14.331.2.1 acked_bfm_t

```
typedef cxx::Bitfield<decltype( raw ), 0 , 0 > L4virtio::Svr::Dev_status::acked_bfm_t
```

Type to access the *acked* bits (0 to 0) of *raw* .

Definition at line 52 of file virtio.

14.331.2.2 driver_bfm_t

```
typedef cxx::Bitfield<decltype( raw ), 1 , 1 > L4virtio::Svr::Dev_status::driver_bfm_t
```

Type to access the *driver* bits (1 to 1) of *raw* .

Definition at line 52 of file virtio.

14.331.2.3 driver_ok_bfm_t

```
typedef cxx::Bitfield<decltype( raw ), 2 , 2 > L4virtio::Svr::Dev_status::driver_ok_bfm_t
```

Type to access the *driver_ok* bits (2 to 2) of *raw* .

Definition at line 53 of file virtio.

14.331.2.4 failed_bfm_t

```
typedef cxx::Bitfield<decltype( raw ), 7 , 7 > L4virtio::Svr::Dev_status::failed_bfm_t
```

Type to access the *failed* bits (7 to 7) of *raw* .

Definition at line 55 of file virtio.

14.331.2.5 features_ok_bfm_t

```
typedef cxx::Bitfield<decltype( raw ), 3 , 3 > L4virtio::Svr::Dev_status::features_ok_bfm_t
```

Type to access the *features_ok* bits (3 to 3) of *raw* .

Definition at line 54 of file virtio.

14.331.3 Member Function Documentation

14.331.3.1 acked() [1/2]

```
acked_bfm_t::Val L4virtio::Svr::Dev_status::acked ( ) const  [inline]
```

Get the *acked* bits (0 to 0) of *raw* .

Definition at line 52 of file virtio.

Referenced by L4virtio::Svr::Device_t< Ds_data >::mem_info().

Here is the caller graph for this function:

14.331.3.2 acked() [2/2]

```
acked_bfm_t::Ref L4virtio::Svr::Dev_status::acked ( )  [inline]
```

Get a reference to the *acked* bits (0 to 0) of *raw* .

Definition at line 52 of file virtio.

14.331.3.3 driver() [1/2]

```
driver_bfm_t::Ref L4virtio::Svr::Dev_status::driver ( )  [inline]
```

Get a reference to the *driver* bits (1 to 1) of *raw* .

Definition at line 53 of file virtio.

14.331.3.4 driver() [2/2]

```
driver_bfm_t::Val L4virtio::Svr::Dev_status::driver ( ) const  [inline]
```

Get the *driver* bits (1 to 1) of *raw* .

Definition at line 53 of file virtio.

Referenced by L4virtio::Svr::Device_t< Ds_data >::mem_info().

Here is the caller graph for this function:

14.331.3.5 driver_ok() [1/2]

```
driver_ok_bfm_t::Val L4virtio::Svr::Dev_status::driver_ok ( ) const  [inline]
```

Get the *driver_ok* bits (2 to 2) of *raw* .

Definition at line 54 of file virtio.

14.331.3.6 driver_ok() [2/2]

```
driver_ok_bfm_t::Ref L4virtio::Svr::Dev_status::driver_ok ( )  [inline]
```

Get a reference to the *driver_ok* bits (2 to 2) of *raw* .

Definition at line 54 of file virtio.

14.331.3.7 failed() [1/2]

```
failed_bfm_t::Val L4virtio::Svr::Dev_status::failed ( ) const [inline]
```

Get the *failed* bits (7 to 7) of *raw* .

Definition at line 56 of file virtio.

Referenced by L4virtio::Svr::Device_t< Ds_data >::handle_mem_cmd_write(), L4virtio::Svr::Device_t< Ds_data >::mem_info(), and L4virtio::Svr::Dev_config::set_failed().

Here is the caller graph for this function:

14.331.3.8 failed() [2/2]

```
failed_bfm_t::Ref L4virtio::Svr::Dev_status::failed ( ) [inline]
```

Get a reference to the *failed* bits (7 to 7) of *raw* .

Definition at line 56 of file virtio.

14.331.3.9 features_ok() [1/2]

```
features_ok_bfm_t::Ref L4virtio::Svr::Dev_status::features_ok ( ) [inline]
```

Get a reference to the *features_ok* bits (3 to 3) of *raw* .

Definition at line 55 of file virtio.

14.331.3.10 features_ok() [2/2]

```
features_ok_bfm_t::Val L4virtio::Svr::Dev_status::features_ok ( ) const  [inline]
```

Get the *features_ok* bits (3 to 3) of *raw* .

Definition at line 55 of file virtio.

14.331.3.11 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 failed() returns false.

Definition at line 66 of file virtio.

Referenced by L4virtio::Svr::Device_t< Ds_data >::handle_mem_cmd_write().

Here is the caller graph for this function:

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

14.332 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_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 ()=0

  callback for registering a single guest IRQ for all queues
- virtual L4::Cap< L4::Irq > device_notify_irq () const =0

  callback to gather the device notification IRQ
- 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 failed 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.

14.332.1 Detailed Description

template<typename DATA>
class L4virtio::Svr::Device_t< DATA >

Server-side L4-VIRTIO device stub.

Definition at line 694 of file l4virtio.

14.332.2 Member Function Documentation

14.332.2.1 device_error()

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::device_error ( ) [inline]
```

Transition device into failed state.

Note

Callers should trigger a guest config IRQ after calling this function.

This function does a full reset, (calls reset()) and sets the failed bit in the device status register.

Definition at line 819 of file l4virtio.

14.332.2.2 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 939 of file l4virtio.

14.332.2.3 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 806 of file l4virtio.

14.332.2.4 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, name num_max to the given value.

Definition at line 796 of file l4virtio.

14.332.2.5 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 838 of file l4virtio.

The documentation for this class was generated from the following file:

- l4/l4virtio/server/l4virtio

14.333 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.

14.333.1 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 531 of file l4virtio.

14.333.2 Member Function Documentation

14.333.2.1 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 571 of file l4virtio.

Referenced by L4virtio::Svr::Device_t< Ds_data >::setup_queue().

Here is the caller graph for this function:

14.333.2.2 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 605 of file l4virtio.

Referenced by L4virtio::Svr::Device_t< Ds_data >::setup_queue().

Here is the caller graph for this function:

14.333.2.3 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 560 of file l4virtio.

Referenced by L4virtio::Svr::Device_t< Ds_data >::setup_queue().

Here is the caller graph for this function:

14.333.2.4 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 552 of file l4virtio.

Referenced by L4virtio::Svr::Device_t< Ds_data >::init_mem_info().

Here is the caller graph for this function:

14.333.2.5 load_desc() [1/3]

```
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 619 of file l4virtio.

14.333.2.6 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 639 of file l4virtio.

14.333.2.7 load_desc() [3/3]

```
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 666 of file l4virtio.

14.333.2.8 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 585 of file l4virtio.

The documentation for this class was generated from the following file:

- l4/l4virtio/server/l4virtio

14.334 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 memroy 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.

14.334.1 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 358 of file l4virtio.

14.334.2 Constructor & Destructor Documentation

14.334.2.1 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 403 of file l4virtio.

14.334.3 Member Function Documentation

14.334.3.1 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 494 of file l4virtio.

14.334.3.2 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 473 of file l4virtio.

14.334.3.3 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 485 of file l4virtio.

14.334.3.4 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 482 of file l4virtio.

14.334.3.5 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 else.

Definition at line 469 of file l4virtio.

14.334.3.6 flags()

```
template<typename DATA>
Flags L4virtio::Svr::Driver_mem_region_t< DATA >::flags ( ) const  [inline]
```

Returns

The flags for this region.

Definition at line 466 of file l4virtio.

14.334.3.7 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 else.

Definition at line 463 of file l4virtio.

14.334.3.8 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 518 of file l4virtio.

Referenced by L4virtio::Svr::Driver_mem_list_t< Ds_data >::load_desc().

Here is the caller graph for this function:

14.334.3.9 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 476 of file l4virtio.

14.334.3.10 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 479 of file l4virtio.

The documentation for this class was generated from the following file:

- l4/l4virtio/server/l4virtio

14.335 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.

14.335.1 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 405 of file virtio.

14.335.2 Member Function Documentation

14.335.2.1 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 475 of file virtio.

References L4virtio::Virtqueue::Desc::flags.

14.335.2.2 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 482 of file virtio.

References L4virtio::Virtqueue::Desc::flags, and L4virtio::Virtqueue::Desc::Flags::next().

Referenced by L4virtio::Svr::Block_request< Ds_data >::data_size().

Here is the call graph for this function:

Here is the caller graph for this function:

14.335.2.3 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

DESCM_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 499 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().

Here is the call graph for this function:

Here is the caller graph for this function:

14.335.2.4 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

DESCM_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 434 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().

Here is the call graph for this function:

Here is the caller graph for this function:

14.335.2.5 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

DESCM_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 465 of file virtio.

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio

14.336 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

14.336.1 Detailed Description

Virtqueue implementation for the device.

This class represents a single virtqueue, with a local running available index.

Definition at line 94 of file virtio.

14.336.2 Member Function Documentation

14.336.2.1 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 173 of file virtio.

Referenced by L4virtio::Svr::Block_request< Ds_data >::header().

Here is the caller graph for this function:

14.336.2.2 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 238 of file virtio.

Referenced by L4virtio::Svr::Request_processor::start().

Here is the caller graph for this function:

14.336.2.3 desc_avail()

```
bool L4virtio::Svr::Virtqueue::desc_avail ( ) const  [inline]
```

Test for available descriptors.

Returns

true if there are descriptors availabale, false if not.

Precondition

The queue must be in working state.

Definition at line 161 of file virtio.

14.336.2.4 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 215 of file virtio.

References L4_LIKELY.

14.336.2.5 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 226 of file virtio.

References L4_LIKELY.

14.336.2.6 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 144 of file virtio.

References L4_LIKELY.

Referenced by L4virtio::Svr::Block_dev< Ds_data >::register_obj().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio

14.337 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

14.337.1 Detailed Description

VIRTIO request, essentially a descriptor from the available ring.

Definition at line 100 of file virtio.

14.337.2 Member Function Documentation

14.337.2.1 desc()

```
Desc const* L4virtio::Svr::Virtqueue::Head_desc::desc ( ) const  [inline]
```

Returns

Pointer to the head descriptor of the request.

Definition at line 121 of file virtio.

Referenced by L4virtio::Svr::Request_processor::start().

Here is the caller graph for this function:

14.337.2.2 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 117 of file virtio.

14.337.2.3 valid()

```
bool L4virtio::Svr::Virtqueue::Head_desc::valid ( ) const  [inline]
```

Returns

True if the request is valid (not NULL).

Definition at line 114 of file virtio.

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio

14.338 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.

14.338.1 Detailed Description

Low-level Virtqueue.

This class represents a single virtqueue, with a local running available index.

Definition at line 87 of file virtqueue.

14.338.2 Member Function Documentation

14.338.2.1 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 291 of file virtqueue.

14.338.2.2 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 283 of file virtqueue.

Referenced by L4virtio::Svr::Device_t< Ds_data >::setup_queue().

Here is the caller graph for this function:

14.338.2.3 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 273 of file virtqueue.

14.338.2.4 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 265 of file virtqueue.

Referenced by L4virtio::Svr::Device_t< Ds_data >::setup_queue().

Here is the caller graph for this function:

14.338.2.5 disable()

```
void L4virtio::Virtqueue::disable ( ) [inline]
```

Completely disable the queue.

setup() must be used to enable the queue again.

Definition at line 228 of file virtqueue.

Referenced by L4virtio::Svr::Block_dev< Ds_data >::register_obj(), and L4virtio::Svr::Device_t< Ds_data >↩ ::setup_queue().

Here is the caller graph for this function:

14.338.2.6 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 395 of file virtqueue.

References L4virtio::Virtqueue::Desc::dump().

Here is the call graph for this function:

14.338.2.7 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 452 of file virtqueue.

References L4virtio::Virtqueue::Avail::idx.

14.338.2.8 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 459 of file virtqueue.

14.338.2.9 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 417 of file virtqueue.

References L4virtio::Virtqueue::Avail::flags, and L4virtio::Virtqueue::Avail::Flags::no_irq().

Here is the call graph for this function:

14.338.2.10 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 429 of file virtqueue.

References L4virtio::Virtqueue::Used::flags, and L4virtio::Virtqueue::Used::Flags::no_notify().

Here is the call graph for this function:

14.338.2.11 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 439 of file virtqueue.

References L4virtio::Virtqueue::Used::flags, and L4virtio::Virtqueue::Used::Flags::no_notify().

Here is the call graph for this function:

14.338.2.12 num()

```
unsigned L4virtio::Virtqueue::num ( ) const  [inline]
```

Returns

The number of entries in the ring.

Definition at line 407 of file virtqueue.

Referenced by L4virtio::Svr::Request_processor::start().

Here is the caller graph for this function:

14.338.2.13 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 403 of file virtqueue.

References L4_LIKELY.

Referenced by L4virtio::Svr::Block_dev< Ds_data >::register_obj().

Here is the caller graph for this function:

14.338.2.14 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 353 of file virtqueue.

References L4_EINVAL.

Referenced by L4virtio::Svr::Device_t< Ds_data >::setup_queue().

Here is the caller graph for this function:

14.338.2.15 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 382 of file virtqueue.

References l4_round_size().

Here is the call graph for this function:

14.338.2.16 total_size() [1/2]

```
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 249 of file virtqueue.

14.338.2.17 total_size() [2/2]

```
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 318 of file virtqueue.

14.338.2.18 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 310 of file virtqueue.

14.338.2.19 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 302 of file virtqueue.

Referenced by L4virtio::Svr::Device_t< Ds_data >::setup_queue().

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

14.339 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.

14.339.1 Detailed Description

Type of available ring, this is read-only for the host.

Definition at line 134 of file virtqueue.

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

14.340 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 .

14.340.1 Detailed Description

Flags of the available ring.

Definition at line 140 of file virtqueue.

14.340.2 Member Typedef Documentation

14.340.2.1 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 149 of file virtqueue.

14.340.3 Member Function Documentation

14.340.3.1 no_irq() [1/2]

```
no_irq_bfm_t::Val L4virtio::Virtqueue::Avail::Flags::no_irq ( ) const  [inline]
```

Get the *no_irq* bits (0 to 0) of *raw* .

Definition at line 149 of file virtqueue.

Referenced by L4virtio::Virtqueue::no_notify_guest().

Here is the caller graph for this function:

14.340.3.2 no_irq() [2/2]

```
no_irq_bfm_t::Ref L4virtio::Virtqueue::Avail::Flags::no_irq ( ) [inline]
```

Get a reference to the *no_irq* bits (0 to 0) of *raw* .

Definition at line 149 of file virtqueue.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtqueue

14.341 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.

14.341.1 Detailed Description

Descriptor in the descriptor table.

Definition at line 93 of file virtqueue.

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

14.342 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 .

14.342.1 Detailed Description

Type for descriptor flags.

Definition at line 99 of file virtqueue.

14.342.2 Member Typedef Documentation

14.342.2.1 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 110 of file virtqueue.

14.342.2.2 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 108 of file virtqueue.

14.342.2.3 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 108 of file virtqueue.

14.342.3 Member Function Documentation

14.342.3.1 indirect() [1/2]

```
indirect_bfm_t::Val L4virtio::Virtqueue::Desc::Flags::indirect ( ) const  [inline]
```

Get the *indirect* bits (2 to 2) of *raw* .

Definition at line 112 of file virtqueue.

Referenced by L4virtio::Svr::Request_processor::next(), and L4virtio::Svr::Request_processor::start().

Here is the caller graph for this function:

14.342.3.2 indirect() [2/2]

```
indirect_bfm_t::Ref L4virtio::Virtqueue::Desc::Flags::indirect ( ) [inline]
```

Get a reference to the *indirect* bits (2 to 2) of *raw* .

Definition at line 112 of file virtqueue.

14.342.3.3 next() [1/2]

```
next_bfm_t::Ref L4virtio::Virtqueue::Desc::Flags::next ( ) [inline]
```

Get a reference to the *next* bits (0 to 0) of *raw* .

Definition at line 108 of file virtqueue.

14.342.3.4 next() [2/2]

```
next_bfm_t::Val L4virtio::Virtqueue::Desc::Flags::next ( ) const [inline]
```

Get the *next* bits (0 to 0) of *raw* .

Definition at line 108 of file virtqueue.

Referenced by L4virtio::Svr::Request_processor::has_more(), and L4virtio::Svr::Request_processor::next().

Here is the caller graph for this function:

14.342.3.5 write() [1/2]

```
write_bfm_t::Val L4virtio::Virtqueue::Desc::Flags::write ( ) const [inline]
```

Get the *write* bits (1 to 1) of *raw* .

Definition at line 110 of file virtqueue.

14.342.3.6 write() [2/2]

```
write_bfm_t::Ref L4virtio::Virtqueue::Desc::Flags::write ( )  [inline]
```

Get a reference to the *write* bits (1 to 1) of *raw* .

Definition at line 110 of file virtqueue.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtqueue

14.343 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.

14.343.1 Detailed Description

Used ring.

Definition at line 179 of file virtqueue.

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

14.344 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 .

14.344.1 Detailed Description

flags for the used ring.

Definition at line 185 of file virtqueue.

14.344.2 Member Typedef Documentation

14.344.2.1 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 194 of file virtqueue.

14.344.3 Member Function Documentation

14.344.3.1 no_notify() [1/2]

```
no_notify_bfm_t::Val L4virtio::Virtqueue::Used::Flags::no_notify ( ) const  [inline]
```

Get the *no_notify* bits (0 to 0) of *raw* .

Definition at line 194 of file virtqueue.

Referenced by L4virtio::Virtqueue::no_notify_host().

Here is the caller graph for this function:

14.344.3.2 no_notify() [2/2]

```
no_notify_bfm_t::Ref L4virtio::Virtqueue::Used::Flags::no_notify ( )  [inline]
```

Get a reference to the *no_notify* bits (0 to 0) of *raw* .

Definition at line 194 of file virtqueue.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtqueue

14.345 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

14.345.1 Detailed Description

Type of an element of the used ring.

Definition at line 160 of file virtqueue.

14.345.2 Constructor & Destructor Documentation

14.345.2.1 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 171 of file virtqueue.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtqueue

14.346 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.

14.346.1 Detailed Description

Device configuration for block devices.

Definition at line 63 of file virtio_block.h.

14.346.2 Field Documentation

14.346.2.1 blk_size

```
l4_uint32_t l4virtio_block_config_t::blk_size
```

Block size of underlying disk.

Definition at line 74 of file virtio_block.h.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_block.h

14.347 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.

14.347.1 Detailed Description

Header structure of a request for a block device.

Definition at line 52 of file virtio_block.h.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_block.h

14.348 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.

14.348.1 Detailed Description

L4-VIRTIO config header, provided in shared data space.

Definition at line 127 of file virtio.h.

14.348.2 Field Documentation

14.348.2.1 magic

```
l4_uint32_t l4virtio_config_hdr_t::magic
```

magic value (must be 'virt').

Definition at line 129 of file virtio.h.

14.348.2.2 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 166 of file virtio.h.

Referenced by l4virtio_get_feature(), L4virtio::Svr::Dev_config::set_failed(), and L4virtio::Svr::Dev_config::set_status().

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio.h

14.349 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)

14.349.1 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 208 of file virtio.h.

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio.h

Chapter 15

File Documentation

15.1 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:

15.1.1 Detailed Description

APIC for AMD64.

Definition in file apic.h.

15.2 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("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 */
```

15.3 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:

15.3.1 Detailed Description

APIC for X86.

Definition in file apic.h.

15.4 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
```

```
#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 */
```

15.5 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.

15.5.1 Detailed Description

IDT related functions.

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file idt.h.

15.6 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
00030
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
```

15.7 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.

15.7.1 Detailed Description

IDT related functions.

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file idt.h.

15.8 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
00030
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
```

15.9 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:

15.9.1 Detailed Description

Perfomance Monitoring using P5/P6 Measurement Counters.

Define either CPU_PENTIUM or CPU_P6

Definition in file perform.h.

15.10 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) */
```

```

/* 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
```

15.11 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:

15.11.1 Detailed Description

Perfomance Monitoring using P5/P6 Measurement Counters.

Define either CPU_PENTIUM or CPU_P6

Definition in file perform.h.

15.12 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 */
```

```

/* 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"
```

```
00124          : "=b" (*counter0), "=a" (*counter1)
00125          : "1" (0), "c" (0x11)
00126          );
00127 }
00128
00129 static inline void
00130 l4_i586_select_event(int event0, int event1)
00131 {
00132    asm volatile(".byte 0x0f, 0x30\n"
00133     :
00134     :
00135     "a" (event0 + (event1 << 16)),
00136     "d" (0),
00137     "c" (0x11)
00138     );
00139 };
00140
00141 #define P5_RD_MISS          0x003 /* 000011B */
00142 #define P5_WR_MISS          0x008 /* 000100B */
00143 #define P5_RW_MISS          0x029 /* 101001B */
00144 #define P5_EX_MISS          0x00e /* 001110B */
00145
00146 #define P5_D_WBACK          0x006 /* 000110B */
00147
00148 #define P5_RW_TLB           0x002 /* 00010B */
00149 #define P5_EX_TLB           0x00d /* 01101B */
00150
00151 #define P5_A_STALL          0x01f /* 11111B */
00152 #define P5_W_STALL          0x019 /* 11001B */
00153 #define P5_R_STALL          0x01a /* 11010B */
00154 #define P5_X_STALL          0x01b /* 11011B */
00155
00156 #define P5_AGI_STALL        0x01f /* 11111B */
00157
00158 #define P5_PIPLINE_FLUSH    0x015 /* 10101B */
00159
00160 #define P5_NON_CACHE_RD     0x01e /* 11110B */
00161 #define P5_NCACHE_REFS      0x01e /* 11110B */
00162 #define P5_LOCKED_BUS       0x01c /* 11100B */
00163
00164 #define P5_MEM2PIPE         0x009 /* 01001B */
00165 #define P5_BANK_CONF        0x00a /* 01010B */
00166
00167
00168 #define P5_INSTRS_EX        0x016 /* 10110B */
00169 #define P5_INSTRS_EX_V      0x017 /* 10111B */
00170
00171
00172 #define P5_CNT_NOTHING      (0x00 << 6) /* 00B << 6 */
00173 #define P5_CNT_EVENT_PL0    (0x01 << 6) /* 01B << 6 */
00174 #define P5_CNT_EVENT_PL3    (0x02 << 6) /* 10B << 6 */
00175 #define P5_CNT_EVENT        (0x03 << 6) /* 11B << 6 */
00176 #define P5_CNT_CLOCKS_PL0   (0x05 << 6) /* 101B << 6 */
00177 #define P5_CNT_CLOCKS_PL3   (0x06 << 6) /* 110B << 6 */
00178 #define P5_CNT_CLOCKS       (0x07 << 6) /* 111B << 6 */
00179
00180
00181 #else
00182 #if defined CPU_P6
00183 /* PPro/PII/PIII section */
00184
00185 /*-
00186  * Copyright (c) 1997 The President and Fellows of Harvard College.
00187  * All rights reserved.
00188  * Copyright (c) 1997 Aaron B. Brown.
00189  *
00190  * Redistribution and use in source and binary forms, with or without
00191  * modification, are permitted provided that the following conditions
00192  * are met:
00193  * 1. Redistributions of source code must retain the above copyright
00194  *    notice, this list of conditions and the following disclaimer.
00195  * 2. Redistributions in binary form must reproduce the above copyright
00196  *    notice, this list of conditions and the following disclaimer in the
00197  *    documentation and/or other materials provided with the distribution.
00198  * 3. All advertising materials mentioning features or use of this software
00199  *    must display the following acknowledgement:
00200  *      This product includes software developed by Harvard University
00201  *      and its contributors.
00202  * 4. Neither the name of the University nor the names of its contributors
00203  *    may be used to endorse or promote products derived from this software
00204  *    without specific prior written permission.
00205  *
00206  * THIS SOFTWARE IS PROVIDED BY HARVARD AND CONTRIBUTORS ``AS IS'' AND
00207  * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00208  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00209  * ARE DISCLAIMED.  IN NO EVENT SHALL HARVARD UNIVERSITY OR CONTRIBUTORS BE
00210  * 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
```

15.13 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:

Macros

- #define L4_TSC_INIT_AUTO 0

  Automatic init.
- #define L4_TSC_INIT_KERNEL 1

  Initialized by kernel.
- #define L4_TSC_INIT_CALIBRATE 2

  Initialized by user-level.

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_cpu_time_t l4_rdpmc (int nr)

  Return current value of CPU-internal performance measurement counter.
- l4_uint32_t l4_rdpmc_32 (int nr)

  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)

  Calibrate scalers for time stamp calculations.
- l4_uint32_t l4_tsc_init (int constraint, l4_kernel_info_t *kip)

  Initialitze scaler for TSC calicaltions.
- l4_uint32_t l4_get_hz (void)

  Get CPU frequency in Hz.

15.13.1 Detailed Description

time stamp counter related functions

Date

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file rdtsc.h.

15.14 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 */
00038
00039 #define L4_TSC_INIT_AUTO             0
00040 #define L4_TSC_INIT_KERNEL           1
00041 #define L4_TSC_INIT_CALIBRATE        2
00042
00043 extern l4_uint32_t l4_scaler_tsc_to_ns;
00044 extern l4_uint32_t l4_scaler_tsc_to_us;
00045 extern l4_uint32_t l4_scaler_ns_to_tsc;
00046 extern l4_uint32_t l4_scaler_tsc_linux;
00047
00052 L4_INLINE l4_cpu_time_t
00053 l4_rdtsc (void);
00054
00060 L4_INLINE
00061 l4_uint32_t l4_rdtsc_32(void);
00062
00067 L4_INLINE l4_cpu_time_t
00068 l4_rdpmc (int nr);
00069
00075 L4_INLINE
00076 l4_uint32_t l4_rdpmc_32(int nr);
00077
```

```
L4_INLINE l4_uint64_t
l4_tsc_to_ns (l4_cpu_time_t tsc);

L4_INLINE l4_uint64_t
l4_tsc_to_us (l4_cpu_time_t tsc);

L4_INLINE void
l4_tsc_to_s_and_ns (l4_cpu_time_t tsc,
l4_uint32_t *s, l4_uint32_t *ns);

L4_INLINE l4_cpu_time_t
l4_ns_to_tsc (l4_uint64_t ns);

L4_INLINE void
l4_busy_wait_ns (l4_uint64_t ns);

L4_INLINE void
l4_busy_wait_us (l4_uint64_t us);

EXTERN_C_BEGIN

L4_INLINE l4_uint32_t
l4_calibrate_tsc (l4_kernel_info_t *kip);

L4_CV l4_uint32_t
l4_tsc_init (int constraint, l4_kernel_info_t *kip);

L4_CV l4_uint32_t
l4_get_hz (void);

EXTERN_C_END

/* implementaion */

L4_INLINE l4_uint32_t
l4_calibrate_tsc (l4_kernel_info_t *kip)
{
  return l4_tsc_init(L4_TSC_INIT_AUTO, kip);
}

L4_INLINE l4_cpu_time_t
l4_rdtsc (void)
{
    l4_cpu_time_t v;

    __asm__ __volatile__
  (".byte 0x0f, 0x31    \n\t"
   "mov   $0xffffffff, %%rcx      \n\t" /* clears the upper 32 bits! */
   "and   %%rcx,%%rax   \n\t"
   "shlq  $32,%%rdx   \n\t"
   "orq %%rdx,%%rax   \n\t"
  :
  "=a" (v)
  : /* no inputs */
  :"rdx", "rcx"
  );

    return v;
}

L4_INLINE l4_cpu_time_t
l4_rdpmc (int nr)
{
    l4_cpu_time_t v;
    l4_uint64_t dummy;

    __asm__ __volatile__ (
   "rdpmc       \n\t"
   "mov   $0xffffffff, %%rcx      \n\t" /* clears the upper 32 bits! */
   "and   %%rcx,%%rax   \n\t"
   "shlq  $32,%%rdx   \n\t"
   "orq %%rdx,%%rax   \n\t"
  :
  "=a" (v), "=c"(dummy)
  : "c" (nr)
        : "rdx"
  );

    return v;
}

/* the same, but only 32 bit. Useful for smaller differences */
L4_INLINE
l4_uint32_t l4_rdpmc_32(int nr)
{
  l4_uint32_t x;
  l4_uint64_t dummy;
```

```
  __asm__ __volatile__ (
          "rdpmc        \n\t"
   "mov   $0xffffffff, %%rcx      \n\t" /* clears the upper 32 bits! */
   "and   %%rcx,%%rax   \n\t"
       : "=a" (x), "=c"(dummy)
       : "c" (nr)
       : "rdx");

  return x;
}

/* 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)
       :
       : "rdx");

  return x;
}

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)
    ;
}

EXTERN_C_END

#endif /* __l4_rdtsc_h */

```

15.15 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:

Macros

- #define L4_TSC_INIT_AUTO 0

  Automatic init.
- #define L4_TSC_INIT_KERNEL 1

  Initialized by kernel.
- #define L4_TSC_INIT_CALIBRATE 2

  Initialized by user-level.

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_cpu_time_t l4_rdpmc (int nr)

  Return current value of CPU-internal performance measurement counter.
- l4_uint32_t l4_rdpmc_32 (int nr)

  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)

  Calibrate scalers for time stamp calculations.
- l4_uint32_t l4_tsc_init (int constraint, l4_kernel_info_t *kip)

  Initialitze scaler for TSC calicaltions.
- l4_uint32_t l4_get_hz (void)

  Get CPU frequency in Hz.

15.15.1 Detailed Description

time stamp counter related functions

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file rdtsc.h.

15.16 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  */
```

```
#ifndef __l4_rdtsc_h
#define __l4_rdtsc_h

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/kip.h>

EXTERN_C_BEGIN

/* interface */

#define L4_TSC_INIT_AUTO             0
#define L4_TSC_INIT_KERNEL           1
#define L4_TSC_INIT_CALIBRATE        2

extern l4_uint32_t l4_scaler_tsc_to_ns;
extern l4_uint32_t l4_scaler_tsc_to_us;
extern l4_uint32_t l4_scaler_ns_to_tsc;
extern l4_uint32_t l4_scaler_tsc_linux;

L4_INLINE l4_cpu_time_t
l4_rdtsc (void);

L4_INLINE
l4_uint32_t l4_rdtsc_32(void);

L4_INLINE l4_cpu_time_t
l4_rdpmc (int nr);

L4_INLINE
l4_uint32_t l4_rdpmc_32(int nr);

L4_INLINE l4_uint64_t
l4_tsc_to_ns (l4_cpu_time_t tsc);

L4_INLINE l4_uint64_t
l4_tsc_to_us (l4_cpu_time_t tsc);

L4_INLINE void
l4_tsc_to_s_and_ns (l4_cpu_time_t tsc,
    l4_uint32_t *s, l4_uint32_t *ns);

L4_INLINE l4_cpu_time_t
l4_ns_to_tsc (l4_uint64_t ns);

L4_INLINE void
l4_busy_wait_ns (l4_uint64_t ns);

L4_INLINE void
l4_busy_wait_us (l4_uint64_t us);

EXTERN_C_BEGIN

L4_INLINE l4_uint32_t
l4_calibrate_tsc (l4_kernel_info_t *kip);

L4_CV l4_uint32_t
l4_tsc_init (int constraint, l4_kernel_info_t *kip);

L4_CV l4_uint32_t
l4_get_hz (void);

EXTERN_C_END

/* implementaion */

L4_INLINE l4_uint32_t
l4_calibrate_tsc (l4_kernel_info_t *kip)
{
  return l4_tsc_init(L4_TSC_INIT_AUTO, 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_cpu_time_t
l4_rdpmc (int nr)
{
    l4_cpu_time_t v;

    __asm__ __volatile__ (
   "rdpmc       \n\t"
  :
  "=A" (v)
  : "c" (nr)
  );

    return v;
}

/* the same, but only 32 bit. Useful for smaller differences,
   needs less cycles. */
L4_INLINE
l4_uint32_t l4_rdpmc_32(int nr)
{
  l4_uint32_t x;

  __asm__ __volatile__ (
       "rdpmc       \n\t"
       : "=a" (x)
       : "c" (nr)
       : "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)
     ;
}

EXTERN_C_END

#endif /* __l4_rdtsc_h */
```

15.17 amd64/l4/util/spin.h File Reference

Spinning for amd64.

`#include <l4/sys/compiler.h>`
Include dependency graph for spin.h:

15.17.1 Detailed Description

Spinning for amd64.

Definition in file spin.h.

15.18 spin.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Torsten Frenzel <frenzel@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.                       *
```

```
* 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
```

15.19 x86/l4/util/spin.h File Reference

Spinning for x86.

`#include <l4/sys/compiler.h>`

Include dependency graph for spin.h:

15.19.1 Detailed Description

Spinning for x86.

Definition in file spin.h.

15.20 spin.h

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@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 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
```

15.21 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.

15.21.1 Detailed Description

Utilities, amd64 version.

Definition in file util.h.

15.21.2 Function Documentation

15.21.2.1 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/main.c.

Referenced by L4Re::Namespace::query().

Here is the caller graph for this function:

15.21.2.2 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

15.22 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)
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 #ifndef __UTIL_H
00015 #define __UTIL_H
00016
00017 #include <l4/sys/types.h>
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/ipc.h>
00020
00021 EXTERN_C_BEGIN
00022
00029 L4_CV l4_timeout_s l4util_micros2l4to(unsigned int mus)
      L4_NOTHROW;
00030
00034 L4_CV void l4_sleep(int ms) L4_NOTHROW;
00035
00036 /* Suspend thread for a period of \a us microseconds.
00037  * \param us Time in microseconds
00038  * WARNING: This function is mostly bogus since the timer resolution of
00039  *          current L4 implementations is about 1ms! */
00040 L4_CV void l4_usleep(int us) L4_NOTHROW;
00041
00047 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW __attribute__((noreturn));
00048
00049 L4_INLINE void
00050 l4_sleep_forever(void) L4_NOTHROW
00051 {
00052   for (;;)
00053     l4_ipc_sleep(L4_IPC_NEVER);
00054 }
00055
00057 static inline void
00058 l4_touch_ro(const void*addr, unsigned size) L4_NOTHROW
00059 {
00060   const char *bptr, *eptr;
00061
00062   bptr = (const char*)(((l4_addr_t)addr) & L4_PAGEMASK);
00063   eptr = (const char*)(((l4_addr_t)addr+size-1) & L4_PAGEMASK);
00064   for(;bptr<=eptr;bptr+=L4_PAGESIZE){
00065     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

```

15.23 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.

15.23.1 Detailed Description

Utilities for x86.

Definition in file util.h.

15.23.2 Function Documentation

15.23.2.1 l4_sleep()

```
void l4_sleep (
            int ms )
```

Suspend thread for a period of *ms* milliseconds.

Parameters

ms	Time in milliseconds

15.23.2.2 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

15.24 util.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>
 *     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,%%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
```

15.25 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 FS register.
- 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 FS register.
- 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.

15.25.1 Detailed Description

l4f specific fs/gs manipulation

Definition in file segment.h.

15.25.2 Function Documentation

15.25.2.1 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 FS register.

Parameters

thread	Thread to get info from.
base	Base address.
utcb	UTCB of the caller.

Returns

System call error

Definition at line 35 of file segment.h.

15.25.2.2 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 FS register.

Parameters

thread	Thread to get info from.
segr	Segment register to set (one of L4_sys_segment).
base	Base address.
utcb	UTCB of the caller.

Returns

System call error

Definition at line 43 of file segment.h.

15.26 segment.h

```
#include_next <l4/sys/segment.h>

/*
 * (c) 2011 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_AMD64__L4API_L4F__SEGMENT_H__
#define __L4_SYS__ARCH_AMD64__L4API_L4F__SEGMENT_H__

#include <l4/sys/compiler.h>

/*****************************************************************************
 *** Implementation
 *****************************************************************************/

L4_INLINE long
fiasco_amd64_set_fs(l4_cap_idx_t thread, 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)L4_AMD64_SEGMENT_FS << 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_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__ */
```

15.27 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.

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 FS register.
- 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 FS register.
- 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.

15.27.1 Detailed Description

Segment handling.

Definition in file segment.h.

15.27.2 Enumeration Type Documentation

15.27.2.1 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 76 of file segment.h.

15.27.3 Function Documentation

15.27.3.1 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 158 of file segment.h.

15.27.3.2 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 FS register.

Parameters

thread	Thread to get info from.
base	Base address.
utcb	UTCB of the caller.

Returns

System call error

Definition at line 35 of file segment.h.

15.27.3.3 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 FS register.

Parameters

thread	Thread to get info from.
segr	Segment register to set (one of L4_sys_segment).
base	Base address.
utcb	UTCB of the caller.

Returns

System call error

Definition at line 43 of file segment.h.

15.28 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
00043 L4_INLINE long
00044 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00045                unsigned int entry_number_start, l4_utcb_t *utcb);
00046
00060 L4_INLINE long
00061 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00062                unsigned int entry_number_start, l4_utcb_t *utcb);
00063
00070 L4_INLINE unsigned
00071 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread,
      l4_utcb_t *utcb);
00072
00076 enum L4_task_ldt_x86_consts
00077 {
00079   L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00081   L4_TASK_LDT_X86_MAX_ENTRIES
00082     = (L4_UTCB_GENERIC_DATA_SIZE - 2)
00083       / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00084 };
00085
00093 L4_INLINE long
00094 fiasco_amd64_set_fs(l4_cap_idx_t thread,
      l4_umword_t base, l4_utcb_t *utcb);
00095
00096 enum L4_sys_segment
00097 {
00098   L4_AMD64_SEGMENT_FS = 0,
00099   L4_AMD64_SEGMENT_GS = 1
00100 };
00101
00110 L4_INLINE long
00111 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum
      L4_sys_segment segr,
00112                               l4_umword_t base, l4_utcb_t *utcb);
00113
00123 L4_INLINE long
00124 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00125                          unsigned *user_cs, unsigned *user32_cs,
00126                          l4_utcb_t *utcb);
00127
00128 /****************************************************************************
00129  *** Implementation
00130  ****************************************************************************/
00131
```

```
#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__ */
```

15.29 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.

15.29.1 Detailed Description

l4f specific segment manipulation

Definition in file segment.h.

15.30 segment.h

```
00001 #include_next <l4/sys/segment.h>
00002
00008 /*
00009  * (c) 2008-2009 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_X86__L4API_L4F__SEGMENT_H__
00026 #define __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__
00027
00028 #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__ */
```

15.31 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.

15.31.1 Detailed Description

Segment handling.

Definition in file segment.h.

15.31.2 Enumeration Type Documentation

15.31.2.1 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 76 of file segment.h.

15.32 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
```

```
 * 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 size,
               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)),
};

/*****************************************************************************
 *** 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];
}

#endif /* ! __L4_SYS__ARCH_X86__SEGMENT_H__ */
```

15.33 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:

15.33.1 Detailed Description

Port I/O functions.

Definition in file port_io.h.

15.34 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>
```

15.35 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:

15.35.1 Detailed Description

Port I/O functions.

Definition in file port_io.h.

15.36 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>
```

15.37 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.

15.37.1 Detailed Description

port I/O functions

Date

06/2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file port_io.h.

15.37.2 Function Documentation

15.37.2.1 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:

15.38 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
```

15.39 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

15.39.1 Detailed Description

x86 port I/O

Date

06/2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file port_io.h.

15.40 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_H
#define _L4UTIL_PORT_IO_H

/* L4 includes */
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

/*****************************************************************************
 *** Prototypes
 *****************************************************************************/

EXTERN_C_BEGIN
L4_INLINE l4_uint8_t
l4util_in8(l4_uint16_t port);

L4_INLINE l4_uint16_t
l4util_in16(l4_uint16_t port);

L4_INLINE l4_uint32_t
l4util_in32(l4_uint16_t port);

L4_INLINE void
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);
00155
00159 L4_INLINE void
00160 l4util_iodelay(void);
00161
00164 EXTERN_C_END
00165
00166
00167 /*****************************************************************************
00168  *** Implementation
00169  *****************************************************************************/
00170
00171 L4_INLINE l4_uint8_t
00172 l4util_in8(l4_uint16_t port)
00173 {
00174   l4_uint8_t value;
00175   asm volatile ("inb %w1, %b0" : "=a" (value) : "Nd" (port));
00176   return value;
00177 }
00178
00179 L4_INLINE l4_uint16_t
00180 l4util_in16(l4_uint16_t port)
00181 {
00182   l4_uint16_t value;
00183   asm volatile ("inw %w1, %w0" : "=a" (value) : "Nd" (port));
00184   return value;
00185 }
00186
00187 L4_INLINE l4_uint32_t
00188 l4util_in32(l4_uint16_t port)
00189 {
00190   l4_uint32_t value;
00191   asm volatile ("inl %w1, %0" : "=a" (value) : "Nd" (port));
00192   return value;
00193 }
00194
00195 L4_INLINE void
00196 l4util_ins8(l4_uint16_t port, l4_umword_t addr,
      l4_umword_t count)
00197 {
00198   l4_umword_t dummy1, dummy2;
00199   asm volatile ("rep insb" : "=D"(dummy1), "=c"(dummy2)
00200          : "d" (port), "D" (addr), "c"(count)
00201          : "memory");
00202 }
00203
00204 L4_INLINE void
00205 l4util_ins16(l4_uint16_t port, l4_umword_t addr,
      l4_umword_t count)
00206 {
00207   l4_umword_t dummy1, dummy2;
00208   asm volatile ("rep insw" : "=D"(dummy1), "=c"(dummy2)
00209          : "d" (port), "D" (addr), "c"(count)
00210          : "memory");
00211 }
00212
00213 L4_INLINE void
00214 l4util_ins32(l4_uint16_t port, l4_umword_t addr,
```

```
      l4_umword_t count)
00215 {
00216   l4_umword_t dummy1, dummy2;
00217   asm volatile ("rep insl" : "=D"(dummy1), "=c"(dummy2)
00218           : "d" (port), "D" (addr), "c"(count)
00219           : "memory");
00220 }
00221
00222 L4_INLINE void
00223 l4util_out8(l4_uint8_t value, l4_uint16_t port)
00224 {
00225   asm volatile ("outb %b0, %w1" : : "a" (value), "Nd" (port));
00226 }
00227
00228 L4_INLINE void
00229 l4util_out16(l4_uint16_t value, l4_uint16_t port)
00230 {
00231   asm volatile ("outw %w0, %w1" : : "a" (value), "Nd" (port));
00232 }
00233
00234 L4_INLINE void
00235 l4util_out32(l4_uint32_t value, l4_uint16_t port)
00236 {
00237   asm volatile ("outl %0, %w1" : : "a" (value), "Nd" (port));
00238 }
00239
00240 L4_INLINE void
00241 l4util_outs8(l4_uint16_t port, l4_umword_t addr,
      l4_umword_t count)
00242 {
00243   l4_umword_t dummy1, dummy2;
00244   asm volatile ("rep outsb" : "=S"(dummy1), "=c"(dummy2)
00245           : "d" (port), "S" (addr), "c"(count)
00246           : "memory");
00247 }
00248
00249 L4_INLINE void
00250 l4util_outs16(l4_uint16_t port, l4_umword_t addr,
      l4_umword_t count)
00251 {
00252   l4_umword_t dummy1, dummy2;
00253   asm volatile ("rep outsw" : "=S"(dummy1), "=c"(dummy2)
00254           : "d" (port), "S" (addr), "c"(count)
00255           : "memory");
00256 }
00257
00258 L4_INLINE void
00259 l4util_outs32(l4_uint16_t port, l4_umword_t addr,
      l4_umword_t count)
00260 {
00261   l4_umword_t dummy1, dummy2;
00262   asm volatile ("rep outsl" : "=S"(dummy1), "=c"(dummy2)
00263           : "d" (port), "S" (addr), "c"(count)
00264           : "memory");
00265 }
00266
00267 L4_INLINE void
00268 l4util_iodelay(void)
00269 {
00270   asm volatile ("outb %al,$0x80");
00271 }
00272
00273 #endif
```

15.41 amd64/l4f/l4/util/setjmp.h File Reference

Inter-thread setjmp/longjmp.

```
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for setjmp.h:

Functions

- int l4_thread_setjmp (l4_thread_jmp_buf env)

  inter-thread setjmp
- void l4_thread_longjmp (l4_threadid_t thread, l4_thread_jmp_buf env, int val)

  inter-thread longjmp

15.41.1 Detailed Description

Inter-thread setjmp/longjmp.

Date

12/21/2005

Author

Jork Loeser `jork.loeser@inf.tu-dresden.de`

Definition in file setjmp.h.

15.41.2 Function Documentation

15.41.2.1 l4_thread_longjmp()

```
void l4_thread_longjmp (
            l4_threadid_t thread,
            l4_thread_jmp_buf env,
            int val )
```

inter-thread longjmp

This function sets `thread` to the location obtained by its former l4_thread_setjump on `env`.

Parameters

thread	thread to apply the longjmp to
env	jump buffer
val	0: setjmp returns with 1
val	!0: return value of setjmp

See also

longjmp(3)

Note

In contrast to longjmp(3), this function returns.

15.41.2.2 l4_thread_setjmp()

```
int l4_thread_setjmp (
            l4_thread_jmp_buf env )
```

inter-thread setjmp

Use this function to prepare a longjmp from another thread for this thread.

Parameters

env	jump buffer

Return values

0	returned directly
!0	returned from longjmp

See also

setjmp(3)

15.42 setjmp.h

```
00001
00009 /*
00010  * (c) 2004-2009 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 #ifndef __UTIL_INCLUDE_ARCH_AMD64_L4API_L4F_SETJMP_H_
```

```
#define __UTIL_INCLUDE_ARCH_AMd64_L4API_L4F_SETJMP_H_
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

EXTERN_C_BEGIN

typedef struct{
    l4_umword_t r8;     /* 0x00 */
    l4_umword_t r9;     /* 0x08 */
    l4_umword_t r10;    /* 0x10 */
    l4_umword_t r11;    /* 0x18 */
    l4_umword_t r12;    /* 0x20 */
    l4_umword_t r13;    /* 0x28 */
    l4_umword_t r14;    /* 0x30 */
    l4_umword_t r15;    /* 0x38 */
    l4_umword_t rbx;    /* 0x40 */
    l4_umword_t rsi;    /* 0x48 */
    l4_umword_t rbp;    /* 0x50 */
    l4_umword_t rsp;    /* 0x58 */
    l4_umword_t rip;    /* 0x60 */
    l4_umword_t rip_caller;  /* 0x68 */
    l4_umword_t rflags;     /* 0x70 */
    l4_umword_t stack[40];
} l4_thread_jmp_buf_s;
typedef int l4_thread_jmp_buf[sizeof(l4_thread_jmp_buf_s)/sizeof(l4_umword_t)];

typedef union{
    l4_thread_jmp_buf_s s;
    l4_thread_jmp_buf raw;
} l4_thread_jmp_buf_u;

L4_CV int l4_thread_setjmp(l4_thread_jmp_buf env);

L4_CV void l4_thread_longjmp(l4_threadid_t thread, l4_thread_jmp_buf env, int val);

EXTERN_C_END

#endif
```

15.43 x86/l4f/l4/util/setjmp.h File Reference

Inter-thread setjmp/longjmp.

```
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for setjmp.h:

Functions

- int l4_thread_setjmp (l4_thread_jmp_buf env)

  inter-thread setjmp
- void l4_thread_longjmp (l4_threadid_t thread, l4_thread_jmp_buf env, int val)

  inter-thread longjmp

15.43.1 Detailed Description

Inter-thread setjmp/longjmp.

Date

11/26/2004

Author

Jork Loeser `jork.loeser@inf.tu-dresden.de`

Definition in file setjmp.h.

15.43.2 Function Documentation

15.43.2.1 l4_thread_longjmp()

```
void l4_thread_longjmp (
            l4_threadid_t thread,
            l4_thread_jmp_buf env,
            int val )
```

inter-thread longjmp

This function sets `thread` to the location obtained by its former l4_thread_setjump on `env`.

Parameters

thread	thread to apply the longjmp to
env	jump buffer
val	0: setjmp returns with 1
val	!0: return value of setjmp

See also

longjmp(3)

Note

In contrast to longjmp(3), this function returns.

15.43.2.2 l4_thread_setjmp()

```
int l4_thread_setjmp (
            l4_thread_jmp_buf env )
```

inter-thread setjmp

Use this function to prepare a longjmp from another thread for this thread.

Parameters

env	jump buffer

Return values

0	returned directly
!0	returned from longjmp

See also

setjmp(3)

15.44 setjmp.h

```
00001
00009 /*
00010  * (c) 2004-2009 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 #ifndef __UTIL_INCLUDE_ARCH_X86_L4API_L4F_SETJMP_H_
00017 #define __UTIL_INCLUDE_ARCH_X86_L4API_L4F_SETJMP_H_
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020
00021 EXTERN_C_BEGIN
00022
00023 typedef struct{
00024     l4_umword_t ebx;   /* 0 */
00025     l4_umword_t esi;   /* 4 */
00026     l4_umword_t edi;   /* 8 */
00027     l4_umword_t ebp;   /* 12 */
00028     l4_umword_t esp;   /* 16 */
00029     l4_umword_t eip;   /* 20 */
00030     l4_umword_t eip_caller;  /* 24 */
00031     l4_umword_t eflags;    /* 28 */
00032     l4_umword_t stack[40];
00033 } l4_thread_jmp_buf_s;
00034 typedef int l4_thread_jmp_buf[sizeof(l4_thread_jmp_buf_s)/sizeof(l4_umword_t)];
00035
00036 typedef union{
00037     l4_thread_jmp_buf_s s;
00038     l4_thread_jmp_buf raw;
```

```
} l4_thread_jmp_buf_u;

extern int l4_thread_setjmp(l4_thread_jmp_buf env);

void l4_thread_longjmp(l4_threadid_t thread, l4_thread_jmp_buf env, int val);

EXTERN_C_END

#endif
```

15.45 arm/l4/sys/linkage.h File Reference

Linkage.

Macros

- #define L4_CV

  Define calling convention.

15.45.1 Detailed Description

Linkage.

Definition in file linkage.h.

15.46 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__ */
```

15.47 amd64/l4/sys/linkage.h File Reference

Linkage.

Macros

- #define L4_CV

  Define calling convention.

15.47.1 Detailed Description

Linkage.

Definition in file linkage.h.

15.48 linkage.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  *               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 #ifndef __L4__SYS__ARCH_AMD64__LINKAGE_H__
00026 #define __L4__SYS__ARCH_AMD64__LINKAGE_H__
00027
00028 #ifdef __ASSEMBLY__
00029
00030 #ifndef ENTRY
00031 #define ENTRY(name) \
00032   .globl name; \
00033   .p2align(2); \
00034   name:
00035
00036 #endif /* __ASSEMBLY__ */
00037 #endif /* ! ENTRY */
00038
00039 #define L4_FASTCALL(x)      x
00040 #define l4_fastcall
00041
00047 #define L4_CV
00048
00049 #endif /* ! __L4__SYS__ARCH_AMD64__LINKAGE_H__ */
```

15.49 x86/l4/sys/linkage.h File Reference

Linkage.

This graph shows which files directly or indirectly include this file:

Macros

- #define L4_CV __attribute__((regparm(0)))

  Define calling convention.

15.49.1 Detailed Description

Linkage.

Definition in file linkage.h.

15.50 linkage.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  *               Frank Mehnert <fm3@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_X86__LINKAGE_H__
00026 #define __L4__SYS__ARCH_X86__LINKAGE_H__
00027
00028 #ifdef __ASSEMBLY__
00029
00030 #ifndef ENTRY
00031 #define ENTRY(name) \
00032   .globl name; \
00033   .p2align(2); \
00034   name:
00035
00036 #endif /* ! ENTRY */
00037 #endif /* __ASSEMBLY__ */
00038
00039 #define L4_FASTCALL(x)  x __attribute__((regparm(3)))
00040 #define l4_fastcall __attribute__((regparm(3)))
00041
00047 #define L4_CV   __attribute__((regparm(0)))
00048
00049 #endif /* ! __L4__SYS__ARCH_X86__LINKAGE_H__ */
```

15.51 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_WID↩TH_4BYTE = 2 }

  Memory access width definitions.

Functions

- unsigned long l4_mem_read (unsigned long virtaddress, unsigned width)

  Read memory from kernel privilege level.
- void l4_mem_write (unsigned long virtaddress, unsigned width, unsigned long value)

  Write 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.

15.51.1 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.

15.52 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;
```

```
}

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__ */
```

15.53 arm/l4/sys/vm.h File Reference

ARM virtualization interface.

Data Structures

- struct l4_vm_tz_state

  state structure for TrustZone VMs

15.53.1 Detailed Description

ARM virtualization interface.

Definition in file vm.h.

15.54 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;
}
```

15.55 arm/l4/util/bitops_arch.h File Reference

ARM specific implementation of bitops functions.

15.55.1 Detailed Description

ARM specific implementation of bitops functions.

Definition in file bitops_arch.h.

15.56 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__ */
```

15.57 amd64/l4/util/bitops_arch.h File Reference

amd64 bit manipulation functions

15.57.1 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.

15.58 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__
```

```
    (
     "lock; bts  %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; 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__ */
```

15.59 x86/l4/util/bitops_arch.h File Reference

x86 bit manipulation functions

This graph shows which files directly or indirectly include this file:

15.59.1 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.

15.60 bitops_arch.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__ARCH_X86__BITOPS_ARCH_H__
00019 #define __L4UTIL__INCLUDE__ARCH_X86__BITOPS_ARCH_H__
00020
00021 /*****************************************************************************
00022  *** Implementation
00023  *****************************************************************************/
00024
00025 EXTERN_C_BEGIN
00026
00027 /* set bit */
00028 #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__ */
```

15.61 arm/l4/util/cpu.h File Reference

CPU related functions.

15.61.1 Detailed Description

CPU related functions.

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file cpu.h.

15.62 cpu.h

```
00001
00008 /*
00009  * (c) 2004-2009 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_UTIL__ARCH_ARM__CPU_H__
00017 #define __L4_UTIL__ARCH_ARM__CPU_H__
00018
00019 /* Nothing yet */
00020
00021 #endif /* __L4_UTIL__ARCH_ARM__CPU_H__ */
```

15.63 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.

15.63.1 Detailed Description

CPU related functions.

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file cpu.h.

15.64 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
00027
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 }
```

```
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
```

15.65 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.

15.65.1 Detailed Description

CPU related functions.

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file cpu.h.

15.66 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
00027
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   unsigned long eax;
00069
00070   asm volatile("pushl %%ebx             \t\n"
00071                "pushfl      \t\n"
00072                "popl %%eax    \t\n" /* get eflags */
00073                "movl %%eax, %%ebx \t\n" /* save it */
00074                "xorl $0x200000, %%eax \t\n" /* toggle ID bit */
00075                "pushl %%eax   \t\n"
00076                "popfl     \t\n" /* set again */
00077                "pushfl      \t\n"
00078                "popl %%eax    \t\n" /* get it again */
00079                "xorl %%ebx, %%eax \t\n"
00080                "pushl %%ebx   \t\n"
00081                "popfl     \t\n" /* restore saved flags */
00082                "popl %%ebx              \t\n"
00083                : "=a" (eax)
00084                : /* no input */
00085                );
00086
00087   return eax & 0x200000;
00088 }
00089
00090 L4_INLINE void
00091 l4util_cpu_cpuid(unsigned long mode,
00092                 unsigned long *eax, unsigned long *ebx,
00093                 unsigned long *ecx, unsigned long *edx)
00094 {
```

```
  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

```

15.67 arm/l4/util/l4_macros.h File Reference

Main function.

15.67.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file l4_macros.h.

15.68 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 */
```

15.69 amd64/l4/util/l4_macros.h File Reference

Main function.

15.69.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file l4_macros.h.

15.70 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_AMD64__L4_MACROS_H
#define _L4UTIL__ARCH_AMD64__L4_MACROS_H

#include_next <l4/util/l4_macros.h>

#ifndef l4_addr_fmt
#  define l4_addr_fmt   "%016lx"
#endif

#endif /* !_L4UTIL__ARCH_AMD64__L4_MACROS_H */
```

15.71 l4/util/l4_macros.h File Reference

Some useful generic macros, L4f version.

15.71.1 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.

15.72 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

```

15.73 x86/l4/util/l4_macros.h File Reference

Main function.

15.73.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file l4_macros.h.

15.74 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
00017 #ifndef _L4UTIL__ARCH_X86__L4_MACROS_H
00018 #define _L4UTIL__ARCH_X86__L4_MACROS_H
00019
00020 #include_next <l4/util/l4_macros.h>
00021
00022 #ifndef l4_addr_fmt
00023 #  define l4_addr_fmt   "%08lx"
00024 #endif
00025
00026 #endif /* !_L4UTIL__ARCH_X86__L4_MACROS_H */
```

15.75 arm/l4/util/mbi_argv.h File Reference

Multiboot.

`#include <x86/l4/util/mbi_argv.h>`

Include dependency graph for mbi_argv.h:

15.75.1 Detailed Description

Multiboot.

Definition in file mbi_argv.h.

15.76 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>
```

15.77 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:

15.77.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file mbi_argv.h.

15.78 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
```

15.79 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:

15.79.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file mbi_argv.h.

15.80 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
```

15.81 arm/l4/util/stack_impl.h File Reference

Stack utilities, arm version.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for stack_impl.h:

Functions

- l4_addr_t l4util_stack_get_sp (void)
  Get current stack pointer.

15.81.1 Detailed Description

Stack utilities, arm version.

Definition in file stack_impl.h.

15.81.2 Function Documentation

15.81.2.1 l4util_stack_get_sp()

```
l4_addr_t l4util_stack_get_sp (
            void  )  [inline]
```

Get current stack pointer.

Returns

stack pointer.

Definition at line 23 of file stack_impl.h.

15.82 stack_impl.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__ARCH_ARM__STACK_IMPL_H__
00013 #define __L4UTIL__INCLUDE__ARCH_ARM__STACK_IMPL_H__
00014
00015 #include <l4/sys/compiler.h>
00016
00017 EXTERN_C_BEGIN
00018
00019 #ifndef _L4UTIL_STACK_H
00020 #error Do not include stack_impl.h directly, use stack.h instead
00021 #endif
00022
00023 L4_INLINE l4_addr_t l4util_stack_get_sp(void)
00024 {
00025   register l4_addr_t sp asm ("sp");
00026   return sp;
00027 }
00028
00029 EXTERN_C_END
00030
00031 #endif /* ! __L4UTIL__INCLUDE__ARCH_ARM__STACK_IMPL_H__ */
```

15.83 amd64/l4/util/stack_impl.h File Reference

Stack utilities for amd64.

Functions

- l4_addr_t l4util_stack_get_sp (void)
  Get current stack pointer.

15.83.1 Detailed Description

Stack utilities for amd64.

Definition in file stack_impl.h.

15.83.2 Function Documentation

15.83.2.1 l4util_stack_get_sp()

```
l4_addr_t l4util_stack_get_sp (
            void  )  [inline]
```

Get current stack pointer.

Returns

stack pointer.

Definition at line 22 of file stack_impl.h.

References EXTERN_C_END.

15.84 stack_impl.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Torsten Frenzel <frenzel@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__INCLUDE__ARCH_AMD64__STACK_IMPL_H__
00014 #define __L4UTIL__INCLUDE__ARCH_AMD64__STACK_IMPL_H__
00015
00016 EXTERN_C_BEGIN
00017
00018 #ifndef _L4UTIL_STACK_H
00019 #error Do not include stack_impl.h directly, use stack.h instead
00020 #endif
00021
00022 L4_INLINE l4_addr_t l4util_stack_get_sp(void)
00023 {
00024   l4_addr_t rsp;
00025
00026   asm("movq   %%rsp, %0\n\t" : "=r" (rsp) : );
00027   return rsp;
00028 }
00029
00030 EXTERN_C_END
00031
00032 #endif /* ! __L4UTIL__INCLUDE__ARCH_AMD64__STACK_IMPL_H__ */
```

15.85 x86/l4/util/stack_impl.h File Reference

Stack utilities for x86.

This graph shows which files directly or indirectly include this file:

Functions

- l4_addr_t l4util_stack_get_sp (void)

  Get current stack pointer.

15.85.1 Detailed Description

Stack utilities for x86.

Definition in file stack_impl.h.

15.85.2 Function Documentation

15.85.2.1 l4util_stack_get_sp()

```
l4_addr_t l4util_stack_get_sp (
            void  )  [inline]
```

Get current stack pointer.

Returns

stack pointer.

Definition at line 21 of file stack_impl.h.

References EXTERN_C_END.

15.86 stack_impl.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__ARCH_X86__STACK_IMPL_H__
00013 #define __L4UTIL__INCLUDE__ARCH_X86__STACK_IMPL_H__
00014
00015 #ifndef _L4UTIL_STACK_H
00016 #error Do not include stack_impl.h directly, use stack.h instead
00017 #endif
00018
00019 EXTERN_C_BEGIN
00020
00021 L4_INLINE l4_addr_t l4util_stack_get_sp(void)
00022 {
00023   l4_addr_t esp;
00024
00025   asm("movl   %%esp, %0\n\t" : "=r" (esp) : );
00026   return esp;
00027 }
00028
00029 EXTERN_C_END
00030
00031 #endif /* ! __L4UTIL__INCLUDE__ARCH_ARM__STACK_IMPL_H__ */
```

15.87 arm/l4f/l4/sys/syscall_defs.h File Reference

Syscall entry definitions.

This graph shows which files directly or indirectly include this file:

15.87.1 Detailed Description

Syscall entry definitions.

Definition in file syscall_defs.h.

15.88 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 #define L4_SYSCALL_DEBUGGER   (-0x0000000C-L4_SYSCALL_MAGIC_OFFSET)
00031
00032 #endif /* __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__ */
```

15.89 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_search_iomem_region (l4_addr_t phys, l4_addr_t size, l4_addr_t *rstart, l4_addr_t *rsize)

  Search for a 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.

15.89.1 Function Documentation

15.89.1.1 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 268 of file io.h.

References EXTERN_C_END.

15.89.1.2 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

Return values

dev	Device information, may be NULL
reshandle	Resource handle.

Returns

0 on success, error code otherwise

15.89.1.3 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.

15.89.1.4 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.

15.90 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
```

```
EXTERN_C_BEGIN

L4_CV long L4_EXPORT
l4io_request_irq(int irqnum, l4_cap_idx_t irqcap);

L4_CV l4_cap_idx_t L4_EXPORT
l4io_request_icu(void);

L4_CV long L4_EXPORT
l4io_release_irq(int irqnum, l4_cap_idx_t irq_cap);

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_search_iomem_region(l4_addr_t phys, l4_addr_t size,
                         l4_addr_t *rstart, l4_addr_t *rsize);

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
```

15.91 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_PRO↩TO_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,
  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.

15.91.1 Detailed Description

Common L4 ABI Data Types.

Definition in file types.h.

15.91.2 Function Documentation

15.91.2.1 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 459 of file types.h.

15.92 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  *
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/l4int.h>
00031 #include <l4/sys/compiler.h>
00032 #include <l4/sys/consts.h>
00033
00034
00049 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 throw() { return raw >> 16; }
  void label(long v) throw() { raw = (raw & 0x0ffff) | (v << 16); }
  unsigned words() const throw() { return raw & 0x3f; }
  unsigned items() const throw() { return (raw >> 6) & 0x3f; }
  unsigned flags() const throw() { return raw & 0xf000; }
  bool is_page_fault() const throw() { return label() ==
L4_PROTO_PAGE_FAULT; }
  bool is_preemption() const throw() { return label() ==
L4_PROTO_PREEMPTION; }
  bool is_sys_exception() const throw() { return label() ==
L4_PROTO_SYS_EXCEPTION; }
  bool is_exception() const throw() { return label() ==
L4_PROTO_EXCEPTION; }
  bool is_sigma0() const throw() { return label() ==
L4_PROTO_SIGMA0; }
  bool is_io_page_fault() const throw() { return label() ==
L4_PROTO_IO_PAGE_FAULT; }
  unsigned has_error() const throw() { 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;
00220
00229 L4_INLINE unsigned l4_msgtag_words(l4_msgtag_t t)
      L4_NOTHROW;
00230
00239 L4_INLINE unsigned l4_msgtag_items(l4_msgtag_t t)
      L4_NOTHROW;
00240
00251 L4_INLINE unsigned l4_msgtag_flags(l4_msgtag_t t)
      L4_NOTHROW;
00252
00265 L4_INLINE unsigned l4_msgtag_has_error(l4_msgtag_t t)
      L4_NOTHROW;
00266
00275 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t)
      L4_NOTHROW;
00276
00284 L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t)
      L4_NOTHROW;
00285
00294 L4_INLINE unsigned l4_msgtag_is_sys_exception(
      l4_msgtag_t t) L4_NOTHROW;
00295
00304 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t)
      L4_NOTHROW;
00305
00314 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t)
      L4_NOTHROW;
00315
00324 L4_INLINE unsigned l4_msgtag_is_io_page_fault(
      l4_msgtag_t t) L4_NOTHROW;
00325
00342 typedef unsigned long l4_cap_idx_t;
00343
00353 L4_INLINE unsigned l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW;
00354
00364 L4_INLINE unsigned l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW;
00365
00376 L4_INLINE unsigned l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2)
      L4_NOTHROW;
00377
00386 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c)
      L4_NOTHROW;
00387
00388 /* ********************************************************************** */
00389 /* Implementation */
00390
00391 L4_INLINE unsigned
00392 l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW
00393 { return c & L4_INVALID_CAP_BIT; }
00394
00395 L4_INLINE unsigned
00396 l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW
00397 { return !(c & L4_INVALID_CAP_BIT); }
00398
00399 L4_INLINE unsigned
00400 l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW
00401 { return (c1 >> L4_CAP_SHIFT) == (c2 >> L4_CAP_SHIFT); }
00402
00403
00407 L4_INLINE
00408 l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
00409                       unsigned flags) L4_NOTHROW
00410 {
00411   return (l4_msgtag_t){(label << 16) | (l4_mword_t)(words & 0x3f)
00412                        | (l4_mword_t)((items & 0x3f) << 6)
00413                        | (l4_mword_t)(flags & 0xf000)};
00414 }
00415
00416
00417
00418 L4_INLINE
00419 long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW
00420 { return t.raw >> 16; }
00421
00422 L4_INLINE
00423 unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW
00424 { return t.raw & 0x3f; }
00425
00426 L4_INLINE
00427 unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW
00428 { return (t.raw >> 6) & 0x3f; }
00429
00430 L4_INLINE
00431 unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW
00432 { return t.raw & 0xf000; }
00433
```

```
L4_INLINE
unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW
{ return t.raw & L4_MSGTAG_ERROR; }



L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW
{ return l4_msgtag_label(t) == L4_PROTO_PAGE_FAULT; }

L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t) L4_NOTHROW
{ return l4_msgtag_label(t) == L4_PROTO_PREEMPTION; }

L4_INLINE unsigned l4_msgtag_is_sys_exception(
    l4_msgtag_t t) L4_NOTHROW
{ return l4_msgtag_label(t) == L4_PROTO_SYS_EXCEPTION; }

L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW
{ return l4_msgtag_label(t) == L4_PROTO_EXCEPTION; }

L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW
{ return l4_msgtag_label(t) == L4_PROTO_SIGMA0; }

L4_INLINE unsigned l4_msgtag_is_io_page_fault(
    l4_msgtag_t t) L4_NOTHROW
{ return l4_msgtag_label(t) == L4_PROTO_IO_PAGE_FAULT; }

L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW
{ return c + L4_CAP_OFFSET; }

#include <l4/sys/__l4_fpage.h>
#include <l4/sys/__timeout.h>
```

15.93 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.

15.93.1 Detailed Description

AVL map.

Definition in file avl_map.

15.94 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
```

```
 * 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/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/pair>
#include <l4/cxx/avl_set>

namespace cxx {
namespace Bits {

template<typename KEY_TYPE>
struct Avl_map_get_key
{
  typedef KEY_TYPE Key_type;
  template<typename NODE>
  static Key_type const &key_of(NODE const *n)
  { return n->item.first; }
};
}

template< typename KEY_TYPE, typename DATA_TYPE,
  template<typename A> class COMPARE = Lt_functor,
  template<typename B> class ALLOC = New_allocator >
class Avl_map :
  public Bits::Base_avl_set<Pair<KEY_TYPE, DATA_TYPE>,
                            COMPARE<KEY_TYPE>, ALLOC,
                            Bits::Avl_map_get_key<KEY_TYPE> >
{
private:
  typedef Pair<KEY_TYPE, DATA_TYPE> Local_item_type;
  typedef Bits::Base_avl_set<Local_item_type, COMPARE<KEY_TYPE>, ALLOC,
                             Bits::Avl_map_get_key<KEY_TYPE> > Base_type;

public:
  typedef COMPARE<KEY_TYPE> Key_compare;
  typedef KEY_TYPE Key_type;
  typedef DATA_TYPE Data_type;
  typedef typename Base_type::Node Node;
  typedef typename Base_type::Node_allocator Node_allocator;

  typedef typename Base_type::Iterator Iterator;
  typedef typename Base_type::Iterator iterator;
  typedef typename Base_type::Const_iterator Const_iterator;
  typedef typename Base_type::Const_iterator const_iterator;
  typedef typename Base_type::Rev_iterator Rev_iterator;
  typedef typename Base_type::Rev_iterator reverse_iterator;
  typedef typename Base_type::Const_rev_iterator Const_rev_iterator;
  typedef typename Base_type::Const_rev_iterator const_reverse_iterator;

  Avl_map(Node_allocator const &alloc = Node_allocator())
    : Base_type(alloc)
  {}

  cxx::Pair<Iterator, int> insert(Key_type const &key, Data_type const &data)
  { return Base_type::insert(Pair<Key_type, Data_type>(key, data)); }

  Data_type const &operator [] (Key_type const &key) const
  { return this->find_node(key)->second; }

  Data_type &operator [] (Key_type const &key)
  {
    Node n = this->find_node(key);
    if (n)
      return const_cast<Data_type&>(n->second);
    else
      return insert(key, Data_type()).first->second;
  }
};

}

```

15.95 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.

15.95.1 Detailed Description

AVL set.

Definition in file avl_set.

15.96 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
00049   using Base::_n;
00050   using Base::_r;
```

```
  using Base::inc;

public:
  Avl_set_iter() {}

  Avl_set_iter(Node const *t) : Base(t) {}

  Avl_set_iter(Base const &o) : Base(o) {}

  Avl_set_iter(Avl_set_iter<Node, typename Type_traits<Key>::Non_const_type, Node_op> const &o)
  : Base(o) {}

  Key &operator * () const { return const_cast<Node*>(_n)->item; }
  Key *operator -> () const { return &const_cast<Node*>(_n)->item; }
  Avl_set_iter &operator ++ () { inc(); return *this; }
  Avl_set_iter &operator ++ (int)
  { Avl_set_iter tmp = *this; inc(); return tmp; }

};

template<typename KEY_TYPE>
struct Avl_set_get_key
{
  typedef KEY_TYPE Key_type;
  template<typename NODE>
  static Key_type const &key_of(NODE const *n)
  { return n->item; }
};


template< typename ITEM_TYPE, class COMPARE,
          template<typename A> class ALLOC,
          typename GET_KEY>
class Base_avl_set
{
public:
  enum
  {
    E_noent =  2,
    E_exist = 17,
    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() : Avl_tree_node(), item() {}

    _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) {}

    Node &operator = (Node const &o) { _n = o._n; return *this; }

    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); }
};


//-----------------------------------------------------------------------------
/* 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
```

15.97 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.

15.97.1 Detailed Description

AVL tree.

Definition in file avl_tree.

15.98 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;
00056   Bal _balance;
00058
00059 protected:
00061   Avl_tree_node() {}
00062
00063 private:
00065   Avl_tree_node(Avl_tree_node const &o);
00066
00068   void operator = (Avl_tree_node const &o)
```

```
  {
    Bits::Bst_node::operator = (o);
    _balance = o._balance;
  }

  explicit Avl_tree_node(bool) : Bits::Bst_node(true), _balance(Dir::N) {}

  static Bits::Bst_node *rotate2(Bst_node **t, Bal idir, Bal pre);

  bool balanced() const { return _balance == Bal::N; }

  Bal balance() const { return _balance; }

  void balance(Bal b) { _balance = b; }
};


template< typename Node, typename Get_key,
          typename Compare = Lt_functor<typename Get_key::Key_type> >
class Avl_tree : public Bits::Bst<Node, Get_key, Compare>
{
private:
  typedef Bits::Bst<Node, Get_key, Compare> Bst;

  using Bst::_head;

  using Bst::k;

  typedef typename Avl_tree_node::Bal Bal;
  typedef typename Avl_tree_node::Bal Dir;

  Avl_tree(Avl_tree const &o);

  void operator = (Avl_tree const &o);

public:
  typedef typename Bst::Key_type Key_type;
  typedef typename Bst::Key_param_type Key_param_type;

  // Grab iterator types from Bst
  typedef typename Bst::Iterator Iterator;
  typedef typename Bst::Const_iterator Const_iterator;
  typedef typename Bst::Rev_iterator Rev_iterator;
  typedef typename Bst::Const_rev_iterator Const_rev_iterator;

  Pair<Node *, bool> insert(Node *new_node);

  Node *remove(Key_param_type key);
  Node *erase(Key_param_type key) { return remove(key); }

  Avl_tree() : Bst() {}
  ~Avl_tree() noexcept
  {
    this->remove_all([](Node *){});
  }

#ifdef __DEBUG_L4_AVL
  bool rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char pfx);
  bool rec_dump(bool print)
  {
    int dp=0;
    return rec_dump(static_cast<Avl_tree_node *>(_head), 0, &dp, print, '+');
  }
#endif
};


//-----------------------------------------------------------------------------
/* IMPLEMENTATION: Bits::__Bst_iter_b */


inline
Bits::Bst_node *
Avl_tree_node::rotate2(Bst_node **t, Bal idir, Bal pre)
{
  typedef Bits::Bst_node N;
  typedef Avl_tree_node A;
  N *tmp[2] = { *t, N::next(*t, idir) };
  *t = N::next(tmp[1], !idir);
  A *n = static_cast<A*>(*t);

  N::next(tmp[0], idir, N::next(n, !idir));
  N::next(tmp[1], !idir, N::next(n, idir));
  N::next(n, !idir, tmp[0]);
  N::next(n, idir, tmp[1]);
```

```

  n->balance(Bal::N);

  if (pre == Bal::N)
    {
      static_cast<A*>(tmp[0])->balance(Bal::N);
      static_cast<A*>(tmp[1])->balance(Bal::N);
      return 0;
    }

  static_cast<A*>(tmp[pre != idir])->balance(!pre);
  static_cast<A*>(tmp[pre == idir])->balance(Bal::N);

  return N::next(tmp[pre == idir], !pre);
}

//-----------------------------------------------------------------------------
/* Implementation of AVL Tree */

/* Insert new _Node. */
template< typename Node, typename Get_key, class Compare>
Pair<Node *, bool>
Avl_tree<Node, Get_key, Compare>::insert(Node *new_node)
{
  typedef Avl_tree_node A;
  typedef Bits::Bst_node N;
  N **t = &_head; /* search variable */
  N **s = &_head; /* node where rebalancing may occur */
  Key_param_type new_key = Get_key::key_of(new_node);

  // search insertion point
  for (N *p; (p = *t);)
    {
      Dir b = this->dir(new_key, p);
      if (b == Dir::N)
  return pair(static_cast<Node*>(p), false);

      if (!static_cast<A const *>(p)->balanced())
  s = t;

      t = N::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 = N::next(n, b);
  }
      else if (b == Bal(this->greater(new_key, N::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 = N::next(*s, b);
  }
      else
  {
    // need a double rotation
    n = N::next(N::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 = N::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 = N::next_p(n, dir))
    {
      dir = Dir(this->greater(key, n));
      if (dir == Dir::L && !this->greater(k(n), key))
  /* found node */
  t = q;

      if (!N::next(n, dir))
  break;

      A const *a = static_cast<A const *>(n);
      if (a->balanced() || (a->balance() == !dir && N::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 = N::next_p(n, dir))
    {
      dir = Dir(this->greater(key, n));

      if (!N::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 = N::next<A>(n, !dir)->balance();
    if (b == dir)
      A::rotate2(s, !dir, N::next<A>(N::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 = N::next_p(*s, dir);
  }
    }

  A *n = static_cast<A*>(*q);
  *t = n;
  *q = N::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(n->next<A>(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(n->next<A>(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

}
```

15.99 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.

15.99.1 Detailed Description

Basic IO stream.

Definition in file basic_ostream.

15.100 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:
```

```
    void write(IOModifier m);
    void write(long long int c, int len);
    void write(long long unsigned c, int len);
    void write(long long unsigned c, unsigned char base = 10,
         unsigned char len = 0, char pad = ' ');

    Mode mode() const
    { return int_mode; }

    void mode(Mode m)
    { int_mode = m; }

    int int_mode;
  };

  class BasicOStream
  {
  public:
    BasicOStream(IOBackend *b)
      : iob(b)
    {}

    void write(char const *str, unsigned len)
    { if(iob) iob->write(str,len); }

    void write(long long int c, int len)
    { if(iob) iob->write(c,len); }

    void write(long long unsigned c, unsigned char base = 10,
         unsigned char len = 0, char pad = ' ')
    { 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;
}



inline
L4::BasicOStream &
operator << (L4::BasicOStream &o, L4::IONumFmt const &n)
{ return n.print(o); }

```

15.101 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.

15.101.1 Detailed Description

Basic vector.

Definition in file basic_vector.h.

15.102 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 };
```

15.103 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.

15.103.1 Detailed Description

AVL tree.

Definition in file bst.h.

15.104 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
00076
```

```
  typedef __Bst_iter<Node, Node, Fwd> Iterator;
  typedef __Bst_iter<Node, Node const, Fwd> Const_iterator;
  typedef __Bst_iter<Node, Node, Rev> Rev_iterator;
  typedef __Bst_iter<Node, Node const, Rev> Const_rev_iterator;

protected:

  Bst_node *_head;

  Bst() : _head(0) {}

  Node *head() const { return static_cast<Node*>(_head); }

  static Key_type k(Bst_node const *n)
  { return Get_key::key_of(static_cast<Node const *>(n)); }

  static Dir dir(Key_param_type l, Key_param_type r)
  {
    Compare cmp;
    Dir d(cmp(r, l));
    if (d == Direction::L && !cmp(l, r))
      return Direction::N;
    return d;
  }

  static Dir dir(Key_param_type l, Bst_node const *r)
  { return dir(l, k(r)); }

  static bool greater(Key_param_type l, Key_param_type r)
  { return Compare()(r, l); }

  static bool greater(Key_param_type l, Bst_node const *r)
  { return greater(l, k(r)); }

  static bool greater(Bst_node const *l, Bst_node const *r)
  { return greater(k(l), k(r)); }
  template<typename FUNC>
  static void remove_tree(Bst_node *head, FUNC &&callback)
  {
    if (Bst_node *n = Bst_node::next(head, Dir::L))
      remove_tree(n, callback);

    if (Bst_node *n = Bst_node::next(head, Dir::R))
      remove_tree(n, callback);

    callback(static_cast<Node *>(head));
  }

public:

  Const_iterator begin() const { return Const_iterator(head()); }
  Const_iterator end() const { return Const_iterator(); }

  Iterator begin() { return Iterator(head()); }
  Iterator end() { return Iterator(); }

  Const_rev_iterator rbegin() const { return Const_rev_iterator(
head()); }
  Const_rev_iterator rend() const { return Const_rev_iterator(); }

  Rev_iterator rbegin() { return Rev_iterator(head()); }
  Rev_iterator rend() { return Rev_iterator(); }
  Node *find_node(Key_param_type key) const;

  Node *lower_bound_node(Key_param_type key) const;

  Const_iterator find(Key_param_type key) const;

  template<typename FUNC>
  void remove_all(FUNC &&callback)
  {
    if (!_head)
      return;

    Bst_node *head = _head;
    _head = 0;
    remove_tree(head, cxx::forward<FUNC>(callback));
  }


};

/* find an element */
template< typename Node, typename Get_key, class Compare>
inline
Node *
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);
00306 }
00307
00308 /* find an element */
00309 template< typename Node, typename Get_key, class Compare>
00310 inline
00311 typename Bst<Node, Get_key, Compare>::Const_iterator
00312 Bst<Node, Get_key, Compare>::find(
      Key_param_type key) const
00313 {
00314   Bst_node *q = _head;
00315   Bst_node *r = q;
00316
00317   for (Dir d; q; q = Bst_node::next(q, d))
00318     {
00319       d = dir(key, q);
00320       if (d == Dir::N)
00321   return Iterator(static_cast<Node*>(q), static_cast<Node *>(r));
00322
00323       if (d != Dir::L && q == r)
00324   r = Bst_node::next(q, d);
00325     }
00326   return Iterator();
00327 }
00328
00329 }}
```

15.105 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.

15.105.1 Detailed Description

AVL tree.

Definition in file bst_base.h.

15.106 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() {}
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
00066
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:
00092
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
```

```
  Node *next(Bst_node const *p, Direction d)
  { return static_cast<Node *>(p->_c[d.d]); }

  static void rotate(Bst_node **t, Direction idir);
private:
  Bst_node *_c[2];

protected:
  Bst_node() {}

  explicit Bst_node(bool) { _c[0] = _c[1] = 0; }
};

inline
void
Bst_node::rotate(Bst_node **t, Direction idir)
{
  Bst_node *tmp = *t;
  *t = next(tmp, idir);
  next(tmp, idir, next(*t, !idir));
  next(*t, !idir, tmp);
}

}}
```

15.107 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.

15.107.1 Detailed Description

AVL tree.

Definition in file bst_iter.h.

15.108 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>
```

```
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);
    }
}

}}
```

15.109 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.

15.109.1 Detailed Description

Base exceptions.

Definition in file exceptions.

15.110 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
00026 #pragma once
00027
00028 #include <l4/cxx/l4types.h>
00029 #include <l4/cxx/basic_ostream>
00030 #include <l4/sys/err.h>
00031 #include <l4/sys/capability>
00032
```

```
#ifndef L4_CXX_NO_EXCEPTION_BACKTRACE
# define L4_CXX_EXCEPTION_BACKTRACE 20
#endif

#if defined(L4_CXX_EXCEPTION_BACKTRACE)
#include <l4/util/backtrace.h>
#endif

namespace L4
{
  class Exception_tracer
  {
#if defined(L4_CXX_EXCEPTION_BACKTRACE)
  private:
    void *_pc_array[L4_CXX_EXCEPTION_BACKTRACE];
    int   _frame_cnt;

  protected:
#if defined(__PIC__)
    Exception_tracer() throw() : _frame_cnt(0) {}
#else
    Exception_tracer() throw() : _frame_cnt(l4util_backtrace(_pc_array,
L4_CXX_EXCEPTION_BACKTRACE)) {}
#endif

  public:
    void const *const *pc_array() const throw() { return _pc_array; }
    int frame_count() const throw() { return _frame_cnt; }
#else
  protected:
    Exception_tracer() throw() {}

  public:
    void const *const *pc_array() const throw() { return 0; }
    int frame_count() const throw() { return 0; }
#endif
  };

  class Base_exception : public Exception_tracer
  {
  protected:
    Base_exception() throw() {}

  public:
    virtual char const *str() const throw () = 0;

    virtual ~Base_exception() throw () {}
  };

  class Runtime_error : public Base_exception
  {
  private:
    long _errno;
    char _extra[80];

  public:
    explicit Runtime_error(long err_no, char const *extra = 0) throw ()
      : _errno(err_no)
    {
      if (!extra)
        _extra[0] = 0;
      else
        {
          unsigned i = 0;
          for (; i < sizeof(_extra) && extra[i]; ++i)
            _extra[i] = extra[i];
          _extra[i < sizeof(_extra) ? i : sizeof(_extra) - 1] = 0;
        }
    }
    char const *str() const throw ()
    { return l4sys_errtostr(_errno); }

    char const *extra_str() const { return _extra; }
    ~Runtime_error() throw () {}

    long err_no() const throw() { return _errno; }
  };

  class Out_of_memory : public Runtime_error
  {
  public:
    explicit Out_of_memory(char const *extra = "") throw()
    : Runtime_error(-L4_ENOMEM, extra) {}
    ~Out_of_memory() throw() {}
  };
```

```
00197
00198
00203   class Element_already_exists : public Runtime_error
00204   {
00205   public:
00206     explicit Element_already_exists(char const *e = "") throw()
00207     : Runtime_error(-L4_EEXIST, e) {}
00208     ~Element_already_exists() throw() {}
00209   };
00210
00219   class Unknown_error : public Base_exception
00220   {
00221   public:
00222     Unknown_error() throw() {}
00223     char const *str() const throw() { return "unknown error"; }
00224     ~Unknown_error() throw() {}
00225   };
00226
00227
00232   class Element_not_found : public Runtime_error
00233   {
00234   public:
00235     explicit Element_not_found(char const *e = "") throw()
00236     : Runtime_error(-L4_ENOENT, e) {}
00237   };
00238
00246   class Invalid_capability : public Base_exception
00247   {
00248   private:
00249     Cap<void> const _o;
00250
00251   public:
00256     explicit Invalid_capability(Cap<void> const &o) throw() : _o(o) {}
00257     template< typename T>
00258     explicit Invalid_capability(Cap<T> const &o) throw() : _o(o.
      cap()) {}
00259     char const *str() const throw() { return "invalid object"; }
00260
00265     Cap<void> const &cap() const throw() { return _o; }
00266     ~Invalid_capability() throw() {}
00267   };
00268
00275   class Com_error : public Runtime_error
00276   {
00277   public:
00282     explicit Com_error(long err) throw() : Runtime_error(err) {}
00283
00284     ~Com_error() throw() {}
00285   };
00286
00290   class Bounds_error : public Runtime_error
00291   {
00292   public:
00293     explicit Bounds_error(char const *e = "") throw()
00294     : Runtime_error(-L4_ERANGE, e) {}
00295     ~Bounds_error() throw() {}
00296   };
00298 };
00299
00300 inline
00301 L4::BasicOStream &
00302 operator << (L4::BasicOStream &o, L4::Base_exception const &e)
00303 {
00304   o << "Exception: " << e.str() << ", backtrace ...\n";
00305   for (int i = 0; i < e.frame_count(); ++i)
00306     o << L4::n_hex(l4_addr_t(e.pc_array()[i])) << '\n';
00307
00308   return o;
00309 }
00310
00311 inline
00312 L4::BasicOStream &
00313 operator << (L4::BasicOStream &o, L4::Runtime_error const &e)
00314 {
00315   o << "Exception: " << e.str() << ": ";
00316   if (e.extra_str())
00317     o << e.extra_str() << ": ";
00318   o << "backtrace ...\n";
00319   for (int i = 0; i < e.frame_count(); ++i)
00320     o << L4::n_hex(l4_addr_t(e.pc_array()[i])) << '\n';
00321
00322   return o;
00323 }
00324
```

15.111 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.

15.111.1 Detailed Description

IO Stream.

Definition in file iostream.

15.112 iostream

```
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
00024 #ifndef L4_IOSTREAM_H__
00025 #define L4_IOSTREAM_H__
00026
00027 #include <l4/cxx/basic_ostream>
00028
00029 namespace L4 {
00030
00034   extern BasicOStream cout;
00035
00039   extern BasicOStream cerr;
00040
00041
00042   extern void iostream_init();
00043
00044   static void  __attribute__((used, constructor)) __iostream_init()
00045   { iostream_init(); }
00046
00047 };
00048
00049 #endif /* L4_IOSTREAM_H__ */
```

15.113 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.

15.113.1 Detailed Description

IPC helper.

Definition in file ipc_helper.

15.114 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
00025 #pragma once
00026
00027 #include <l4/cxx/exceptions>
00028 #include <l4/sys/utcb.h>
00029
00030 namespace L4
00031 {
00032
00033
00034 #ifdef __EXCEPTIONS
00035
00043   inline void
00044   throw_ipc_exception(L4::Cap<void> const &o,
      l4_msgtag_t const &err,
00045                       l4_utcb_t *utcb)
00046   {
00047     (void)o;
00048     if (err.has_error())
00049       throw (L4::Com_error(l4_error_u(err, utcb)));
00050   }
00051
00060   inline void
00061   throw_ipc_exception(void const *o, l4_msgtag_t const &err,
00062                       l4_utcb_t *utcb)
00063   { throw_ipc_exception(L4::Cap<void>(o), err, utcb); }
00064 #endif
00065
00066 };
```

15.115 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 Ipc::Istream.
- class L4::Ipc::Msg_ptr< T >

  Pointer to an element of type T in an Ipc::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 Ipc::Ostream.
- 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.
- 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.

15.115.1 Detailed Description

IPC stream.

Definition in file ipc_stream.

15.115.2 Function Documentation

15.115.2.1 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 1193 of file ipc_stream.

References L4::Ipc::Ostream::put().

Referenced by operator>>().

Here is the call graph for this function:

Here is the caller graph for this function:

15.115.2.2 operator<<() [2/4]

```
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 1228 of file ipc_stream.

References L4::Ipc::Ostream::tag().

Here is the call graph for this function:

15.115.2.3 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 1244 of file ipc_stream.

References L4::Ipc::Ostream::put().

Here is the call graph for this function:

15.115.2.4 operator<<() [4/4]

```
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 1265 of file ipc_stream.

References L4_DEPRECATED, L4::Ipc::msg_ptr(), and L4::Ipc::Ostream::put().

Here is the call graph for this function:

15.115.2.5 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 1040 of file ipc_stream.

References L4::Ipc::Istream::get(), and operator<<().

Here is the call graph for this function:

15.115.2.6 operator>>() [2/6]

```
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 1074 of file ipc_stream.

References L4::Ipc::Istream::tag().

Here is the call graph for this function:

15.115.2.7 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 1099 of file ipc_stream.

References L4::Ipc::Istream::get(), and L4::Ipc::Istream::has_more().

Here is the call graph for this function:

15.115.2.8 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 1126 of file ipc_stream.

References L4::Ipc::Istream::get().

Here is the call graph for this function:

15.115.2.9 operator>>() [5/6]

```
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 1147 of file ipc_stream.

References L4::Ipc::Istream::get(), and cxx::min().

Here is the call graph for this function:

15.115.2.10 operator>>() [6/6]

```
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 1168 of file ipc_stream.

References L4::Ipc::Istream::get(), and cxx::min().

Here is the call graph for this function:

15.116 ipc_stream

```
// 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/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>

#define L4_CXX_IPC_BACKWARD_COMPAT

namespace L4 {
namespace Ipc {

class Ostream;
class Istream;

namespace Internal {
template< typename T >
class Buf_cp_out
{
public:
  Buf_cp_out(T const *v, unsigned long size) : _v(v), _s(size) {}

  unsigned long size() const { return _s; }

  T const *buf() const { return _v; }

private:
  friend class Ostream;
  T const *_v;
  unsigned long _s;
};
}

template< typename T >
```

```
00114 Internal::Buf_cp_out<T> buf_cp_out(T const *v, unsigned long size)
00115 { return Internal::Buf_cp_out<T>(v, size); }
00116
00117
00118 namespace Internal {
00131 template< typename T >
00132 class Buf_cp_in
00133 {
00134 public:
00143   Buf_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00144
00145   unsigned long &size() const { return *_s; }
00146   T *buf() const { return _v; }
00147 private:
00148   friend class Istream;
00149   T *_v;
00150   unsigned long *_s;
00151 };
00152 }
00153
00171 template< typename T >
00172 Internal::Buf_cp_in<T> buf_cp_in(T *v, unsigned long &size)
00173 { return Internal::Buf_cp_in<T>(v, size); }
00174
00190 template< typename T >
00191 class Str_cp_in
00192 {
00193 public:
00202   Str_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00203
00204   unsigned long &size() const { return *_s; }
00205   T *buf() const { return _v; }
00206 private:
00207   friend class Istream;
00208   T *_v;
00209   unsigned long *_s;
00210 };
00211
00224 template< typename T >
00225 Str_cp_in<T> str_cp_in(T *v, unsigned long &size)
00226 { return Str_cp_in<T>(v, size); }
00227
00240 template< typename T >
00241 class Msg_ptr
00242 {
00243 private:
00244   T **_p;
00245 public:
00252   explicit Msg_ptr(T *&p) : _p(&p) {}
00253   void set(T *p) const { *_p = p; }
00254 };
00255
00263 template< typename T >
00264 Msg_ptr<T> msg_ptr(T *&p)
00265 { return Msg_ptr<T>(p); }
00266
00267
00268 namespace Internal {
00282 template< typename T >
00283 class Buf_in
00284 {
00285 public:
00292   Buf_in(T *&v, unsigned long &size) : _v(&v), _s(&size) {}
00293
00294   void set_size(unsigned long s) const { *_s = s; }
00295   T *&buf() const { return *_v; }
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:
```

```
  Istream(l4_utcb_t *utcb)
  : _tag(), _utcb(utcb),
    _current_msg(reinterpret_cast<char*>(l4_utcb_mr_u(utcb)->mr)),
    _pos(0), _current_buf(0)
  {}

  void reset()
  {
    _pos = 0;
    _current_buf = 0;
    _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
  }

  template< typename T >
  bool has_more(unsigned long count = 1)
  {
    auto const max_bytes = L4_UTCB_GENERIC_DATA_SIZE * sizeof(
l4_umword_t);
    unsigned apos = cxx::Type_traits<T>::align(_pos);
    return (count <= max_bytes / sizeof(T))
           && (apos + (sizeof(T) * count)
               <= _tag.words() * sizeof(l4_umword_t));
  }


  template< typename T >
  unsigned long get(T *buf, unsigned long elems)
  {
    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>())
      {
```

```
00487         va->tag(0);
00488   return 0;
00489       }
00490     get(t);
00491     va->tag(t);
00492     char const *d;
00493     get(msg_ptr(d), va->length());
00494     va->data(d);
00495
00496     return 1;
00497   }
00498
00508   l4_msgtag_t tag() const { return _tag; }
00509
00510
00520   l4_msgtag_t &tag() { return _tag; }
00521
00523
00528   inline bool put(Buf_item const &);
00529
00534   inline bool put(Small_buf const &);
00535
00536
00541
00551   inline l4_msgtag_t wait(l4_umword_t *src)
00552   { return wait(src, L4_IPC_NEVER); }
00553
00564   inline l4_msgtag_t wait(l4_umword_t *src, l4_timeout_t timeout);
00565
00575   inline l4_msgtag_t receive(l4_cap_idx_t src)
00576   { return receive(src, L4_IPC_NEVER); }
00577   inline l4_msgtag_t receive(l4_cap_idx_t src,
      l4_timeout_t timeout);
00578
00580
00584   inline l4_utcb_t *utcb() const { return _utcb; }
00585
00586 protected:
00587   l4_msgtag_t _tag;
00588   l4_utcb_t *_utcb;
00589   char *_current_msg;
00590   unsigned _pos;
00591   unsigned char _current_buf;
00592 };
00593
00594 class Istream_copy : public Istream
00595 {
00596 private:
00597   l4_msg_regs_t _mrs;
00598
00599 public:
00600   Istream_copy(Istream const &o) : Istream(o), _mrs(*l4_utcb_mr_u(o.
      utcb()))
00601   {
00602     // do some reverse mr to utcb trickery
00603     _utcb = (l4_utcb_t*)((l4_addr_t)&_mrs - (l4_addr_t)l4_utcb_mr_u((
      l4_utcb_t*)0));
00604     _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00605   }
00606
00607 };
00608
00623 class Ostream
00624 {
00625 public:
00629   Ostream(l4_utcb_t *utcb)
00630   : _tag(), _utcb(utcb),
00631     _current_msg(reinterpret_cast<char *>(l4_utcb_mr_u(_utcb)->mr)),
00632     _pos(0), _current_item(0)
00633   {}
00634
00638   void reset()
00639   {
00640     _pos = 0;
00641     _current_item = 0;
00642     _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00643   }
00644
00652
00659   template< typename T >
00660   bool put(T *buf, unsigned long size)
00661   {
00662     size *= sizeof(T);
00663     _pos = cxx::Type_traits<T>::align(_pos);
00664     if (Utcb_stream_check::check_utcb_data_offset(_pos + size))
00665       return false;
00666
```

```
    __builtin_memcpy(_current_msg + _pos, buf, size);
    _pos += size;
    return true;
  }

  template< typename T >
  bool put(T const &v)
  {
    _pos = cxx::Type_traits<T>::align(_pos);
    if (Utcb_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
      return false;

    *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
    _pos += sizeof(T);
    return true;
  }

  int put(Varg const &va)
  {
    put(va.tag());
    put(va.data(), va.length());

    return 0;
  }

  template< typename T >
  int put(Varg_t<T> const &va)
  { return put(static_cast<Varg const &>(va)); }

  l4_msgtag_t tag() const { return _tag; }

  l4_msgtag_t &tag() { return _tag; }


  inline bool put_snd_item(Snd_item const &);



  inline l4_msgtag_t send(l4_cap_idx_t dst, long proto = 0, unsigned flags = 0);


  inline l4_utcb_t *utcb() const { return _utcb; }
#if 0

  unsigned long tell() const
  {
    unsigned w = (_pos + sizeof(l4_umword_t)-1) / sizeof(l4_umword_t);
    w -= _current_item * 2;
    _tag = l4_msgtag(0, w, _current_item, 0);
  }
#endif
public:
  l4_msgtag_t prepare_ipc(long proto = 0, unsigned flags = 0)
  {
    unsigned w = (_pos + sizeof(l4_umword_t)-1) / sizeof(l4_umword_t);
    w -= _current_item * 2;
    return l4_msgtag(proto, w, _current_item, flags);
  }

  // XXX: this is a hack for <l4/sys/cxx/ipc_server> adaption
  void set_ipc_params(l4_msgtag_t tag)
  {
    _pos = (tag.words() + tag.items()*2)*sizeof(l4_umword_t);
    _current_item = tag.items();
  }
protected:
  l4_msgtag_t _tag;
  l4_utcb_t *_utcb;
  char *_current_msg;
  unsigned _pos;
  unsigned char _current_item;
};


class Iostream : public Istream, public Ostream
{
public:

  explicit Iostream(l4_utcb_t *utcb)
  : Istream(utcb), Ostream(utcb)
  {}

  // disambiguate those functions
  l4_msgtag_t tag() const { return Istream::tag(); }
  l4_msgtag_t &tag() { return Istream::tag(); }
  l4_utcb_t *utcb() const { return Istream::utcb(); }

```

```
00817   void reset()
00818   {
00819     Istream::reset();
00820     Ostream::reset();
00821   }
00822
00823
00831
00832   using Istream::get;
00833   using Istream::put;
00834   using Ostream::put;
00835
00837
00842
00858   inline l4_msgtag_t call(l4_cap_idx_t dst, l4_timeout_t timeout, long
      proto = 0);
00859   inline l4_msgtag_t call(l4_cap_idx_t dst, long proto = 0);
00860
00876   inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst, long proto =
      0)
00877   { return reply_and_wait(src_dst, L4_IPC_SEND_TIMEOUT_0, proto); }
00878
00879   inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest,
      l4_umword_t *src,
00880                                    long proto = 0)
00881   { return send_and_wait(dest, src, L4_IPC_SEND_TIMEOUT_0, proto); }
00882
00899   inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst,
00900                                     l4_timeout_t timeout, long proto = 0);
00901   inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest,
      l4_umword_t *src,
00902                                    l4_timeout_t timeout, long proto = 0);
00903   inline l4_msgtag_t reply(l4_timeout_t timeout, long proto = 0);
00904   inline l4_msgtag_t reply(long proto = 0)
00905   { return reply(L4_IPC_SEND_TIMEOUT_0, proto); }
00906
00908 };
00909
00910
00911 inline bool
00912 Ostream::put_snd_item(Snd_item const &v)
00913 {
00914   typedef Snd_item T;
00915   _pos = cxx::Type_traits<Snd_item>::align(_pos);
00916   if (Utcb_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
00917     return false;
00918
00919   *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00920   _pos += sizeof(T);
00921   ++_current_item;
00922   return true;
00923 }
00924
00925
00926 inline bool
00927 Istream::put(Buf_item const &item)
00928 {
00929   if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 3)
00930     return false;
00931
00932   l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00933
00934   reinterpret_cast<Buf_item&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00935   _current_buf += 2;
00936   return true;
00937 }
00938
00939
00940 inline bool
00941 Istream::put(Small_buf const &item)
00942 {
00943   if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 2)
00944     return false;
00945
00946   l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00947
00948   reinterpret_cast<Small_buf&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00949   _current_buf += 1;
00950   return true;
00951 }
00952
00953
00954 inline l4_msgtag_t
00955 Ostream::send(l4_cap_idx_t dst, long proto, unsigned flags)
00956 {
00957   l4_msgtag_t tag = prepare_ipc(proto, L4_MSGTAG_FLAGS & flags);
00958   return l4_ipc_send(dst, _utcb, tag, L4_IPC_NEVER);
```

```
}

inline l4_msgtag_t
Iostream::call(l4_cap_idx_t dst, l4_timeout_t timeout, long label)
{
  l4_msgtag_t tag = prepare_ipc(label);
  tag = l4_ipc_call(dst, Ostream::_utcb, tag, timeout);
  Istream::tag() = tag;
  Istream::_pos = 0;
  return tag;
}

inline l4_msgtag_t
Iostream::call(l4_cap_idx_t dst, long label)
{ return call(dst, L4_IPC_NEVER, label); }


inline l4_msgtag_t
Iostream::reply_and_wait(l4_umword_t *src_dst,
l4_timeout_t timeout, long proto)
{
  l4_msgtag_t tag = prepare_ipc(proto);
  tag = l4_ipc_reply_and_wait(Ostream::_utcb, tag, src_dst, timeout);
  Istream::tag() = tag;
  Istream::_pos = 0;
  return tag;
}


inline l4_msgtag_t
Iostream::send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
                        l4_timeout_t timeout, long proto)
{
  l4_msgtag_t tag = prepare_ipc(proto);
  tag = l4_ipc_send_and_wait(dest, Ostream::_utcb, tag, src, timeout);
  Istream::tag() = tag;
  Istream::_pos = 0;
  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;
}


#ifdef L4_CXX_IPC_BACKWARD_COMPAT
namespace L4 {

#if 0
template< typename T > class Ipc_buf_cp_out : public Ipc::Buf_cp_out<T> {};
template< typename T > class Ipc_buf_cp_in : public Ipc::Buf_cp_in<T> {};
template< typename T > class Ipc_buf_in : public Ipc::Buf_in<T> {};
template< typename T > class Msg_ptr : public Ipc::Msg_ptr<T> {};
#endif

template< typename T >
Ipc::Internal::Buf_cp_out<T> ipc_buf_cp_out(T *v, unsigned long size)
  L4_DEPRECATED("Use L4::Ipc::buf_cp_out() now");

template< typename T >
Ipc::Internal::Buf_cp_out<T> ipc_buf_cp_out(T *v, unsigned long size)
{ return Ipc::Internal::Buf_cp_out<T>(v, size); }

```

```

template< typename T >
Ipc::Internal::Buf_cp_in<T> ipc_buf_cp_in(T *v, unsigned long &size)
  L4_DEPRECATED("Use L4::Ipc::buf_cp_in() now");

template< typename T >
Ipc::Internal::Buf_cp_in<T> ipc_buf_cp_in(T *v, unsigned long &size)
{ return Ipc::Internal::Buf_cp_in<T>(v, size); }


template< typename T >
Ipc::Internal::Buf_in<T> ipc_buf_in(T *&v, unsigned long &size)
  L4_DEPRECATED("Use L4::Ipc::buf_in() now");

template< typename T >
Ipc::Internal::Buf_in<T> ipc_buf_in(T *&v, unsigned long &size)
{ return Ipc::Internal::Buf_in<T>(v, size); }


template< typename T >
Ipc::Msg_ptr<T> msg_ptr(T *&p)
  L4_DEPRECATED("Use L4::Ipc::msg_ptr() now");

template< typename T >
Ipc::Msg_ptr<T> msg_ptr(T *&p)
{ return Ipc::Msg_ptr<T>(p); }

typedef Ipc::Istream Ipc_istream L4_DEPRECATED("Use L4::Ipc::Istream now");
typedef Ipc::Ostream Ipc_ostream L4_DEPRECATED("Use L4::Ipc::Ostream now");;
typedef Ipc::Iostream Ipc_iostream L4_DEPRECATED("Use L4::Ipc::Iostream now");;
typedef Ipc::Snd_fpage Snd_fpage L4_DEPRECATED("Use L4::Ipc::Snd_fpage
 now");;
typedef Ipc::Rcv_fpage Rcv_fpage L4_DEPRECATED("Use L4::Ipc::Rcv_fpage
 now");;
typedef Ipc::Small_buf Small_buf L4_DEPRECATED("Use L4::Ipc::Small_buf now");;


namespace Ipc {
template< typename T > class Buf_cp_in : public Internal::Buf_cp_in<T>
{
public:
  Buf_cp_in(T *v, unsigned long &size) : Internal::Buf_cp_in<T>(v, size) {}
};

template< typename T >
class Buf_cp_out : public Internal::Buf_cp_out<T>
{
public:
  Buf_cp_out(T const *v, unsigned long size) : Internal::Buf_cp_out<T>(v, size) {}
};

template< typename T >
class Buf_in : public Internal::Buf_in<T>
{
public:
  Buf_in(T *&v, unsigned long &size) : Internal::Buf_in<T>(v, size) {}
};
} // namespace Ipc
} // namespace L4

template< typename T >
inline
L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
L4::Ipc::Buf_cp_in<T> const &v)
  L4_DEPRECATED("Use L4::Ipc::buf_cp_in() now");

template< typename T >
inline
L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
L4::Ipc::Buf_cp_in<T> const &v)
{ return operator>>(s, static_cast<L4::Ipc::Internal::Buf_cp_in<T> >(v)); }

template< typename T >
inline
L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
L4::Ipc::Buf_in<T> const &v)
  L4_DEPRECATED("Use L4::Ipc::buf_in() now");

template< typename T >
inline
L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
L4::Ipc::Buf_in<T> const &v)
{ return operator>>(s, static_cast<L4::Ipc::Internal::Buf_in<T> >(v)); }

template< typename T >
inline
L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Ipc::Buf_cp_out<T> const &v)
```

```
  L4_DEPRECATED("Use L4::Ipc::buf_cp_out() now");

template< typename T >
inline
L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Ipc::Buf_cp_out<T> const &v)
{ return operator<<(s, static_cast<L4::Ipc::Internal::Buf_cp_out<T> >(v)); }
#endif

namespace L4 { namespace Ipc {
template< typename T >
inline
T read(Istream &s) { T t; s >> t; return t; }

} // namespace Ipc
} // namespace L4
```

15.117 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:

This graph shows which files directly or indirectly include this file:

15.117.1 Detailed Description

L4 IO stream.

Definition in file l4iostream.

15.118 l4iostream

```
00001
00005 /*
00006  * (c) 2004-2009 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
00023 #ifndef L4_L4IOSTREAM_H__
00024 #define L4_L4IOSTREAM_H__
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 }
00047
00048
00049 #endif
```

15.119 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:

15.119.1 Detailed Description

L4 Types.

Definition in file l4types.h.

15.120 l4types.h

```
/*
 * (c) 2004-2009 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/types.h>
#include <l4/cxx/basic_ostream>

```

15.121 l4/cxx/main_thread File Reference

Main thread.

`#include <l4/cxx/thread>`

Include dependency graph for main_thread:

Namespaces

- cxx

  Our C++ library.

15.121.1 Detailed Description

Main thread.

Definition in file main_thread.

15.122 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)
```

```
 *
 * 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_CXX_MAIN_THREAD_H__
#define L4_CXX_MAIN_THREAD_H__

#include <l4/cxx/thread>

namespace cxx {
  class MainThread : public Thread
  {
  public:
    MainThread() : Thread(true)
    {}
  };
};

#endif /* L4_CXX_MAIN_THREAD_H__ */
```

15.123 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.

15.123.1 Detailed Description

Pair implementation.

Definition in file pair.

15.124 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() {}
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
00080 public:
00085   Pair_first_compare(Cmp const &cmp = Cmp()) : _cmp(cmp) {}
00086
00092   bool operator () (Typ const &l, Typ const &r) const
00093   { return _cmp(l.first,r.first); }
00094 };
00095
00096 }
00097
00098 template< typename OS, typename A, typename B >
00099 inline
00100 OS &operator << (OS &os, cxx::Pair<A,B> const &p)
00101 {
00102   os << p.first << ';' << p.second;
00103   return os;
00104 }
00105
```

15.125 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:

15.125.1 Detailed Description

Base exceptions std stream operator.

Definition in file std_exc_io.

15.126 std_exc_io

```
// vi:set ft=cpp: -*- Mode: C++ -*-
/*
 * (c) 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.
 */

#pragma once

#include <l4/cxx/exceptions>
#include <iostream>

inline
std::ostream &
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;
}
```

15.127 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.

15.127.1 Detailed Description

String.

Definition in file string.h.

15.128 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
00024 #ifndef L4_CXX_STRING_H__
00025 #define L4_CXX_STRING_H__
00026
00027 #include <l4/cxx/basic_ostream>
00028
00029 namespace L4 {
00030
00035   class String
00036   {
00037   public:
00038     String(char const *str = "") : _str(str)
00039     {}
00040
00041     unsigned length() const
00042     {
00043       unsigned l;
00044       for (l = 0; _str[l]; l++)
00045         ;
00046       return l;
00047     }
00048
00049     char const *p_str() const { return _str; }
00050
00051   private:
00052     char const *_str;
00053   };
00054
00055 };
00056
00057 inline
00058 L4::BasicOStream &operator << (L4::BasicOStream &o, L4::String const &s)
00059 {
00060   o << s.p_str();
00061   return o;
00062 }
00063
00064 #endif /* L4_CXX_STRING_H__ */
```

15.129 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.

15.129.1 Detailed Description

IRQ handling routines.

Definition in file irq.h.

15.130 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
00028 struct l4irq_t;
00029 typedef struct l4irq_t l4irq_t;
00030
00041 L4_CV l4irq_t *
00042 l4irq_attach(int irqnum);
00043
00055 L4_CV l4irq_t *
00056 l4irq_attach_ft(int irqnum, unsigned mode);
00057
00068 L4_CV l4irq_t *
00069 l4irq_attach_thread(int irqnum, l4_cap_idx_t to_thread);
00070
00082 L4_CV l4irq_t *
00083 l4irq_attach_thread_ft(int irqnum, l4_cap_idx_t to_thread,
00084                        unsigned mode);
00085
00093 L4_CV long
00094 l4irq_wait(l4irq_t *irq);
00095
00104 L4_CV long
00105 l4irq_unmask_and_wait_any(l4irq_t *unmask_irq, l4irq_t **ret_irq);
00106
00114 L4_CV long
00115 l4irq_wait_any(l4irq_t **irq);
00116
00127 L4_CV long
00128 l4irq_unmask(l4irq_t *irq);
00129
00137 L4_CV long
00138 l4irq_detach(l4irq_t *irq);
00139
00140
00141
00142 /*************************************************************************/
00164 L4_CV l4irq_t *
00165 l4irq_request(int irqnum, void (*isr_handler)(void *), void *isr_data,
00166              int irq_thread_prio, unsigned mode);
00167
00175 L4_CV long
00176 l4irq_release(l4irq_t *irq);
00177
00178
```

```
/*****************************************************************************/
/*****************************************************************************/

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
```

15.131 arm/l4/util/irq.h File Reference

ARM specific implementation of irq functions.

`#include <l4/sys/compiler.h>`

Include dependency graph for irq.h:

15.131.1 Detailed Description

ARM specific implementation of irq functions.

Do not use.

Definition in file irq.h.

15.132 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
00039 L4_INLINE
00040 void
00041 l4util_sti(void)
00042 {
00043   extern void __do_not_use_l4util_sti(void);
00044   __do_not_use_l4util_sti();
00045 }
00046
00047
00048 L4_INLINE
00049 void
00050 l4util_flags_save(l4_umword_t *flags)
00051 {
00052   (void)flags;
00053   extern void __do_not_use_l4util_flags_save(void);
00054   __do_not_use_l4util_flags_save();
00055 }
00056
00057 L4_INLINE
00058 void
00059 l4util_flags_restore(l4_umword_t *flags)
00060 {
00061   (void)flags;
00062   extern void __do_not_use_l4util_flags_restore(void);
00063   __do_not_use_l4util_flags_restore();
00064 }
00065
00066 EXTERN_C_END
00067
00068 #endif //__GNUC__
00069
00070 #endif /* ! __L4UTIL__ARCH_ARCH__IRQ_H__ */
```

15.133 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.

15.133.1 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.

15.133.2 Function Documentation

15.133.2.1 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 EXTERN_C_END, l4util_in8(), and l4util_out8().

Here is the call graph for this function:

15.134 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 }
```

```
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
```

15.135 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.

15.135.1 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.

15.135.2 Function Documentation

15.135.2.1 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 EXTERN_C_END, l4util_in8(), and l4util_out8().

Here is the call graph for this function:

15.136 irq.h

```
00001
00010 /*
00011  * (c) 2003-2009 Author(s)
00012  *     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_IRQ_H__
#define __L4_IRQ_H__

#include <l4/sys/compiler.h>
#include <l4/util/port_io.h>

EXTERN_C_BEGIN

L4_INLINE void
l4util_irq_acknowledge(unsigned int irq);

static inline void
l4util_cli (void)
{
  __asm__ __volatile__ ("cli" : : : "memory");
}

static inline void
l4util_sti (void)
{
  __asm__ __volatile__ ("sti" : : : "memory");
}

static inline void
l4util_flags_save (l4_umword_t *flags)
{
  __asm__ __volatile__ ("pushfl ; popl %0 " :"=g" (*flags) : :"memory");
}

static inline void
l4util_flags_restore (l4_umword_t *flags)
{
  __asm__ __volatile__ ("pushl %0 ; popfl" : :"g" (*flags) : "memory");
}

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
```

15.137 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_attach (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread) L4_NOTHROW

  Attach a thread to an interrupt source.
- l4_msgtag_t l4_irq_attach_u (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW

  Attach a thread to this interrupt.
- 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.
- 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.

15.137.1 Detailed Description

C Irq interface.

Definition in file irq.h.

15.138 irq.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.
 */
#pragma once

#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>
#include <l4/sys/rcv_endpoint.h>

L4_INLINE l4_msgtag_t
l4_irq_attach(l4_cap_idx_t irq, l4_umword_t label,
              l4_cap_idx_t thread) L4_NOTHROW
  L4_DEPRECATED("Use l4_rcv_ep_bind_thread().");
```

```
L4_INLINE l4_msgtag_t
l4_irq_attach_u(l4_cap_idx_t irq, l4_umword_t label,
                l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW
  L4_DEPRECATED("Use l4_rcv_ep_bind_thread_u().");

L4_INLINE l4_msgtag_t
l4_irq_mux_chain(l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW;

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_INLINE l4_msgtag_t
l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;


L4_INLINE l4_msgtag_t
l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
            l4_timeout_t to) L4_NOTHROW;

L4_INLINE 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;

L4_INLINE l4_msgtag_t
l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;

enum L4_irq_sender_op
{
  L4_IRQ_SENDER_OP_ATTACH    = 0,
  L4_IRQ_SENDER_OP_DETACH    = 1
};

enum L4_irq_mux_op
{
  L4_IRQ_MUX_OP_CHAIN = 0
};

enum L4_irq_op
{
  L4_IRQ_OP_TRIGGER   = 2,
  L4_IRQ_OP_EOI       = 4
};

/*****************************************************************************
 * Implementations
 */

L4_INLINE l4_msgtag_t
l4_irq_attach_u(l4_cap_idx_t irq, l4_umword_t label,
                l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW
{
  int items = 0;
  l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
  m->mr[0] = L4_IRQ_SENDER_OP_ATTACH;
  m->mr[1] = label;

  if (!l4_is_invalid_cap(thread))
    {
```

```
      items = 1;
      m->mr[2] = l4_map_obj_control(0, 0);
      m->mr[3] = l4_obj_fpage(thread, 0, L4_FPAGE_RWX).raw;
    }
  return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_SENDER, 2, items, 0),
                     L4_IPC_NEVER);
}

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_FPAGE_RWX).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_attach(l4_cap_idx_t irq, l4_umword_t label,
              l4_cap_idx_t thread) L4_NOTHROW
{
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wdeprecated-declarations"
  return l4_irq_attach_u(irq, label, thread, l4_utcb());
#pragma GCC diagnostic pop
}

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
{
  return l4_irq_unmask_u(irq, l4_utcb());
}

```

15.139 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.

15.140 edid.h

```
00001
00004 /*
00005  * (c) 2014 Matthias Lange <matthias.lange@kernkonzept.com>
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU Lesser General Public License 2.1.
00009  * Please see the COPYING-LGPL-2.1 file for details.
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/compiler.h>
00014
00023 enum Libedid_consts
00024 {
00025   Libedid_block_size = 128,
00026 };
00027
00028 __BEGIN_DECLS
00029
00036 int libedid_check_header(const unsigned char *edid);
00037
00044 int libedid_checksum(const unsigned char *edid);
00045
00052 unsigned libedid_version(const unsigned char *edid);
00053
00060 unsigned libedid_revision(const unsigned char *edid);
00061
00068 void libedid_pnp_id(const unsigned char *edid, unsigned char *id);
00069
00077 void libedid_prefered_resolution(const unsigned char *edid,
00078                                  unsigned *w, unsigned *h);
00079
00086 unsigned libedid_num_ext_blocks(const unsigned char *edid);
00087
00094 unsigned libedid_dump_standard_timings(const unsigned char *edid);
00095
00101 void libedid_dump(const unsigned char *edid);
00102
00105 __END_DECLS
```

15.141 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[]

Macros

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

Typedefs

- typedef unsigned int gfxbitmap_color_t

  Standard color type.
- typedef unsigned int gfxbitmap_color_pix_t

  Specific color type.

Functions

- 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.

15.141.1 Detailed Description

Bitmap renderer header file.

Definition in file bitmap.h.

15.141.2 Macro Definition Documentation

15.141.2.1 pSLIM_BMAP_START_LSB

```
#define pSLIM_BMAP_START_LSB 0x01
```

the other way round

Definition at line 43 of file bitmap.h.

15.142 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
00044
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
```

15.143 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.

15.143.1 Detailed Description

Bitmap font renderer header file.

Definition in file font.h.

15.144 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
00031 #define GFXBITMAP_DEFAULT_FONT (void *)0
00032
00038 enum { GFXBITMAP_USE_STRLEN = ~0U };
00039
00040 EXTERN_C_BEGIN
00041
00043 typedef void *gfxbitmap_font_t;
00044
00053 L4_CV int gfxbitmap_font_init(void);
00054
00062 L4_CV gfxbitmap_font_t gfxbitmap_font_get(const char *name);
00063
00070 L4_CV unsigned
00071 gfxbitmap_font_width(gfxbitmap_font_t font);
00072
00079 L4_CV unsigned
00080 gfxbitmap_font_height(gfxbitmap_font_t font);
00081
00089 L4_CV void *
00090 gfxbitmap_font_data(gfxbitmap_font_t font, unsigned c);
00091
```

```
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
```

15.145 l4/libgfxbitmap/support File Reference

Terminal support functionality.

`#include <l4/re/video/view>`

Include dependency graph for support:

15.145.1 Detailed Description

Terminal support functionality.

Date

2009

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file support.

15.146 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
```

15.147 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 { , L4RE_DS_MAP_NORMAL = 0x00, L4RE_DS_MAP_CACHEABLE = L4RE_DS_MAP_NORMAL, L4RE_DS_MAP_BUFFERABLE = 0x10, L4RE_DS_MAP_UNCACHEABLE = 0x20, L4RE_DS_MAP_CACHING_MASK = 0x30, L4RE_DS_MAP_CACHING_SHIFT = 4 }

  Flags to specify the memory mapping type of a request.

Functions

- long l4re_ds_clear (const l4re_ds_t ds, l4_addr_t offset, unsigned long size) L4_NOTHROW
- long l4re_ds_allocate (const l4re_ds_t ds, l4_addr_t offset, l4_size_t size) L4_NOTHROW
- int l4re_ds_copy_in (const l4re_ds_t ds, l4_addr_t dst_offs, const l4re_ds_t src, l4_addr_t src_offs, unsigned long size) L4_NOTHROW
- unsigned long l4re_ds_size (const l4re_ds_t ds) L4_NOTHROW
- long l4re_ds_flags (const l4re_ds_t ds) L4_NOTHROW
- int l4re_ds_info (const l4re_ds_t ds, l4re_ds_stats_t *stats) L4_NOTHROW
- int l4re_ds_phys (const l4re_ds_t ds, l4_addr_t offset, l4_addr_t *phys_addr, l4_size_t *phys_size) L4_NOTHROW

  Return physical address.

15.147.1 Detailed Description

Data space C interface.

Definition in file dataspace.h.

15.148 dataspace.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/types.h>

EXTERN_C_BEGIN

typedef l4_cap_idx_t l4re_ds_t;

typedef struct {
  unsigned long size;
  unsigned long flags;
} l4re_ds_stats_t;

enum l4re_ds_map_flags {
  L4RE_DS_MAP_FLAG_RO = 0,
  L4RE_DS_MAP_FLAG_RW = 1,

  L4RE_DS_MAP_NORMAL        = 0x00,
  L4RE_DS_MAP_CACHEABLE     = L4RE_DS_MAP_NORMAL,
  L4RE_DS_MAP_BUFFERABLE    = 0x10,
  L4RE_DS_MAP_UNCACHEABLE   = 0x20,
  L4RE_DS_MAP_CACHING_MASK  = 0x30,
  L4RE_DS_MAP_CACHING_SHIFT = 4,
};

L4_CV int
l4re_ds_map(const l4re_ds_t ds, l4_addr_t offset, unsigned long flags,
            l4_addr_t local_addr, l4_addr_t min_addr,
    l4_addr_t max_addr) L4_NOTHROW;

L4_CV int
l4re_ds_map_region(const l4re_ds_t ds, l4_addr_t offset, unsigned long flags,
                   l4_addr_t min_addr, l4_addr_t max_addr)
    L4_NOTHROW;

L4_CV long
l4re_ds_clear(const l4re_ds_t ds, l4_addr_t offset, unsigned long size)
    L4_NOTHROW;

L4_CV long
l4re_ds_allocate(const l4re_ds_t ds,
                 l4_addr_t offset, l4_size_t size) L4_NOTHROW;

L4_CV int
l4re_ds_copy_in(const l4re_ds_t ds, l4_addr_t dst_offs, const
    l4re_ds_t src,
                l4_addr_t src_offs, unsigned long size) L4_NOTHROW;

L4_CV unsigned long
l4re_ds_size(const l4re_ds_t ds) L4_NOTHROW;

L4_CV long
l4re_ds_flags(const l4re_ds_t ds) L4_NOTHROW;

L4_CV int
l4re_ds_info(const l4re_ds_t ds, l4re_ds_stats_t *stats)
    L4_NOTHROW;

L4_CV int
l4re_ds_phys(const l4re_ds_t ds, l4_addr_t offset,
             l4_addr_t *phys_addr, l4_size_t *phys_size)
    L4_NOTHROW;


EXTERN_C_END
```

15.149 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.

15.149.1 Detailed Description

Debug C interface.

Definition in file debug.h.

15.150 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
```

```
 * 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>

EXTERN_C_BEGIN

L4_CV long
l4re_debug_obj_debug(l4_cap_idx_t srv, unsigned long function)
    L4_NOTHROW;

EXTERN_C_END
```

15.151 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, l4_addr_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.

15.151.1 Detailed Description

DMA space C interface.

Definition in file dma_space.h.

15.151.2 Typedef Documentation

15.151.2.1 l4re_dma_space_dma_addr_t

```
typedef l4_uint64_t l4re_dma_space_dma_addr_t
```

Data type for DMA addresses.

Definition at line 62 of file dma_space.h.

15.151.3 Enumeration Type Documentation

15.151.3.1 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.

15.151.3.2 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	
L4RE_DMA_SPACE_PHYS_SPACE	

Definition at line 48 of file dma_space.h.

15.152 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,
```

```
  L4RE_DMA_SPACE_NONE
};

enum l4re_dma_space_space_attribs
{
  L4RE_DMA_SPACE_COHERENT   = 1 << 0,
  L4RE_DMA_SPACE_PHYS_SPACE = 1 << 1,
};

typedef l4_cap_idx_t l4re_dma_space_t;

typedef l4_uint64_t l4re_dma_space_dma_addr_t;

L4_CV long
l4re_dma_space_map(l4re_dma_space_t dma,
l4re_ds_t src, l4_addr_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;


L4_CV 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;

L4_CV long
l4re_dma_space_associate(l4re_dma_space_t dma,
l4_cap_idx_t dma_task,
                         unsigned long attr) L4_NOTHROW;

L4_CV long
l4re_dma_space_disassociate(l4re_dma_space_t dma);


EXTERN_C_END

```

15.153 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_N↩OTHROW

  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.

15.153.1 Detailed Description

Event C interface.

Definition in file event.h.

15.154 event.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/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/event.h>

EXTERN_C_BEGIN

typedef struct
{
  long long time;
  unsigned short type;
  unsigned short code;
  int value;
  l4_umword_t stream_id;
} l4re_event_t;

L4_CV long
l4re_event_get_buffer(const l4_cap_idx_t server,
                      const l4re_ds_t ds) L4_NOTHROW;

L4_CV long
l4re_event_get_num_streams(const l4_cap_idx_t server) L4_NOTHROW;

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
```

15.155 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:

15.155.1 Detailed Description

Events.

Definition in file event.h.

15.156 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
```

```
 * 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/compiler.h>
#include <l4/sys/l4int.h>

typedef struct L4_EXPORT_TYPE l4re_event_stream_id_t
{
  l4_uint16_t bustype;
  l4_uint16_t vendor;
  l4_uint16_t product;
  l4_uint16_t version;
} l4re_event_stream_id_t;

typedef struct L4_EXPORT_TYPE l4re_event_absinfo_t
{
  l4_int32_t value;
  l4_int32_t min;
  l4_int32_t max;
  l4_int32_t fuzz;
  l4_int32_t flat;
  l4_int32_t resolution;
} l4re_event_absinfo_t;

enum l4re_event_stream_max_values_t
{
  L4RE_EVENT_EV_MAX  = 0x1f,
  L4RE_EVENT_KEY_MAX = 0x1ff,
  L4RE_EVENT_REL_MAX = 0xf,
  L4RE_EVENT_ABS_MAX = 0x3f,
  L4RE_EVENT_PROP_MAX = 0x1f,
  L4RE_EVENT_SW_MAX   = 0xf, // should be >= L4RE_SW_MAX
};

enum l4re_event_stream_props_t
{
  L4RE_EVENT_STREAM_CALIBRATE = 0x001,
};


#define __UNUM_B(x) ((x+1) + sizeof(unsigned long)*8 - 1) / (sizeof(unsigned long)*8)

typedef struct L4_EXPORT_TYPE l4re_event_stream_info_t
{
  l4_umword_t stream_id;
  char name[32];
  char phys[32];
  l4re_event_stream_id_t id;

  unsigned long propbits[__UNUM_B(L4RE_EVENT_PROP_MAX)];

  unsigned long evbits[__UNUM_B(L4RE_EVENT_EV_MAX)];
  unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
  unsigned long relbits[__UNUM_B(L4RE_EVENT_REL_MAX)];
  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

```

15.157 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.

15.157.1 Detailed Description

Log C interface.

Definition in file log.h.

15.158 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.
```

```
 * 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>

EXTERN_C_BEGIN

L4_CV L4_INLINE void
l4re_log_print(char const *string) L4_NOTHROW;

L4_CV L4_INLINE void
l4re_log_printn(char const *string, int len) L4_NOTHROW;



L4_CV void
l4re_log_print_srv(const l4_cap_idx_t logcap,
                   char const *string) L4_NOTHROW;

L4_CV void
l4re_log_printn_srv(const l4_cap_idx_t logcap,
                    char const *string, int len) L4_NOTHROW;


/********** Implementations **************************/

L4_CV L4_INLINE void
l4re_log_print(char const *string) L4_NOTHROW
{
  l4re_log_print_srv(l4re_global_env->log, string);
}

L4_CV L4_INLINE void
l4re_log_printn(char const *string, int len) L4_NOTHROW
{
  l4re_log_printn_srv(l4re_global_env->log, string, len);
}

EXTERN_C_END
```

15.159 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_N↩OTHROW

  Allocate memory.
- long l4re_ma_free (l4re_ds_t const mem) L4_NOTHROW

  Free 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.
- long l4re_ma_free_srv (l4_cap_idx_t srv, l4re_ds_t const mem) L4_NOTHROW

  Free memory.

15.159.1 Detailed Description

Memory allocator C interface.

Definition in file mem_alloc.h.

15.160 mem_alloc.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/re/env.h>
#include <l4/sys/consts.h>

#include <l4/re/c/dataspace.h>

EXTERN_C_BEGIN

enum l4re_ma_flags {
  L4RE_MA_CONTINUOUS  = 0x01,
  L4RE_MA_PINNED      = 0x02,
  L4RE_MA_SUPER_PAGES = 0x04,
};


L4_CV L4_INLINE long
l4re_ma_alloc(long size, l4re_ds_t const mem,
              unsigned long flags) L4_NOTHROW;

L4_CV L4_INLINE long
l4re_ma_alloc_align(long size, l4re_ds_t const mem,
                    unsigned long flags, unsigned long align) L4_NOTHROW;

/* Deprecation message added Q4 2016.
 * Remove together with L4Re::Mem_alloc::free() */
L4_CV L4_INLINE long
l4re_ma_free(l4re_ds_t const mem) L4_NOTHROW
L4_DEPRECATED("This function is an empty stub and remains for backward compatibility only. See documentation for replacement options.");



L4_CV 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;

/* Deprecation message added Q4 2016.
 * Remove together with L4Re::Mem_alloc::free() */
L4_CV long
l4re_ma_free_srv(l4_cap_idx_t srv, l4re_ds_t const mem) L4_NOTHROW
L4_DEPRECATED("This function is an empty stub and remains for backward compatibility only. See documentation for replacement options.");



/***************** Implementation *****************/

/* Just warn actual users, but not for internal implementations */
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wdeprecated-declarations"

L4_CV L4_INLINE long
l4re_ma_alloc(long size, l4re_ds_t const mem,
              unsigned long flags) L4_NOTHROW
{
  return l4re_ma_alloc_align_srv(l4re_global_env->mem_alloc, size, mem,
                                 flags, 0);
}

```

```
L4_CV L4_INLINE long
l4re_ma_alloc_align(long size, l4re_ds_t const mem,
                    unsigned long flags, unsigned long align) L4_NOTHROW
{
  return l4re_ma_alloc_align_srv(l4re_global_env->mem_alloc, size, mem,
                                 flags, align);
}

L4_CV L4_INLINE long
l4re_ma_free(l4re_ds_t const mem) L4_NOTHROW
{
  return l4re_ma_free_srv(l4re_global_env->mem_alloc, mem);
}

#pragma GCC diagnostic pop

EXTERN_C_END
```

15.161 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.

15.161.1 Detailed Description

Namespace functions, C interface.

Definition in file namespace.h.

15.162 namespace.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
00039 enum l4re_ns_register_flags {
00040   L4RE_NS_REGISTER_RO  = L4_FPAGE_RO,
00041   L4RE_NS_REGISTER_DIR = 0x10,
00042   L4RE_NS_REGISTER_RW  = L4_FPAGE_RX,
00043   L4RE_NS_REGISTER_RWS = L4_FPAGE_RWX,
00044   L4RE_NS_REGISTER_S   = L4_FPAGE_W,
00045 };
00046
00047 EXTERN_C_BEGIN
00048
00053 typedef l4_cap_idx_t l4re_namespace_t;
00054
00055
00056
00065 L4_CV long
00066 l4re_ns_query_to_srv(l4re_namespace_t srv, char const *name,
00067                      l4_cap_idx_t const cap, int timeout) L4_NOTHROW;
00068
00085 L4_CV L4_INLINE long
00086 l4re_ns_query_srv(l4re_namespace_t srv, char const *name,
00087                   l4_cap_idx_t const cap) L4_NOTHROW;
00088
00096 L4_CV long
00097 l4re_ns_register_obj_srv(l4re_namespace_t srv, char const *name,
00098                          l4_cap_idx_t const obj, unsigned flags)
      L4_NOTHROW;
00099
00100
```

```
/****** 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
```

15.163 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_t {
  L4RE_RM_READ_ONLY = 0x01, L4RE_RM_NO_ALIAS = 0x02, L4RE_RM_PAGER = 0x04, L4RE_RM_↩
  RESERVED = 0x08,
  L4RE_RM_REGION_FLAGS = 0x0f, L4RE_RM_OVERMAP = 0x10, L4RE_RM_SEARCH_ADDR = 0x20,
  L4RE_RM_IN_AREA = 0x40,
  L4RE_RM_EAGER_MAP = 0x80, L4RE_RM_ATTACH_FLAGS = 0xf0 }

  Flags for region operations.

Functions

- int l4re_rm_reserve_area (l4_addr_t *start, unsigned long size, unsigned flags, unsigned char align) L4_NOTHROW
- int l4re_rm_free_area (l4_addr_t addr) L4_NOTHROW
- int l4re_rm_attach (void **start, unsigned long size, unsigned long flags, l4re_ds_t const mem, l4_addr_t offs, unsigned char align) L4_NOTHROW
- int l4re_rm_detach (void *addr) L4_NOTHROW

  Detach and unmap in current task.
- int l4re_rm_detach_ds (void *addr, l4re_ds_t *ds) L4_NOTHROW

  Detach, unmap and return affected dataspace in 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, l4_addr_t *offset, unsigned *flags, l4re_ds_t *m) L4_NOTHROW
- 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, unsigned 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, unsigned long flags, l4re_ds_t const mem, l4_addr_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, l4_addr_t *offset, unsigned *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.

15.163.1 Detailed Description

Region map interface, C interface.

Definition in file rm.h.

15.164 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
```

```
#include <l4/re/env.h>
#include <l4/re/c/dataspace.h>

EXTERN_C_BEGIN

enum l4re_rm_flags_t {
  L4RE_RM_READ_ONLY    = 0x01,
  L4RE_RM_NO_ALIAS     = 0x02,
  L4RE_RM_PAGER        = 0x04,
  L4RE_RM_RESERVED     = 0x08,
  L4RE_RM_REGION_FLAGS = 0x0f,
  L4RE_RM_OVERMAP      = 0x10,
  L4RE_RM_SEARCH_ADDR  = 0x20,
  L4RE_RM_IN_AREA      = 0x40,
  L4RE_RM_EAGER_MAP    = 0x80,
  L4RE_RM_ATTACH_FLAGS = 0xf0,
};



L4_CV L4_INLINE int
l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
                     unsigned 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, unsigned long flags,
               l4re_ds_t const mem, l4_addr_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,
l4_addr_t *offset,
             unsigned *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,
                         unsigned 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,
                   unsigned long flags, l4re_ds_t const mem, l4_addr_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, l4_addr_t *offset,
                 unsigned *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,
                     unsigned 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, unsigned long flags,
               l4re_ds_t const mem, l4_addr_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,
l4_addr_t *offset,
             unsigned *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
```

15.165 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.

15.165.1 Detailed Description

Capability allocator C interface.

Definition in file cap_alloc.h.

15.166 cap_alloc.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/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
```

15.167 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.

15.167.1 Detailed Description

Kumem allocator utility C interface.

Definition in file kumem_alloc.h.

15.167.2 Function Documentation

15.167.2.1 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 minmum allocatable amount is one page. A portable implementation should not depend on allocations greater than 16KiB to succeed.

Examples:

examples/sys/aliens/main.c.

15.168 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
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/sys/linkage.h>
00033
00034 EXTERN_C_BEGIN
00035
00039 L4_CV int
00040 l4re_util_kumem_alloc(l4_addr_t *mem, unsigned pages_order,
00041                       l4_cap_idx_t task, l4_cap_idx_t rm)
      L4_NOTHROW;
00042
00043 EXTERN_C_END
```

15.169 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:

15.169.1 Detailed Description

Framebuffer utility functionality.

Definition in file goos_fb.h.

15.170 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  */
```

```
#pragma once

#include <l4/sys/compiler.h>
#include <l4/re/c/video/goos.h>
#include <l4/sys/err.h>
#include <l4/re/c/dataspace.h>

EXTERN_C_BEGIN

typedef struct
{
  unsigned long _obj_buf[6];
} l4re_util_video_goos_fb_t;

L4_CV int
l4re_util_video_goos_fb_setup_name(l4re_util_video_goos_fb_t *goosfb,
                                   char const *name) L4_NOTHROW;

L4_CV void
l4re_util_video_goos_fb_destroy(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;

L4_CV int
l4re_util_video_goos_fb_view_info(l4re_util_video_goos_fb_t *goosfb,
                                  l4re_video_view_info_t *info)
    L4_NOTHROW;

L4_CV void *
l4re_util_video_goos_fb_attach_buffer(l4re_util_video_goos_fb_t *goosfb)
    L4_NOTHROW;

L4_CV int
l4re_util_video_goos_fb_refresh(l4re_util_video_goos_fb_t *goosfb,
                                int x, int y, int w, int h) L4_NOTHROW;

L4_CV l4re_ds_t
l4re_util_video_goos_fb_buffer(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;

L4_CV l4_cap_idx_t
l4re_util_video_goos_fb_goos(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;

EXTERN_C_END
```

15.171 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.

15.171.1 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.

15.172 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
```

15.173 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.

15.173.1 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.

15.174 goos.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
```

```
 * 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
```

15.175 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_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.

15.175.1 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.

15.176 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,
00141                                int h, unsigned long bofs) L4_NOTHROW;
00142
00152 L4_CV int
00153 l4re_video_view_stack(l4re_video_view_t *view,
      l4re_video_view_t *pivot,
00154                       int behind) L4_NOTHROW;
00155
00156 EXTERN_C_END
00157
```

15.177 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 Auto_cap and Auto_del_cap.
- class L4Re::Smart_count_cap< Unmap_flags >

  Helper for Ref_cap and Ref_del_cap.

Namespaces

- L4Re

  L4Re C++ Interfaces.

15.177.1 Detailed Description

Abstract capability-allocator interface.

Definition in file cap_alloc.

15.178 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() throw() = 0;
00057   virtual void take(L4::Cap<void> cap) throw() = 0;
00058
00063   template< typename T >
```

```
  L4::Cap<T> alloc() throw()
  { return L4::cap_cast<T>(alloc()); }

  virtual void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
                        unsigned unmap_flags = L4_FP_ALL_SPACES) throw() = 0;
  virtual bool release(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
                        unsigned unmap_flags = L4_FP_ALL_SPACES) throw() = 0;

  virtual ~Cap_alloc() = 0;

  template< typename CAP_ALLOC >
  static inline L4Re::Cap_alloc *
  get_cap_alloc(CAP_ALLOC &ca)
  {
    struct CA : public L4Re::Cap_alloc
    {
      CAP_ALLOC &_ca;
      L4::Cap<void> alloc() throw() override { return _ca.alloc(); }
      void take(L4::Cap<void> cap) throw() override { _ca.take(cap); }

      void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
                unsigned unmap_flags = L4_FP_ALL_SPACES) throw() override
      { _ca.free(cap, task, unmap_flags); }

      bool release(L4::Cap<void> cap, l4_cap_idx_t task,
                   unsigned unmap_flags) throw() override
      { return _ca.release(cap, task, unmap_flags); }

      void operator delete(void *) {}

      CA(CAP_ALLOC &ca) : _ca(ca) {}
    };

    static CA _ca(ca);
    return &_ca;
  }
};

template<typename ALLOC>
struct Cap_alloc_t : ALLOC, L4Re::Cap_alloc
{
  template<typename ...ARGS>
  Cap_alloc_t(ARGS &&...args) : ALLOC(cxx::forward<ARGS>(args)...) {}

  L4::Cap<void> alloc() throw() override { return ALLOC::alloc(); }
  void take(L4::Cap<void> cap) throw() override { ALLOC::take(cap); }

  void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
            unsigned unmap_flags = L4_FP_ALL_SPACES) throw() override
  { ALLOC::free(cap, task, unmap_flags); }

  bool release(L4::Cap<void> cap, l4_cap_idx_t task,
               unsigned unmap_flags) throw() override
  { return ALLOC::release(cap, task, unmap_flags); }

  void operator delete(void *) {}
};

inline
Cap_alloc::~Cap_alloc()
{}

extern Cap_alloc *virt_cap_alloc;

template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
class Smart_cap_auto
{
private:
  Cap_alloc *_ca;

public:
  Smart_cap_auto() : _ca(0) {}
  Smart_cap_auto(Cap_alloc *ca) : _ca(ca) {}

  void free(L4::Cap_base &c)
  {
    if (c.is_valid() && _ca)
      _ca->free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);

    invalidate(c);
  }

  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) throw()
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) throw()
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 };
00222
00248 template< typename T >
00249 struct Auto_cap
00250 {
00251   typedef L4::Smart_cap<T, Smart_cap_auto<> > Cap;
00252 } L4_DEPRECATED("use L4::Re::Unique_cap");
00253
00284 template< typename T >
00285 struct Auto_del_cap
00286 {
00287   typedef L4::Smart_cap<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
       > Cap;
00288 } L4_DEPRECATED("use L4::Re::Unique_del_cap");
00291 }
```

15.179 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 Auto_cap and Auto_del_cap.
- class L4Re::Util::Smart_count_cap< Unmap_flags >

  Helper for Ref_cap and Ref_del_cap.
- struct L4Re::Util::Auto_cap< T >

  Automatic capability that implements automatic free and unmap of the capability selector.
- struct L4Re::Util::Auto_del_cap< T >

  Automatic capability that implements automatic free and unmap+delete of the capability selector.
- 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 >
  Auto_cap< T >::Cap L4Re::Util::make_auto_cap ()

  Allocate a capability slot and wrap it in an Auto_cap.
- template<typename T >
  Auto_del_cap< T >::Cap L4Re::Util::make_auto_del_cap ()

  Allocate a capability slot and wrap it in an Auto_del_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<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.

15.179.1 Detailed Description

Capability allocator.

Definition in file cap_alloc.

15.180 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
```

```
#pragma once

#include <l4/re/util/cap_alloc_impl.h>
#include <l4/sys/smart_capability>
#include <l4/sys/task>
#include <l4/re/consts>

namespace L4Re { namespace Util {

extern _Cap_alloc &cap_alloc;

template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
class Smart_cap_auto
{
public:
  static void free(L4::Cap_base &c)
  {
    if (c.is_valid())
      {
        cap_alloc.free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
        c.invalidate();
      }
  }

  static void invalidate(L4::Cap_base &c)
  {
    if (c.is_valid())
      c.invalidate();
  }

  static L4::Cap_base copy(L4::Cap_base const &src)
  {
    L4::Cap_base r = src;
    invalidate(const_cast<L4::Cap_base &>(src));
    return r;
  }
};


template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
class Smart_count_cap
{
public:
  static void free(L4::Cap_base &c) throw()
  {
    if (c.is_valid())
      {
        if (cap_alloc.release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
    c.invalidate();
      }
  }

  static void invalidate(L4::Cap_base &c) throw()
  {
    if (c.is_valid())
      c.invalidate();
  }

  static L4::Cap_base copy(L4::Cap_base const &src)
  {
    cap_alloc.take(L4::Cap<void>(src.cap()));
    return src;
  }
};


template< typename T >
struct Auto_cap
{
  typedef L4::Smart_cap<T, Smart_cap_auto< L4_FP_ALL_SPACES>
 > Cap;
} L4_DEPRECATED("use L4Re::Util::Unique_cap");

template< typename T >
struct Auto_del_cap
{
  typedef L4::Smart_cap<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
 > Cap;
} L4_DEPRECATED("use L4Re::Util::Unique_cap");

template< typename T >
struct Ref_cap
{
  typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_ALL_SPACES>
 > Cap;
};

```

```
template< typename T >
struct Ref_del_cap
{
  typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_DELETE_OBJ>
 > Cap;
};

#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wdeprecated-declarations"
template< typename T >
typename Auto_cap<T>::Cap
L4_DEPRECATED("use make_unique_cap")
make_auto_cap()
{ return typename Auto_cap<T>::Cap(cap_alloc.alloc<T>()); }

template< typename T >
typename Auto_del_cap<T>::Cap
make_auto_del_cap()
{ return typename Auto_del_cap<T>::Cap(cap_alloc.alloc<T>()); }
#pragma GCC diagnostic pop

template< typename T >
typename Ref_cap<T>::Cap
make_ref_cap() { return typename Ref_cap<T>::Cap(cap_alloc.alloc<T>()); }

template< typename T >
typename Ref_del_cap<T>::Cap
make_ref_del_cap()
{ return typename Ref_del_cap<T>::Cap(cap_alloc.alloc<T>()); }

}}

```

15.181 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.

15.181.1 Detailed Description

Constants.

Definition in file consts.

15.182 consts

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
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 #pragma once
00025
00026 #include <l4/sys/capability>
00027 #include <l4/sys/consts.h>
00028 #include <l4/re/env.h>
00029
00030 namespace L4Re {
00031   static L4::Cap<L4::Task>::Cap_type const This_task
00032     = (L4::Cap<L4::Task>::Cap_type)(L4RE_THIS_TASK_CAP);
00033 }
```

15.183 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 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::Stats

  Information about the dataspace.

Namespaces

- L4Re

  L4Re C++ Interfaces.

15.183.1 Detailed Description

Dataspace interface.

Definition in file dataspace.

15.184 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
00037 namespace L4Re
00038 {
00039
00040   // MISSING:
00041   // * size support in map, mapped size in reply
00042
00059 class L4_EXPORT Dataspace :
00060   public L4::Kobject_t<Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE,
00061                        L4::Type_info::Demand_t<1> >
00062 {
00063 public:
00064
00068   enum Map_flags
00069   {
00070     Map_ro = 0,
00071     Map_rw = 1,
00072
00073     Map_normal        = 0x00,
00074     Map_cacheable     = Map_normal,
00075     Map_bufferable    = 0x10,
00076     Map_uncacheable   = 0x20,
00077
00078     Map_caching_mask  = 0x30,
00079     Map_caching_shift = 4,
00080   };
00081
00085   struct Stats {
00086     unsigned long size;
00087     unsigned long flags;
```

```
  };


  long map(l4_addr_t offset, unsigned long flags, l4_addr_t local_addr,
           l4_addr_t min_addr, l4_addr_t max_addr) const throw();

  long map_region(l4_addr_t offset, unsigned long flags,
                  l4_addr_t min_addr, l4_addr_t max_addr) const throw();

  L4_RPC(long, clear, (l4_addr_t offset, unsigned long size));

  L4_RPC(long, allocate, (l4_addr_t offset, l4_size_t size));

  L4_RPC(long, copy_in, (l4_addr_t dst_offs,
L4::Ipc::Cap<Dataspace> src,
                         l4_addr_t src_offs, unsigned long size));


  L4_RPC(long, phys, (l4_addr_t offset, l4_addr_t &phys_addr,
l4_size_t &phys_size));

  unsigned long size() const throw();

  long flags() const throw();

  L4_RPC(long, info, (Stats *stats));

  L4_RPC(long, take, ());

  L4_RPC(long, release, ());

  L4_RPC_NF(long, map, (unsigned long offset, l4_addr_t spot,
                        unsigned long flags, L4::Ipc::Rcv_fpage r,
                        L4::Ipc::Snd_fpage &fp));
private:

  long __map(l4_addr_t offset, unsigned char *size, unsigned long flags,
             l4_addr_t local_addr) const throw();

public:
  typedef L4::Typeid::Rpcs<map_t, clear_t, info_t, copy_in_t, take_t,
                           release_t, phys_t, allocate_t> Rpcs;

};


}

```

15.185 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.

15.185.1 Detailed Description

Dataspace protocol defintion.

Definition in file dataspace-sys.h.

15.186 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, Phys, Allocate };
00035   };
00036 };
00037
```

15.187 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.

15.187.1 Detailed Description

Debug interface.

Definition in file debug.

15.188 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
00063   L4_INLINE_RPC(long, debug, (unsigned long function));
00064   typedef L4::Typeid::Rpc_nocode<debug_t> Rpcs;
00065 };
00066
00067 template<typename BASE>
00068 class Debug_obj_t :
00069   public L4::Kobject_2t<Debug_obj_t<BASE>, BASE, Debug_obj, L4::PROTO_EMPTY>
00070 {
00071   typedef L4::Typeid::Rpcs<> Rpcs;
00072 };
00073 }
```

15.189 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.

15.190 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
00061 class Dma_space :
00062   public L4::Kobject_0t< Dma_space,
00063                          L4RE_PROTO_DMA_SPACE,
00064                          L4::Type_info::Demand_t<1> >
00065 {
00066 public:
00068   typedef l4_uint64_t Dma_addr;
00069
00073   enum Direction
00074   {
00075     Bidirectional,
00076     To_device,
00077     From_device,
00078     None
00079   };
00080
00085   enum Attribute
00086   {
00098     No_sync
00099   };
00100
00106   typedef L4::Types::Flags<Attribute> Attributes;
00107
00113   enum Space_attrib
00114   {
00121     Coherent,
00122
00127     Phys_space
00128   };
00129
00131   typedef L4::Types::Flags<Space_attrib>
      Space_attribs;
00132
00157   L4_INLINE_RPC(
00158       long, map, (L4::Ipc::Cap<L4Re::Dataspace> src,
      l4_addr_t offset,
00159                   L4::Ipc::In_out<l4_size_t *> size,
00160                   Attributes attrs, Direction dir,
00161                   Dma_addr *dma_addr));
00162
00171   L4_INLINE_RPC(
00172       long, unmap, (Dma_addr dma_addr,
00173                     l4_size_t size, Attributes attrs, Direction dir));
00174
00186   L4_INLINE_RPC(
00187       long, associate, (L4::Ipc::Opt<L4::Ipc::Cap<L4::Task> >
      dma_task,
00188                         Space_attribs attr),
00189       L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00190
00196   L4_INLINE_RPC(
```

```
00197       long, disassociate, (),
00198       L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00199
00200   typedef L4::Typeid::Rpcs<map_t, unmap_t, associate_t, disassociate_t>
      Rpcs;
00201 };
00202
00203 }
00204
```

15.191 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_EL↩F_AUX_T_STACK_ADDR,
  L4RE_ELF_AUX_T_KIP_ADDR }

15.191.1 Detailed Description

Auxiliary information for binaries.

Definition in file elf_aux.h.

15.192 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
00039
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
00105 {
00106   l4_umword_t type;
00107   l4_umword_t length;
00108 } l4re_elf_aux_t;
00109
00113 typedef struct l4re_elf_aux_vma_t
00114 {
00115   l4_umword_t type;
00116   l4_umword_t length;
00117   l4_umword_t start;
00118   l4_umword_t end;
00119 } l4re_elf_aux_vma_t;
00120
00124 typedef struct l4re_elf_aux_mword_t
00125 {
00126   l4_umword_t type;
00127   l4_umword_t length;
00128   l4_umword_t value;
00129 } l4re_elf_aux_mword_t;
00130
```

15.193 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.

15.193.1 Detailed Description

Environment interface.

Definition in file env.

15.194 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 {
00082   class L4_EXPORT Env
00083   {
00084   private:
00085     l4re_env_t _env;
00086   public:
00087
00091     typedef l4re_env_cap_entry_t Cap_entry;
00092
00100     static Env const *env() throw()
```

```
    { return reinterpret_cast<Env*>(l4re_global_env); }

    L4::Cap<Parent> parent() const throw()
    { return L4::Cap<Parent>(_env.parent); }
    L4::Cap<Mem_alloc> mem_alloc() const throw()
    { return L4::Cap<Mem_alloc>(_env.mem_alloc); }
    L4::Cap<L4::Factory> user_factory() const throw()
    { return L4::Cap<L4::Factory>(_env.mem_alloc); }
    L4::Cap<Rm> rm() const throw()
    { return L4::Cap<Rm>(_env.rm); }
    L4::Cap<Log> log() const throw()
    { return L4::Cap<Log>(_env.log); }
    L4::Cap<L4::Thread> main_thread() const throw()
    { return L4::Cap<L4::Thread>(_env.main_thread); }
    L4::Cap<L4::Task> task() const throw()
    { return L4::Cap<L4::Task>(L4RE_THIS_TASK_CAP); }
    L4::Cap<L4::Factory> factory() const throw()
    { return L4::Cap<L4::Factory>(_env.factory); }
    l4_cap_idx_t first_free_cap() const throw()
    { return _env.first_free_cap; }
    l4_fpage_t utcb_area() const throw()
    { return _env.utcb_area; }
    l4_addr_t first_free_utcb() const throw()
    { return _env.first_free_utcb; }

    Cap_entry const *initial_caps() const throw()
    { return _env.caps; }

    Cap_entry const *get(char const *name, unsigned l) const throw()
    { return l4re_env_get_cap_l(name, l, &_env); }

    template< typename T >
    L4::Cap<T> get_cap(char const *name, unsigned l) const throw()
    {
      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 throw()
    { return get_cap<T>(name, __builtin_strlen(name)); }

    void parent(L4::Cap<Parent> const &c) throw()
    { _env.parent = c.cap(); }
    void mem_alloc(L4::Cap<Mem_alloc> const &c) throw()
    { _env.mem_alloc = c.cap(); }
    void rm(L4::Cap<Rm> const &c) throw()
    { _env.rm = c.cap(); }
    void log(L4::Cap<Log> const &c) throw()
    { _env.log = c.cap(); }
    void main_thread(L4::Cap<L4::Thread> const &c) throw()
    { _env.main_thread = c.cap(); }
    void factory(L4::Cap<L4::Factory> const &c) throw()
    { _env.factory = c.cap(); }
    void first_free_cap(l4_cap_idx_t c) throw()
    { _env.first_free_cap = c; }
    void utcb_area(l4_fpage_t utcbs) throw()
    { _env.utcb_area = utcbs; }
    void first_free_utcb(l4_addr_t u) throw()
    { _env.first_free_utcb = u; }

    L4::Cap<L4::Scheduler> scheduler() const throw()
    { return L4::Cap<L4::Scheduler>(_env.scheduler); }

    void scheduler(L4::Cap<L4::Scheduler> const &c) throw()
    { _env.scheduler = c.cap(); }

    void initial_caps(Cap_entry *first) throw()
    { _env.caps = first; }
  };
};
```

15.195 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/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.

15.195.1 Detailed Description

Environment interface.

Definition in file env.h.

15.195.2 Typedef Documentation

15.195.2.1 l4re_env_t

```
typedef struct l4re_env_t l4re_env_t
```

Initial environment data structure.

See also

Initial environment

15.196 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  */
```

```
#pragma once

#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.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() : cap(L4_INVALID_CAP), flags(~0) {}

  l4re_env_cap_entry_t(char const *n, l4_cap_idx_t c, l4_umword_t f = 0)
  : 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)
  {
    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()); }
```

15.197 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

- 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.

15.197.1 Detailed Description

Error helper.

Definition in file error_helper.

15.198 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 namespace Priv {
00036 inline long __attribute__((__noreturn__)) __runtime_error(long err, char const *extra);
00037
00038 inline long __runtime_error(long err, char const *extra)
00039 {
00040   switch (err)
00041     {
00042     case -L4_ENOENT: throw (L4::Element_not_found(extra));
00043     case -L4_ENOMEM: throw (L4::Out_of_memory(extra));
00044     case -L4_EEXIST: throw (L4::Element_already_exists(extra));
00045     case -L4_ERANGE: throw (L4::Bounds_error(extra));
00046     default: throw (L4::Runtime_error(err, extra));
00047     }
00048 }
00049
00050 }
```

```
00051
00062 inline
00063 long chksys(long err, char const *extra = "", long ret = 0)
00064 {
00065   if (L4_UNLIKELY(err < 0))
00066     Priv::__runtime_error(ret ? ret : err, extra);
00067
00068   return err;
00069 }
00070
00083 inline
00084 long chksys(l4_msgtag_t const &t, char const *extra = "",
00085             l4_utcb_t *utcb = l4_utcb(), long ret = 0)
00086 {
00087   if (L4_UNLIKELY(t.has_error()))
00088     Priv::__runtime_error(ret ? ret : l4_error_u(t, utcb), extra);
00089   else if (L4_UNLIKELY(t.label() < 0))
00090     throw L4::Runtime_error(ret ? ret: t.label(), extra);
00091
00092   return t.label();
00093 }
00094
00106 inline
00107 long chksys(l4_msgtag_t const &t, l4_utcb_t *utcb, char const *extra = "")
00108 { return chksys(t, extra, utcb); }
00109
00110 #if 0
00111 inline
00112 long chksys(long ret, long err, char const *extra = "")
00113 {
00114   if (L4_UNLIKELY(ret < 0))
00115     Priv::__runtime_error(err, extra);
00116
00117   return ret;
00118 }
00119 #endif
00120
00137 template<typename T>
00138 inline
00139 #if __cplusplus >= 201103L
00140 T chkcap(T &&cap, char const *extra = "", long err = -L4_ENOMEM)
00141 #else
00142 T chkcap(T cap, char const *extra = "", long err = -L4_ENOMEM)
00143 #endif
00144 {
00145   if (L4_UNLIKELY(!cap.is_valid()))
00146     Priv::__runtime_error(err ? err : cap.cap(), extra);
00147
00148 #if __cplusplus >= 201103L
00149   return cxx::forward<T>(cap);
00150 #else
00151   return cap;
00152 #endif
00153 }
00154
00169 inline
00170 l4_msgtag_t
00171 chkipc(l4_msgtag_t tag, char const *extra = "",
00172        l4_utcb_t *utcb = l4_utcb())
00173 {
00174   if (L4_UNLIKELY(tag.has_error()))
00175     chksys(l4_error_u(tag, utcb), extra);
00176
00177   return tag;
00178 }
00179 #endif
00180
00181 }
```

15.199 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++.

15.200 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>
```

```
#include <l4/re/util/event_buffer>
#include <l4/sys/factory>
#include <l4/cxx/type_traits>

namespace L4Re { namespace Util {

template< typename PAYLOAD >
class Event_t
{
public:
  enum Mode
  {
    Mode_irq,
    Mode_polling,
  };

  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))
  {
    Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<
L4Re::Dataspace>());
    if (!ev_ds.is_valid())
      return -L4_ENOMEM;

    int r;

    Unique_del_cap<IRQ_TYPE> ev_irq(ca->alloc<IRQ_TYPE>());
    if (!ev_irq.is_valid())
      return -L4_ENOMEM;

    if ((r = l4_error(env->factory()->create(ev_irq.get())))
      return r;

    if ((r = l4_error(event->bind(0, ev_irq.get())))
      return r;

    if ((r = event->get_buffer(ev_ds.get())))
      return r;

    long sz = ev_ds->size();
    if (sz < 0)
      return sz;

    Rm::Unique_region<void*> buf;

    if ((r = env->rm()->attach(&buf, sz, L4Re::Rm::Search_addr,
                               L4::Ipc::make_cap_rw(ev_ds.get())))
      return r;

    _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
    _ev_ds     = cxx::move(ev_ds);
    _ev_irq    = cxx::move(ev_irq);
    _buf       = cxx::move(buf);

    return 0;
  }

  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))
  {
    Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<
L4Re::Dataspace>());
    if (!ev_ds.is_valid())
      return -L4_ENOMEM;

    int r;

    if ((r = event->get_buffer(ev_ds.get())))
      return r;

    long sz = ev_ds->size();
    if (sz < 0)
      return sz;

    Rm::Unique_region<void*> buf;

    if ((r = env->rm()->attach(&buf, sz, L4Re::Rm::Search_addr,
                               L4::Ipc::make_cap_rw(ev_ds.get())))
      return r;

    _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
```

```
    _ev_ds      = cxx::move(ev_ds);
    _buf        = cxx::move(buf);

    return 0;
  }

  L4Re::Event_buffer_t<PAYLOAD> &buffer() { return _ev_buffer; }

  L4::Cap<L4::Triggerable> irq() const { return _ev_irq.get(); }

private:
  Unique_cap<L4Re::Dataspace> _ev_ds;
  Unique_del_cap<L4::Triggerable> _ev_irq;
  L4Re::Event_buffer_t<PAYLOAD> _ev_buffer;
  Rm::Unique_region<void*> _buf;
};

typedef Event_t<Default_event_payload> Event;

}}
```

15.201 l4/re/impl/dataspace_impl.h File Reference

Dataspace client stub implementation.

```
#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
```

Include dependency graph for dataspace_impl.h:

Namespaces

- L4Re

  L4Re C++ Interfaces.

15.201.1 Detailed Description

Dataspace client stub implementation.

Definition in file dataspace_impl.h.

15.202 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
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/dataspace>
00024 #include <l4/sys/cxx/ipc_client>
00025
00026 L4_RPC_DEF(L4Re::Dataspace::clear);
00027 L4_RPC_DEF(L4Re::Dataspace::allocate);
00028 L4_RPC_DEF(L4Re::Dataspace::copy_in);
00029 L4_RPC_DEF(L4Re::Dataspace::phys);
00030 L4_RPC_DEF(L4Re::Dataspace::info);
00031 L4_RPC_DEF(L4Re::Dataspace::take);
00032 L4_RPC_DEF(L4Re::Dataspace::release);
00033
00034 namespace L4Re {
00035
00036 long
00037 Dataspace::__map(unsigned long offset, unsigned char *size, unsigned long flags,
00038                  l4_addr_t local_addr) const throw()
00039 {
00040   l4_addr_t spot = local_addr & ~(~0UL << l4_umword_t(*size));
00041   l4_addr_t base = local_addr & (~0UL << l4_umword_t(*size));
00042   L4::Ipc::Rcv_fpage r;
00043   r = L4::Ipc::Rcv_fpage::mem(base, *size, 0);
00044
00045   L4::Ipc::Snd_fpage fp;
00046   long err = map_t::call(c(), offset, spot, flags, r, fp, l4_utcb());
00047   if (L4_UNLIKELY(err < 0))
00048     return err;
00049
00050   *size = fp.rcv_order();
00051   return err;
00052 }
00053
00054 long
00055 Dataspace::map_region(l4_addr_t offset, unsigned long flags,
00056                       l4_addr_t min_addr, l4_addr_t max_addr) const throw()
00057 {
00058   min_addr   = l4_trunc_page(min_addr);
00059   max_addr   = l4_round_page(max_addr);
00060   unsigned char order = L4_LOG2_PAGESIZE;
00061
00062   long err = 0;
00063
00064   while (min_addr < max_addr)
00065     {
00066       unsigned char order_mapped;
00067       order_mapped = order
00068         = l4_fpage_max_order(order, min_addr, min_addr, max_addr, min_addr);
00069       err = __map(offset, &order_mapped, flags, min_addr);
00070       if (L4_UNLIKELY(err < 0))
00071   return err;
00072
00073       if (order > order_mapped)
00074   order = order_mapped;
```

```
00075
00076       min_addr += 1UL << order;
00077       offset   += 1UL << order;
00078
00079       if (min_addr >= max_addr)
00080   return 0;
00081
00082       while (min_addr != l4_trunc_size(min_addr, order)
00083              || max_addr < l4_round_size(min_addr + 1,order))
00084   --order;
00085     }
00086
00087   return 0;
00088 }
00089
00090
00091 long
00092 Dataspace::map(l4_addr_t offset, unsigned long flags,
00093                l4_addr_t local_addr,
00094                l4_addr_t min_addr, l4_addr_t max_addr) const throw()
00095 {
00096   min_addr   = l4_trunc_page(min_addr);
00097   max_addr   = l4_round_page(max_addr);
00098   local_addr = l4_trunc_page(local_addr);
00099   unsigned char order
00100     = l4_fpage_max_order(L4_LOG2_PAGESIZE, local_addr, min_addr, max_addr
      , local_addr);
00101
00102   return __map(offset, &order, flags, local_addr);
00103 }
00104
00105 unsigned long
00106 Dataspace::size() const throw()
00107 {
00108   Stats stats = Stats();
00109   int err = info(&stats);
00110   if (err < 0)
00111     return 0;
00112   return stats.size;
00113 }
00114
00115 long
00116 Dataspace::flags() const throw()
00117 {
00118   Stats stats = Stats();
00119   int err = info(&stats);
00120   if (err < 0)
00121     return err;
00122   return stats.flags;
00123 }
00124
00125 };
```

15.203 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.

15.203.1 Detailed Description

Memory allocator client stub implementation.

Definition in file mem_alloc_impl.h.

15.204 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 throw()
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 long
00047 Mem_alloc::free(L4::Cap<Dataspace>) const throw()
00048 {
00049   return 0;
00050 }
00051
00052 };
```

15.205 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.

15.205.1 Detailed Description

Namespace client stub implementation.

Definition in file namespace_impl.h.

15.206 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  */
00023 #include <l4/re/namespace>
00024
00025 #include <l4/util/util.h>
00026 #include <l4/sys/cxx/ipc_client>
00027 #include <l4/sys/assert.h>
00028
00029 #include <cstring>
00030
00031 L4_RPC_DEF(L4Re::Namespace::query);
00032 L4_RPC_DEF(L4Re::Namespace::register_obj);
00033 L4_RPC_DEF(L4Re::Namespace::unlink);
00034
00035 namespace L4Re {
00036
00037 long
00038 Namespace::_query(char const *name, unsigned len,
00039                   L4::Cap<void> const &target,
00040                   l4_umword_t *local_id, bool iterate) const throw()
00041 {
00042   l4_assert(target.is_valid());
00043
00044   L4::Cap<Namespace> ns = c();
00045   L4::Ipc::Array<char const, unsigned long> _name(len, name);
00046
00047   while (_name.length > 0)
00048     {
00049       L4::Ipc::Snd_fpage cap;
00050       L4::Opcode dummy;
00051       int err = query_t::call(ns, _name,
00052                               L4::Ipc::Small_buf(target.cap(),
00053                                                  local_id
00054                                                    ? L4_RCV_ITEM_LOCAL_ID
00055                                                    : 0),
00056                               cap, dummy, _name);
00057       if (err < 0)
00058         return err;
00059
00060       bool const partly = err & Partly_resolved;
00061       if (cap.id_received())
00062         {
00063           *local_id = cap.data();
00064           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 throw()
{
  if (L4_UNLIKELY(len == 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 < 0 && (ret != -L4_EAGAIN)))
        return ret;

      if (rem == to)
        return ret;

      l4_sleep(to);

      if (rem > 0)
        {
          rem -= to;
          if (to > rem)
            to = rem;
        }

      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 throw()
{
  return query(name, __builtin_strlen(name), target,
               timeout, local_id, iterate);
}

}
```

15.207 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.

15.207.1 Detailed Description

Region map client stub implementation.

Definition in file rm_impl.h.

15.208 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
```

```
L4_RPC_DEF(L4Re::Rm::reserve_area);
L4_RPC_DEF(L4Re::Rm::free_area);
L4_RPC_DEF(L4Re::Rm::attach);
L4_RPC_DEF(L4Re::Rm::detach);
L4_RPC_DEF(L4Re::Rm::get_regions);
L4_RPC_DEF(L4Re::Rm::get_areas);
L4_RPC_DEF(L4Re::Rm::find);

namespace L4Re
{

long
Rm::attach(l4_addr_t *start, unsigned long size, unsigned long flags,
           L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs,
           unsigned char align) const throw()
{
  if (flags & Reserved)
    mem = L4::Ipc::Cap<L4Re::Dataspace>();

  long e = attach_t::call(c(), start, size, flags, mem, offs, align, mem.cap().cap());
  if (e < 0)
    return e;

  if (flags & Eager_map)
    {
      unsigned long fl = (flags & Read_only)
        ? Dataspace::Map_ro
        : Dataspace::Map_rw;
      fl |= (flags & Caching) >> Caching_ds_shift;
      e = mem.cap()->map_region(offs, fl, *start, *start + size);
    }
  return e;
}

int
Rm::detach(l4_addr_t start, unsigned long size,
L4::Cap<Dataspace> *mem,
           L4::Cap<L4::Task> task, unsigned flags) const throw()
{
  l4_addr_t rstart = 0, rsize = 0;
  l4_cap_idx_t mem_cap = L4_INVALID_CAP;
  long e = detach_t::call(c(), start, size, flags, rstart, rsize, mem_cap);
  if (L4_UNLIKELY(e < 0))
    return e;

  if (mem)
    *mem = L4::Cap<L4Re::Dataspace>(mem_cap);

  if (!task.is_valid())
    return e;

  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;
}
}
```

15.209 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.

15.209.1 Detailed Description

Auxiliary definitions.

Definition in file l4aux.h.

15.210 l4aux.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.
 */

#include <l4/sys/types.h>

enum l4re_aux_ldr_flags_t
{
  L4RE_AUX_LDR_FLAG_EAGER_MAP    = 0x1,
  L4RE_AUX_LDR_FLAG_ALL_SEGS_COW = 0x2,
  L4RE_AUX_LDR_FLAG_PINNED_SEGS  = 0x4,
};

typedef struct l4re_aux_t
{
  char const *    binary;
  l4_cap_idx_t    kip_ds;
  l4_umword_t     dbg_lvl;
  l4_umword_t     ldr_flags;
} l4re_aux_t;

```

15.211 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.

15.211.1 Detailed Description

Log interface.

Definition in file log.

15.212 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 throw();
00055
00061   void print(char const *string) const throw();
00062 };
00063 }
```

15.213 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.

15.213.1 Detailed Description

Log protocol definition.

Definition in file log-sys.h.

15.214 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 };
```

15.215 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.

15.215.1 Detailed Description

Memory allocator interface.

Definition in file mem_alloc.

15.216 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
00040 // * shall we support superpages in noncont memory?
00041
00061 class L4_EXPORT Mem_alloc :
00062   public L4::Kobject_t<Mem_alloc, L4::Factory, L4::PROTO_EMPTY>
00063 {
00064 public:
00071   enum Mem_alloc_flags
00072   {
00073     Continuous   = 0x01,
00074     Pinned       = 0x02,
00075     Super_pages  = 0x04,
00076   };
00077
00100   long alloc(long size, L4::Cap<Dataspace> mem,
00101              unsigned long flags = 0, unsigned long align = 0) const throw();
00102
00115   /* Deprecation message added Q4 2016 */
00116   long free(L4::Cap<Dataspace> mem) const throw()
00117   L4_DEPRECATED("This function is an empty stub and remains for backward compatibility only.
      Check documentation for details.");
00118
00119 };
00120
00121 };
```

15.217 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.

15.217.1 Detailed Description

Memory allocator protocol definitions.

Definition in file mem_alloc-sys.h.

15.218 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 };
```

15.219 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.

15.219.1 Detailed Description

Namespace interface.

Definition in file namespace.

15.220 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
```

```
  enum Query_result_flags
  {
    Partly_resolved = 0x020,
  };

  enum Query_timeout
  {
    To_default      = 3600000,
    To_non_blocking = 0,
    To_forever      = -1,
  };

  L4_RPC_NF(
      long, query, (L4::Ipc::Array_ref<char const, unsigned long> name,
                    L4::Ipc::Small_buf cap,
                    L4::Ipc::Snd_fpage &snd_cap, L4::Ipc::Opt<L4::Opcode &> dummy,
                    L4::Ipc::Opt<L4::Ipc::Array_ref<char const, unsigned long> &> out_name));

  long query(char const *name, L4::Cap<void> const &cap,
             int timeout = To_default,
             l4_umword_t *local_id = 0, bool iterate = true) const throw();

  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 throw();

  L4_RPC_NF(long, register_obj, (unsigned flags,
                                 L4::Ipc::Array<char const, unsigned long> name,
                                 L4::Ipc::Opt< L4::Ipc::Cap<void> > obj),
            L4::Ipc::Call_t<L4_CAP_FPAGE_W>);

  long register_obj(char const *name, L4::Ipc::Cap<void> obj,
                    unsigned flags = Rw) const throw()
  {
    return register_obj_t::call(c(), flags,
                                L4::Ipc::Array<char const, unsigned long> (
                                  __builtin_strlen(name), name),
                                obj);
  }

  L4_RPC_NF_OP(3, // backward compatibility opcode
               long, unlink, (L4::Ipc::Array<char const, unsigned long> name),
               L4::Ipc::Call_t<L4_CAP_FPAGE_W>);

  long unlink(char const* name)
  {
    return unlink_t::call(c(), L4::Ipc::Array<char const, unsigned long> (
                                  __builtin_strlen(name), name));
  }

  typedef L4::Typeid::Rpcs<query_t, register_obj_t, unlink_t> Rpcs;

private:
  long _query(char const *name, unsigned len,
              L4::Cap<void> const &target, l4_umword_t *local_id,
              bool iterate) const throw();

};

};
```

15.221 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.

15.221.1 Detailed Description

Namespace protocol definitions.

Definition in file namespace-sys.h.

15.222 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 };
```

15.223 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.

15.223.1 Detailed Description

Parent interface.

Definition in file parent.

15.224 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
00051 class L4_EXPORT Parent :
00052   public L4::Kobject_t<Parent, L4::Kobject, L4RE_PROTO_PARENT>
00053 {
00054 public:
00064   L4_INLINE_RPC(long, signal, (unsigned long sig, unsigned long val));
00065   typedef L4::Typeid::Rpcs<signal_t> Rpcs;
00066 };
00067 };
00068
```

15.225 l4/re/parent-sys.h File Reference

Parent protocol definition.

Namespaces

- L4Re

  L4Re C++ Interfaces.

Enumerations

- enum L4Re::Parent_::Opcodes

  Parent communication-protocol opcodes.

15.225.1 Detailed Description

Parent protocol definition.

Definition in file parent-sys.h.

15.226 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 };
```

15.227 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 }

15.227.1 Detailed Description

L4Re Protocol Constants (C version)

Definition in file protocols.h.

15.227.2 Enumeration Type Documentation

15.227.2.1 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.

15.228 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
```

15.229 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/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.

Namespaces

- L4Re

  L4Re C++ Interfaces.

15.229.1 Detailed Description

Region mapper interface.

Definition in file rm.

15.230 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/re/consts>
00037 #include <l4/re/dataspace>
00038
00039 namespace L4Re {
00040
00069 class L4_EXPORT Rm :
00070   public L4::Kobject_t<Rm, L4::Pager, L4RE_PROTO_RM,
00071                        L4::Type_info::Demand_t<1> >
00072 {
00073 public:
00075   enum Detach_result
00076   {
00077     Detached_ds  = 0,
00078     Kept_ds      = 1,
00079     Split_ds     = 2,
00080     Detach_result_mask = 3,
00081
00082     Detach_again = 4,
00083   };
00084
00086   enum Region_flags
00087   {
00088     Read_only        = 0x01,
00089     Detach_free      = 0x02,
00091     Pager            = 0x04,
00092     Reserved         = 0x08,
00093
00095     Caching_shift    = 8,
00097     Caching_ds_shift = Caching_shift - Dataspace::Map_caching_shift,
00099     Caching          = Dataspace::Map_caching_mask << Caching_ds_shift,
```

```
00101     Cache_normal      = Dataspace::Map_normal << Caching_ds_shift,
00103     Cache_buffered    = Dataspace::Map_bufferable << Caching_ds_shift,
00105     Cache_uncached    = Dataspace::Map_uncacheable << Caching_ds_shift,
00106
00107     Region_flags      = Caching | 0x0f,
00108   };
00109
00111   enum Attach_flags
00112   {
00113     Search_addr         = 0x20,
00114     In_area             = 0x40,
00115     Eager_map           = 0x80,
00116
00117     Attach_flags        = 0xf0,
00118   };
00119
00121   enum Detach_flags
00122   {
00132     Detach_exact    = 1,
00142     Detach_overlap = 2,
00143
00151     Detach_keep = 4,
00152   };
00153
00154
00155   template< typename T >
00156   class Auto_region
00157   {
00158   private:
00159     T _addr;
00160     mutable L4::Cap<Rm> _rm;
00161
00162   public:
00163     Auto_region() throw()
00164     : _addr(0), _rm(L4::Cap<Rm>::Invalid) {}
00165
00166     explicit Auto_region(T addr) throw()
00167     : _addr(addr), _rm(L4::Cap<Rm>::Invalid) {}
00168
00169     Auto_region(T addr, L4::Cap<Rm> const &rm) throw()
00170     : _addr(addr), _rm(rm) {}
00171
00172     Auto_region(Auto_region const &o) throw() : _addr(o.get()), _rm(o._rm)
00173     { o.release(); }
00174
00175     Auto_region &operator = (Auto_region const &o) throw()
00176     {
00177       if (&o != this)
00178         {
00179     if (_rm.is_valid())
00180       _rm->detach(l4_addr_t(_addr), 0);
00181     _rm = o._rm;
00182     _addr = o.release();
00183         }
00184       return *this;
00185     }
00186
00187     ~Auto_region() throw()
00188     {
00189       if (_rm.is_valid())
00190   _rm->detach(l4_addr_t(_addr), 0);
00191     }
00192
00193     T get() const throw() { return _addr; }
00194     T release() const throw() { _rm = L4::Cap<Rm>::Invalid; return _addr; }
00195     void reset(T addr, L4::Cap<Rm> const &rm) throw()
00196     {
00197       if (_rm.is_valid())
00198   _rm->detach(l4_addr_t(_addr), 0);
00199
00200       _rm = rm;
00201       _addr = addr;
00202     }
00203
00204     void reset() throw()
00205     { reset(0, L4::Cap<Rm>::Invalid); }
00206
00208     T operator * () const throw() { return _addr; }
00209
00211     T operator -> () const throw() { return _addr; }
00212
00213   } L4_DEPRECATED("use L4Re::Rm::Unique_region");
00214
00239   long reserve_area(l4_addr_t *start, unsigned long size,
00240                     unsigned flags = 0,
00241                     unsigned char align = L4_PAGESHIFT) const throw()
00242   { return reserve_area_t::call(c(), start, size, flags, align); }
```

```
00243
00244   L4_RPC_NF(long, reserve_area, (L4::Ipc::In_out<l4_addr_t *> start,
00245                                  unsigned long size,
00246                                  unsigned flags,
00247                                  unsigned char align));
00248
00264   template< typename T >
00265   long reserve_area(T **start, unsigned long size,
00266                     unsigned flags = 0,
00267                     unsigned char align = L4_PAGESHIFT) const throw()
00268   { return reserve_area_t::call(c(), (l4_addr_t*)start, size, flags, align); }
00269
00282   L4_RPC(long, free_area, (l4_addr_t addr));
00283
00284   L4_RPC_NF(long, attach, (L4::Ipc::In_out<l4_addr_t *> start,
00285                            unsigned long size, unsigned long flags,
00286                            L4::Ipc::Opt<L4::Ipc::Cap<Dataspace> > mem,
00287                            l4_addr_t offs, unsigned char align,
00288                            L4::Ipc::Opt<l4_cap_idx_t> client_cap));
00289
00290   L4_RPC_NF(long, detach, (l4_addr_t addr, unsigned long size, unsigned flags,
00291                            l4_addr_t &start, l4_addr_t &rsize,
00292                            l4_cap_idx_t &mem_cap));
00293
00337   long attach(l4_addr_t *start, unsigned long size, unsigned long flags,
00338               L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs = 0,
00339               unsigned char align = L4_PAGESHIFT) const throw();
00340
00344   template< typename T >
00345   long attach(T **start, unsigned long size, unsigned long flags,
00346               L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs = 0,
00347               unsigned char align = L4_PAGESHIFT) const throw()
00348   {
00349     union X { l4_addr_t a; T* t; };
00350     X *x = reinterpret_cast<X*>(start);
00351     return attach(&x->a, size, flags, mem, offs, align);
00352   }
00353
00354 #pragma GCC diagnostic push
00355 #pragma GCC diagnostic ignored "-Wdeprecated-declarations"
00356   template< typename T >
00357   long attach(Auto_region<T> *start, unsigned long size, unsigned long flags,
00358               L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs = 0,
00359               unsigned char align = L4_PAGESHIFT) const throw()
00360   {
00361     l4_addr_t addr = (l4_addr_t)start->get();
00362
00363     long res = attach(&addr, size, flags, mem, offs, align);
00364     if (res < 0)
00365       return res;
00366
00367     start->reset((T)addr, L4::Cap<Rm>(cap()));
00368     return res;
00369   }
00370 #pragma GCC diagnostic pop
00371
00372 #if __cplusplus >= 201103L
00373   template< typename T >
00374   class Unique_region
00375   {
00376   private:
00377     T _addr;
00378     L4::Cap<Rm> _rm;
00379
00380   public:
00381     Unique_region(Unique_region const &) = delete;
00382     Unique_region &operator = (Unique_region const &) = delete;
00383
00384     Unique_region() noexcept
00385     : _addr(0), _rm(L4::Cap<Rm>::Invalid) {}
00386
00387     explicit Unique_region(T addr) noexcept
00388     : _addr(addr), _rm(L4::Cap<Rm>::Invalid) {}
00389
00390     Unique_region(T addr, L4::Cap<Rm> const &rm) noexcept
00391     : _addr(addr), _rm(rm) {}
00392
00393     Unique_region(Unique_region &&o) noexcept : _addr(o.get()), _rm(o._rm)
00394     { o.release(); }
00395
00396     Unique_region &operator = (Unique_region &&o) noexcept
00397     {
00398       if (&o != this)
00399         {
00400           if (_rm.is_valid())
00401             _rm->detach(l4_addr_t(_addr), 0);
00402           _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, unsigned long flags,
              L4::Ipc::Cap<Dataspace> mem, l4_addr_t 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 throw();

  int detach(void *addr, L4::Cap<Dataspace> *mem,
             L4::Cap<L4::Task> const &task = This_task) const throw();

  int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
             L4::Cap<L4::Task> const &task) const throw();

  int find(l4_addr_t *addr, unsigned long *size, l4_addr_t *offset,
           unsigned *flags, L4::Cap<Dataspace> *m) throw()
  { 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,
                        unsigned *flags, l4_addr_t *offset,
                        L4::Ipc::As_value<L4::Cap<Dataspace> > *m));

  struct Region
  {
    l4_addr_t start;
    l4_addr_t end;
    l4_addr_t offset;
    L4::Cap<Dataspace> ds;
  };

  struct Area
  {
    l4_addr_t start;
    l4_addr_t end;
  };
```

```
  L4_RPC(long, get_regions, (l4_addr_t start,
L4::Ipc::Ret_array<Region> regions));
  L4_RPC(long, get_areas, (l4_addr_t start,
L4::Ipc::Ret_array<Area> areas));

  int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
             L4::Cap<L4::Task> task, unsigned flags) const throw();

  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 throw()
{  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 throw()
{  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 throw()
{  return detach(addr, size, mem, task, Detach_exact); }

};

```

15.231 l4/re/rm-sys.h File Reference

Region mapper protocol definitions.

This graph shows which files directly or indirectly include this file:

Namespaces

- L4Re

  L4Re C++ Interfaces.

Enumerations

- enum L4Re::Rm_::Opcodes

  Region-map communication-protocol opcodes.

15.231.1 Detailed Description

Region mapper protocol definitions.

Definition in file rm-sys.h.

15.232 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 };
```

15.233 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.

15.233.1 Detailed Description

Shared_cap / Shared_del_cap.

Definition in file shared_cap.

15.234 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>>;
00046 template< typename T >
00047 using shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>>;
00048
00058 template< typename T >
00059 Shared_cap<T>
00060 make_shared_cap(L4Re::Cap_alloc *ca)
00061 { return Shared_cap<T>(ca->alloc<T>(), ca); }
00062
00079 template< typename T >
00080 using Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>;
```

```
00082 template<typename T>
00083 using shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>;
00084
00094 template< typename T >
00095 Shared_del_cap<T>
00096 make_shared_del_cap(L4Re::Cap_alloc *ca)
00097 { return Shared_del_cap<T>(ca->alloc<T>(), ca); }
00098
00099 } // namespace L4Re
```

15.235 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:

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.

15.235.1 Detailed Description

Shared_cap / Shared_del_cap.

Definition in file shared_cap.

15.236 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
```

15.237 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.

15.237.1 Detailed Description

Unique_cap / Unique_del_cap.

Definition in file unique_cap.

15.238 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/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 }
```

15.239 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.

15.239.1 Detailed Description

Unique_cap / Unique_del_cap.

Definition in file unique_cap.

15.240 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_lx>
```

```
namespace L4Re { namespace Util {

template< typename T >
using Unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>;
template< typename T >
using unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>;

template< typename T >
Unique_cap<T>
make_unique_cap()
{ return Unique_cap<T>(cap_alloc.alloc<T>()); }

template< typename T >
using Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>;
template< typename T >
using unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>;

template< typename T >
Unique_del_cap<T>
make_unique_del_cap()
{ return Unique_del_cap<T>(cap_alloc.alloc<T>()); }

}}
```

15.241 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++.

15.241.1 Detailed Description

Bitmap capability allocator.

Definition in file bitmap_cap_alloc.

15.242 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     throw() : _bias(bias), _items(max, mem) {}
00048
00049   L4::Cap<void> alloc() throw()
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   }
```

```
  long hint() const { return _items.hint(); }

  template< typename T >
  L4::Cap<T> alloc() throw()
  { return L4::Cap<T>(alloc().cap()); }

  State is_allocated(L4::Cap<void> c) const throw()
  {
    long idx = (c.cap() >> L4_CAP_SHIFT);

    if (idx < _bias)
      return Unknown;

    idx -= _bias;
    return _items.is_allocated(idx) ? Allocated : Free;
  }

  template< typename T>
  void free(L4::Cap<T> const &cap, l4_cap_idx_t task = -1UL,
            l4_umword_t unmap_flags = L4_FP_ALL_SPACES) throw()
  {
    long idx = (cap.cap() >> L4_CAP_SHIFT);
    if (idx < _bias)
      return;

    idx -= _bias;

    _items.free(idx);

    if (task != -1UL)
      l4_task_unmap(task, cap.fpage(), unmap_flags | 2);
  }

  // since we have no counters assume counter always > 0
  void take(L4::Cap<void>) throw() {}
  bool release(L4::Cap<void>, l4_cap_idx_t task = L4_INVALID_CAP,
               unsigned unmap_flags = L4_FP_ALL_SPACES) throw()
  { (void)task; (void)unmap_flags; return false; }

  long last() throw()
  {
    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) throw()
    : Cap_alloc_base(Size, _bits, bias) {}

};

}
}
```

15.243 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++.

15.243.1 Detailed Description

Capability utility functions.

Definition in file cap.

15.244 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 }}
```

15.245 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++.

15.245.1 Detailed Description

Capability allocator implementation.

Definition in file cap_alloc_impl.h.

15.246 cap_alloc_impl.h

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
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 //#define L4RE_STATIC_CAP_ALLOC
00028 #if defined(L4RE_STATIC_CAP_ALLOC)
00029
00030 #include <l4/re/util/bitmap_cap_alloc>
00031
00032 namespace L4Re { namespace Util {
00033
00034 typedef Cap_alloc_base _Cap_alloc;
00035
```

```
}}

#else
#include <l4/re/util/counting_cap_alloc>

namespace L4Re { namespace Util {

typedef Counting_cap_alloc<L4Re::Util::Counter<unsigned char> > _Cap_alloc;

}}
#endif
```

15.247 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++.

15.247.1 Detailed Description

Reference-counting capability allocator.

Definition in file counting_cap_alloc.

15.248 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
```

```
{
private:
  void operator = (Counting_cap_alloc const &) { }
  typedef COUNTERTYPE Counter;

  COUNTERTYPE *_items;
  long _free_hint;
  long _bias;
  long _capacity;


public:

  template <unsigned COUNT>
  struct Counter_storage
  {
    COUNTERTYPE _buf[COUNT];
    typedef COUNTERTYPE Buf_type[COUNT];
    enum { Size = COUNT };
  };

protected:

  Counting_cap_alloc() throw()
  : _items(0), _free_hint(0), _bias(0), _capacity(0)
  {}

  void setup(void *m, long capacity, long bias) throw()
  {
    _items = (Counter*)m;
    _capacity = capacity;
    _bias = bias;
  }

public:
  L4::Cap<void> alloc() throw()
  {
    if (_free_hint >= _capacity)
      return L4::Cap_base::Invalid;

    for (long i = _free_hint; i < _capacity; ++i)
      {
  if (_items[i].is_free())
    {
      _items[i].alloc();
      _free_hint = i + 1;

      return L4::Cap<void>((i + _bias) << L4_CAP_SHIFT);
    }
      }

    return L4::Cap<void>::Invalid;
  }

  template <typename T>
  L4::Cap<T> alloc() throw()
  {
    return L4::cap_cast<T>(alloc());
  }


  void take(L4::Cap<void> cap) throw()
  {
    long c = cap.cap() >> L4_CAP_SHIFT;
    if (c < _bias)
      return;

    c -= _bias;
    if (c >= _capacity)
      return;

    _items[c].inc();
  }


  bool free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
            unsigned unmap_flags = L4_FP_ALL_SPACES) throw()
  {
    long c = cap.cap() >> L4_CAP_SHIFT;
    if (c < _bias)
      return false;

    c -= _bias;

    if (c >= _capacity)
      return false;
```

```
00209
00210     l4_assert(!_items[c].is_free());
00211
00212     if (task != L4_INVALID_CAP)
00213       l4_task_unmap(task, cap.fpage(), unmap_flags);
00214
00215     if (c < _free_hint)
00216       _free_hint = c;
00217
00218     _items[c].free();
00219
00220     return true;
00221   }
00222
00241   bool release(L4::Cap<void> cap, l4_cap_idx_t task =
      L4_INVALID_CAP,
00242                unsigned unmap_flags = L4_FP_ALL_SPACES) throw()
00243   {
00244     long c = cap.cap() >> L4_CAP_SHIFT;
00245     if (c < _bias)
00246       return false;
00247
00248     c -= _bias;
00249
00250     if (c >= _capacity)
00251       return false;
00252
00253     l4_assert(!_items[c].is_free());
00254
00255     if (_items[c].dec() == Counter::nil())
00256       {
00257   if (task != L4_INVALID_CAP)
00258     l4_task_unmap(task, cap.fpage(), unmap_flags);
00259
00260   if (c < _free_hint)
00261     _free_hint = c;
00262
00263   return true;
00264       }
00265     return false;
00266   }
00267
00268
00272   long last() throw()
00273   {
00274     return _capacity + _bias - 1;
00275   }
00276 };
00277
00278 }}
00279
```

15.249 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++.

15.249.1 Detailed Description

Item allocator.

Definition in file item_alloc.

15.250 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.
```

```
 */

#pragma once

#include <l4/cxx/bitmap>

namespace L4Re { namespace Util {

using cxx::Bitmap_base;
using cxx::Bitmap;

class Item_alloc_base
{
private:
  long _capacity;
  long _free_hint;
  Bitmap_base _bits;

public:
  bool is_allocated(long item) const throw()
  { return _bits[item]; }

  long hint() const { return _free_hint; }

  bool alloc(long item) throw()
  {
    if (!_bits[item])
      {
  _bits.set_bit(item);
  return true;
      }
    return false;
  }

  void free(long item) throw()
  {
    if (item < _free_hint)
      _free_hint = item;

    _bits.clear_bit(item);
  }

  Item_alloc_base(long size, void *mem) throw()
    : _capacity(size), _free_hint(0), _bits(mem)
  {}

  long alloc() throw()
  {
    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 throw()
  {
    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() throw() : Item_alloc_base(Bits, _bits) {}
};

}}
```

15.251 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()) throw ()

  Allocate state area.

15.251.1 Detailed Description

Kumem allocator helper.

Definition in file kumem_alloc.

15.252 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.
```

```
 *
 * 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>

namespace L4Re { namespace Util {


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()) throw();

}}
```

15.253 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++.

15.253.1 Detailed Description

Region handling.

Definition in file region_mapping.

15.254 region_mapping

```
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
00027 #pragma once
00028
00029 #include <l4/cxx/avl_map>
00030 #include <l4/sys/types.h>
00031 #include <l4/re/rm>
00032
00033
00034 namespace L4Re { namespace Util {
```

```
class Region
{
private:
  l4_addr_t _start, _end;

public:
  Region() throw() : _start(~0UL), _end(~0UL) {}
  Region(l4_addr_t addr) throw() : _start(addr), _end(addr) {}
  Region(l4_addr_t start, l4_addr_t end) throw()
    : _start(start), _end(end) {}
  l4_addr_t start() const throw() { return _start; }
  l4_addr_t end() const throw() { return _end; }
  unsigned long size() const throw() { return end() - start() + 1; }
  bool invalid() const throw() { return _start == ~0UL && _end == ~0UL; }
  bool operator < (Region const &o) const throw()
  { return end() < o.start(); }
  bool contains(Region const &o) const throw()
  { return o.start() >= start() && o.end() <= end(); }
  bool operator == (Region const &o) const throw()
  { return o.start() == start() && o.end() == end(); }
  ~Region() throw() {}
};

template< typename DS, typename OPS >
class Region_handler
{
private:
  l4_addr_t _offs;
  DS _mem;
  l4_cap_idx_t _client_cap = L4_INVALID_CAP;
  unsigned short _flags;

public:
  typedef DS Dataspace;
  typedef OPS Ops;
  typedef typename OPS::Map_result Map_result;

  Region_handler() throw() : _offs(0), _mem(), _flags() {}
  Region_handler(Dataspace const &mem, l4_cap_idx_t client_cap,
      l4_addr_t offset = 0, unsigned flags = 0) throw()
    : _offs(offset), _mem(mem), _client_cap(client_cap), _flags(flags)
  {}
  Dataspace const &memory() const throw() { return _mem; }
  l4_cap_idx_t client_cap_idx() const throw() { return _client_cap; }
  l4_addr_t offset() const throw() { return _offs; }
  l4_addr_t is_ro() const throw() { return _flags & L4Re::Rm::Read_only; }
  unsigned long caching() const throw() { return _flags & L4Re::Rm::Caching; }
  unsigned flags() const throw() { return _flags; }

  Region_handler operator + (long offset) const throw()
  { Region_handler n = *this; n._offs += offset; return n; }

  void unmap(l4_addr_t va, l4_addr_t ds_offs, unsigned long size) const throw()
  { Ops::unmap(this, va, ds_offs, size); }

  void free(l4_addr_t start, unsigned long size) const throw()
  { Ops::free(this, start, size); }

  void take() const    { Ops::take(this); }
  void release() const { Ops::release(this); }

  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) throw()
  {
    _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() throw() { return _rm.begin(); }
  Const_iterator begin() const throw() { return _rm.begin(); }
  Iterator end() throw() { return _rm.end(); }
  Const_iterator end() const throw() { return _rm.end(); }

  Iterator area_begin() throw() { return _am.begin(); }
  Const_iterator area_begin() const throw() { return _am.begin(); }
  Iterator area_end() throw() { return _am.end(); }
  Const_iterator area_end() const throw() { return _am.end(); }
  Node area_find(Key_type const &c) const throw() { return _am.find_node(c); }

  enum Attach_flags
  {
    None      = 0,
    Search    = L4Re::Rm::Search_addr,
    In_area   = L4Re::Rm::In_area,
  };

  l4_addr_t min_addr() const throw() { return _start; }
  l4_addr_t max_addr() const throw() { return _end; }


  Region_map(l4_addr_t start, l4_addr_t end) throw() : _start(start), _end(end) {}

  Node find(Key_type const &key) const throw()
  {
    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 throw()
  {
    Node n = _rm.lower_bound_node(key);
    return n;
  }

  Node lower_bound_area(Key_type const &key) const throw()
  {
    Node n = _am.lower_bound_node(key);
    return n;
  }

  l4_addr_t attach_area(l4_addr_t addr, unsigned long size,
                        unsigned flags = None,
                        unsigned char align = L4_PAGESHIFT) throw()
  {
    if (size < 2)
      return L4_INVALID_ADDR;


    Region c;

    if (!(flags & Search))
      {
  c = Region(addr, addr + size - 1);
  Node r = _am.find_node(c);
  if (r)
    return L4_INVALID_ADDR;
      }

    while (flags & Search)
      {
  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, flags)).second == 0)
      return addr;

    return L4_INVALID_ADDR;
  }

  bool detach_area(l4_addr_t addr) throw()
  {
    if (_am.remove(addr))
      return false;

    return true;
  }

  void *attach(void *addr, unsigned long size, Hdlr const &hdlr,
               unsigned flags = None, unsigned char align = L4_PAGESHIFT) throw()
  {
    if (size < 2)
      return L4_INVALID_PTR;

    l4_addr_t end = max_addr();
    l4_addr_t beg = (l4_addr_t)addr;

    if (flags & In_area)
      {
  Node r = _am.find_node(Region(beg, beg + size - 1));
  if (!r || (r->second.flags() & L4Re::Rm::Reserved))
    return L4_INVALID_PTR;

  end = r->first.end();
      }

    if (flags & Search)
      {
  beg = find_free(beg, end, size, align, flags);
  if (beg == L4_INVALID_ADDR)
    return L4_INVALID_PTR;
      }

    if (!(flags & (Search | 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) throw()
  {
    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::Detach_free))
    h.free(0, g.size());

  if (hdlr) *hdlr = h;
  if (reg) *reg = g;
```

```
00294
00295   if (find(dr))
00296     return Rm::Detached_ds | Rm::Detach_again;
00297   else
00298     return Rm::Detached_ds;
00299       }
00300     else if (dr.start() <= g.start())
00301       {
00302   // move the start of a region
00303
00304   if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::Detach_free))
00305     h.free(0, dr.end() + 1 - g.start());
00306
00307   unsigned long sz = dr.end() + 1 - g.start();
00308   Item *cn = const_cast<Item*>((Item const *)r);
00309   cn->first = Region(dr.end() + 1, g.end());
00310   cn->second = cn->second + sz;
00311   if (hdlr) *hdlr = Hdlr();
00312   if (reg) *reg = Region(g.start(), dr.end());
00313   if (find(dr))
00314     return Rm::Kept_ds | Rm::Detach_again;
00315   else
00316     return Rm::Kept_ds;
00317       }
00318     else if (dr.end() >= g.end())
00319       {
00320   // move the end of a region
00321
00322   if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::Detach_free))
00323     h.free(dr.start() - g.start(), g.end() + 1 - dr.start());
00324
00325   Item *cn = const_cast<Item*>((Item const*)r);
00326   cn->first = Region(g.start(), dr.start() -1);
00327   if (hdlr) *hdlr = Hdlr();
00328   if (reg) *reg = Region(dr.start(), g.end());
00329
00330   if (find(dr))
00331     return Rm::Kept_ds | Rm::Detach_again;
00332   else
00333     return Rm::Kept_ds;
00334       }
00335     else if (g.contains(dr))
00336       {
00337   // split a single region that contains the new region
00338
00339   if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::Detach_free))
00340     h.free(dr.start() - g.start(), dr.size());
00341
00342   // first move the end off the existing region before the new one
00343   const_cast<Item*>((Item const *)r)->first = Region(g.start(), dr.start()-1);
00344
00345   int err;
00346
00347   // insert a second region for the remaining tail of
00348   // the old existing region
00349   err = _rm.insert(Region(dr.end() + 1, g.end()), h + (dr.end() + 1 - g.start())).second;
00350
00351   if (err)
00352     return err;
00353
00354   if (hdlr) *hdlr = h;
00355   if (reg) *reg = dr;
00356   return Rm::Split_ds;
00357       }
00358     return -L4_ENOENT;
00359   }
00360
00361   l4_addr_t find_free(l4_addr_t start, l4_addr_t end,
      l4_addr_t size,
00362                       unsigned char align, unsigned flags) const throw();
00363
00364 };
00365
00366
00367 template< typename Hdlr, template<typename T> class Alloc >
00368 l4_addr_t
00369 Region_map<Hdlr, Alloc>::find_free(l4_addr_t start, l4_addr_t end,
00370     unsigned long size, unsigned char align, unsigned flags) const throw()
00371 {
00372   l4_addr_t addr = start;
00373
00374   if (addr == ~0UL || addr < min_addr() || addr >= end)
00375     addr = min_addr();
00376
00377   addr = l4_round_size(addr, align);
00378   Node r;
00379
```

```
  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 & 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;
}

}}
```

15.255 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.

15.255.1 Detailed Description

Goos protocol definition.

Definition in file goos-sys.h.

15.256 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
    };
  };
}}
```

15.257 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.

15.257.1 Detailed Description

Shared memory library header file.

Definition in file shmc.h.

15.258 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
00114 l4shmc_attach_to(const char *shmc_name, l4_umword_t timeout_ms,
00115                  l4shmc_area_t *shmarea);
00116
00129 L4_CV long
00130 l4shmc_add_chunk(l4shmc_area_t *shmarea,
00131                  const char *chunk_name,
00132                  l4_umword_t chunk_capacity,
00133                  l4shmc_chunk_t *chunk);
00134
00146 L4_CV long
00147 l4shmc_add_signal(l4shmc_area_t *shmarea,
00148                   const char *signal_name,
00149                   l4shmc_signal_t *signal);
00150
00160 L4_CV L4_INLINE long
00161 l4shmc_trigger(l4shmc_signal_t *signal);
00162
00172 L4_CV L4_INLINE long
00173 l4shmc_chunk_try_to_take(l4shmc_chunk_t *chunk);
00174
00185 L4_CV L4_INLINE long
00186 l4shmc_chunk_ready(l4shmc_chunk_t *chunk, l4_umword_t size);
00187
00198 L4_CV L4_INLINE long
00199 l4shmc_chunk_ready_sig(l4shmc_chunk_t *chunk, l4_umword_t size);
00200
00212 L4_CV L4_INLINE long
00213 l4shmc_get_chunk(l4shmc_area_t *shmarea,
00214                  const char *chunk_name,
00215                  l4shmc_chunk_t *chunk);
00216
00230 L4_CV long
00231 l4shmc_get_chunk_to(l4shmc_area_t *shmarea,
00232                     const char *chunk_name,
00233                     l4_umword_t timeout_ms,
00234                     l4shmc_chunk_t *chunk);
00235
00249 L4_CV long
00250 l4shmc_iterate_chunk(l4shmc_area_t *shmarea, const char **chunk_name,
00251                      long offs);
00252
00265 L4_CV L4_INLINE long
00266 l4shmc_attach_signal(l4shmc_area_t *shmarea,
00267                      const char *signal_name,
00268                      l4_cap_idx_t thread,
00269                      l4shmc_signal_t *signal);
00270
00285 L4_CV long
00286 l4shmc_attach_signal_to(l4shmc_area_t *shmarea,
00287                         const char *signal_name,
00288                         l4_cap_idx_t thread,
00289                         l4_umword_t timeout_ms,
00290                         l4shmc_signal_t *signal);
00291
00305 L4_CV long
00306 l4shmc_get_signal_to(l4shmc_area_t *shmarea,
00307                      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>

```

```
00647 __END_DECLS
```

15.259 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_TBUF 0xB0

  TBUF.
- #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_TBUF (SIGMA0_REQ(TBUF))

  TBUF.
- #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_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 pager 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.
- int l4sigma0_map_tbuf (l4_cap_idx_t sigma0, l4_addr_t virt)

  Request Fiasco trace buffer.
- 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 a user readable error messages for the return codes.

15.259.1 Detailed Description

Sigma0 interface.

Definition in file sigma0.h.

15.260 sigma0.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  *               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 __L4_SIGMA0_SIGMA0_H
00016 #define __L4_SIGMA0_SIGMA0_H
00017
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/kip.h>
00029
00036 #undef SIGMA0_REQ_MAGIC
00037 #undef SIGMA0_REQ_MASK
00038
00039 # define SIGMA0_REQ_MAGIC   ~0xFFUL
00040 # define SIGMA0_REQ_MASK    ~0xFFUL
00042 /* Starting with 0x60 allows to detect components which still use the old
00043  * constants (0x00 ... 0x50) */
00044 #define SIGMA0_REQ_ID_MASK      0xF0
00045 #define SIGMA0_REQ_ID_FPAGE_RAM     0x60
00046 #define SIGMA0_REQ_ID_FPAGE_IOMEM   0x70
00047 #define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED  0x80
00048 #define SIGMA0_REQ_ID_FPAGE_ANY     0x90
00049 #define SIGMA0_REQ_ID_KIP     0xA0
00050 #define SIGMA0_REQ_ID_TBUF      0xB0
00051 #define SIGMA0_REQ_ID_DEBUG_DUMP    0xC0
00052 #define SIGMA0_REQ_ID_NEW_CLIENT    0xD0
00054 #define SIGMA0_IS_MAGIC_REQ(d1) \
00055   ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)
00057 #define SIGMA0_REQ(x) \
00058   (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## x)
00060 /* Use these constants in your code! */
00061 #define SIGMA0_REQ_FPAGE_RAM            (SIGMA0_REQ(FPAGE_RAM))
00062 #define SIGMA0_REQ_FPAGE_IOMEM          (SIGMA0_REQ(FPAGE_IOMEM))
00063 #define SIGMA0_REQ_FPAGE_IOMEM_CACHED   (SIGMA0_REQ(FPAGE_IOMEM_CACHED))
00064 #define SIGMA0_REQ_FPAGE_ANY            (SIGMA0_REQ(FPAGE_ANY))
00065 #define SIGMA0_REQ_KIP                  (SIGMA0_REQ(KIP))
00066 #define SIGMA0_REQ_TBUF                 (SIGMA0_REQ(TBUF))
00067 #define SIGMA0_REQ_DEBUG_DUMP           (SIGMA0_REQ(DEBUG_DUMP))
00068 #define SIGMA0_REQ_NEW_CLIENT           (SIGMA0_REQ(NEW_CLIENT))
00070
00071
00075
00079 enum l4sigma0_return_flags_t {
00080   L4SIGMA0_OK,
00081   L4SIGMA0_NOTALIGNED,
00082   L4SIGMA0_IPCERROR,
00083   L4SIGMA0_NOFPAGE,
00084   L4SIGMA0_4,
00085   L4SIGMA0_5,
00086   L4SIGMA0_SMALLERFPAGE,
00087 };
00088
```

```
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 int l4sigma0_map_tbuf(l4_cap_idx_t sigma0,
l4_addr_t virt);

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 */
```

15.261 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:

15.261.1 Detailed Description

Low-level kernel debugger functions.

Definition in file __kernel_object_impl.h.

15.262 __kernel_object_impl.h

```
00001
00005 #pragma once
00006
00007 #include <l4/sys/ipc.h>
00008
00009 L4_INLINE l4_msgtag_t
00010 l4_invoke_debugger(l4_cap_idx_t obj, l4_msgtag_t tag,
      l4_utcb_t *utcb) L4_NOTHROW
00011 {
00012   l4_msgtag_t t2;
00013   l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00014
00015   if (l4_is_invalid_cap(obj))
```

```
    return l4_msgtag(-L4_EINVAL, 0, 0, 0);

  mr->mr[0] += 0x100;
  mr->mr[l4_msgtag_words(tag)]      = L4_ITEM_MAP;
  mr->mr[l4_msgtag_words(tag) + 1] = l4_obj_fpage(obj, 0,
L4_FPAGE_RWX).raw;
  t2 = l4_msgtag(L4_PROTO_DEBUGGER, l4_msgtag_words(tag),
                 1, l4_msgtag_flags(tag));

  return l4_ipc_call(L4_BASE_DEBUGGER_CAP, utcb, t2,
L4_IPC_NEVER);
}
```

15.263 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_tracebuffer_status_t ∗ fiasco_tbuf_get_status (void)

  Return trace-buffer status.
- l4_addr_t fiasco_tbuf_get_status_phys (void)

  Return the physical address of the trace-buffer status struct.
- 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.

15.263.1 Detailed Description

L4 kernel event tracing.

Definition in file __ktrace-impl.h.

15.264 __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
```

```
 * 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/kdebug.h>

/*****************************************************************************
 *** Implementation
 *****************************************************************************/

L4_INLINE l4_tracebuffer_status_t *
fiasco_tbuf_get_status(void)
{
  return 0;
  /* Not implemented */
}

L4_INLINE l4_addr_t
fiasco_tbuf_get_status_phys(void)
{
  return ~0UL;
}

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));
}
```

15.265 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::Kobject_demand< T >

  Get the combined server-side resource requirements for all type T...
- struct L4::Proto_t< P >

  Data type for defining protocol numbers.
- struct L4::Kobject_x< Derived, ARGS >

  Generic Kobject inheritance template.

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 ()

  Get the L4::Type_info for the L4Re interface given in T.

15.265.1 Detailed Description

Type information handling.

Definition in file __typeinfo.h.

15.266 __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)
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)
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
```

```
/*********************/
template<long P, typename T>
struct Iface
{
  typedef Iface type;
  typedef T iface_type;
  enum { Proto = P };
};


/*********************/
struct Iface_list_end
{
  typedef Iface_list_end type;
  static bool contains(long) { 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)
  { 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 &)
```

```
    { 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) : rights(rights) {}
    unsigned operator & (unsigned rhs) const { 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)
    { 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)
    : caps(caps), flags(flags), mem(mem), ports(ports) {}

    bool no_demand() const
    { return caps == 0 && mem == 0 && ports == 0 && flags == 0; }

    Demand operator | (Demand const &rhs) const
    {
      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() : 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 { return _type; }
  Type_info const *base(unsigned idx) const { return _bases[idx]; }
  unsigned num_bases() const { return _num_bases; }
  long proto() const { return _proto; }
  char const *name() const { return L4_kobject_type_name(type()); }
  bool has_proto(long proto) const
  {
    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() { return &T::__Kobject_typeid::_m; }

  static Type_info::Demand demand()
  { 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()
{ 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__()
  {
    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 { 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__()
  {
    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 throw()
  { return Base1::cap(); }

  L4::Cap<Class> c() const { 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
  { 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())
  { 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__()
  {
    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 throw()
  { return Base1::cap(); }

  L4::Cap<Class> c() const { 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
  { 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())
  { 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__()
  {
    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 throw()
  { return Typeid_xx::First<BASES...>::type::cap(); }

  L4::Cap<Class> c() const { 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
  { 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())
  { 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
```

15.267 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.

15.267.1 Detailed Description

Cache-consistency functions.

Date

2007-11

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file cache.h.

15.268 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
```

```
L4_INLINE int
l4_cache_coherent(unsigned long start,
                  unsigned long end) L4_NOTHROW;

L4_INLINE int
l4_cache_dma_coherent(unsigned long start,
                      unsigned long end) L4_NOTHROW;

L4_INLINE int
l4_cache_dma_coherent_full(void) L4_NOTHROW;

EXTERN_C_END

#endif /* ! __L4SYS__INCLUDE__CACHE_H__ */
```

15.269 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.

15.269.1 Detailed Description

Cache functions.

Date

2007-11

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file cache.h.

15.270 cache.h

```
00001
00009 /*
00010  * (c) 2007-2009 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 #ifndef __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00027 #define __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/syscall_defs.h>
00031
00032 #include_next <l4/sys/cache.h>
00033
00037 L4_INLINE void
00038 l4_cache_op_arm_call(unsigned long op,
00039                      unsigned long start,
00040                      unsigned long end);
00041
00042 L4_INLINE void
00043 l4_cache_op_arm_call(unsigned long op,
00044                      unsigned long start,
00045                      unsigned long end)
00046 {
00047   register unsigned long _op    __asm__ ("r0") = op;
00048   register unsigned long _start __asm__ ("r1") = start;
00049   register unsigned long _end   __asm__ ("r2") = end;
00050
00051   __asm__ __volatile__
00052     ("@ l4_cache_op_arm_call(start) \n\t"
00053      "mov     lr, pc              \n\t"
00054      "mov     pc, %[sc]             \n\t"
00055      "@ l4_cache_op_arm_call(end)   \n\t"
00056        :
00057   "=r" (_op),
00058   "=r" (_start),
00059   "=r" (_end)
00060        :
00061        [sc] "i" (L4_SYSCALL_MEM_OP),
00062   "0" (_op),
00063   "1" (_start),
00064   "2" (_end)
00065        :
00066   "cc", "memory", "lr"
00067        );
```

```
}

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_MEM_CACHE_OP_L2_CLEAN          = 6,
  L4_MEM_CACHE_OP_L2_FLUSH          = 7,
  L4_MEM_CACHE_OP_L2_INV            = 8,
};

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;
}

L4_INLINE int
l4_cache_l2_clean(unsigned long start,
                  unsigned long end) L4_NOTHROW
{
  l4_cache_op_arm_call(L4_MEM_CACHE_OP_L2_CLEAN, start, end);
  return 0;
}

L4_INLINE int
l4_cache_l2_flush(unsigned long start,
                  unsigned long end) L4_NOTHROW
{
  l4_cache_op_arm_call(L4_MEM_CACHE_OP_L2_FLUSH, start, end);
  return 0;
}

L4_INLINE int
l4_cache_l2_inv(unsigned long start,
                unsigned long end) L4_NOTHROW
{
  l4_cache_op_arm_call(L4_MEM_CACHE_OP_L2_INV, start, end);
  return 0;
}

#endif /* ! __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__ */
```

15.271 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.

15.271.1 Detailed Description

Cache functions.

Definition in file cache.h.

15.272 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_AMD64__CACHE_H__
00023 #define __L4SYS__INCLUDE__ARCH_AMD64__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
```

```
{
  (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__ */
```

15.273 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.

15.273.1 Detailed Description

Cache functions.

Definition in file cache.h.

15.274 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
00054 {
00055   (void)start; (void)end;
00056   return 0;
00057 }
00058
00059 L4_INLINE int
00060 l4_cache_dma_coherent(unsigned long start,
00061                       unsigned long end) L4_NOTHROW
00062 {
00063   (void)start; (void)end;
00064   return 0;
00065 }
00066
00067 L4_INLINE int
00068 l4_cache_dma_coherent_full(void) L4_NOTHROW
00069 {
00070   return 0;
00071 }
00072
00073 #endif /* ! __L4SYS__INCLUDE__ARCH_X86__CACHE_H__ */
```

15.275 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) throw ()

  `dynamic_cast` *for capabilities.*

15.275.1 Detailed Description

L4::Cap related definitions.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file capability.

15.275.2 Macro Definition Documentation

15.275.2.1 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.

15.276 capability

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00009 /*
00010  * (c) 2008-2009,2015 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
```

```
 * 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/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kobject>
#include <l4/sys/task.h>

namespace L4
{

/* Forward declarations for our kernel object classes. */
class Task;
class Thread;
class Factory;
class Irq;
class Log;
class Vm;
class Kobject;

#if __cplusplus >= 201103L
#  define L4_DISABLE_COPY(_class)                \
  public:                                        \
    _class(_class const &) = delete;             \
    _class operator = (_class const &) = delete; \
  private:
#else
#  define L4_DISABLE_COPY(_class)        \
  private:                               \
    _class(_class const &);              \
    _class operator = (_class const &);
#endif


#define L4_KOBJECT_DISABLE_COPY(_class) \
  protected:                            \
    _class();                           \
  L4_DISABLE_COPY(_class)


#define L4_KOBJECT(_class) L4_KOBJECT_DISABLE_COPY(_class)

inline l4_msgtag_t
Cap_base::validate(Cap<Task> task, l4_utcb_t *u) const throw()
{ return l4_task_cap_valid_u(task.cap(), _c, u);  }

inline l4_msgtag_t
Cap_base::validate(l4_utcb_t *u) const throw()
{ return l4_task_cap_valid_u(L4_BASE_TASK_CAP, _c, u);  }

}; // namespace L4

#include <l4/sys/meta>

namespace L4 {

template< typename T, typename F >
inline
Cap<T> cap_dynamic_cast(Cap<F> const &c) throw()
{
  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;
```

```
00136 }
00137
00138 }
```

15.277 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

- void l4_barrier (void)

  Memory barrier.
- void l4_mb (void)

  Memory barrier.
- void l4_wmb (void)

  Write memory barrier.

15.277.1 Detailed Description

L4 compiler related defines.

Definition in file compiler.h.

15.278 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 */
#  define L4_NOTHROW throw()
#  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)));


#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 __GXX_EXPERIMENTAL_CXX0X__
#ifndef static_assert
#define static_assert(x, y) \
  do { (void)sizeof(char[-(!(x))]); } while (0)
#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__ */
```

15.279 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_MASK, L4_INVALID_CAP }

  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 }

  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 = L4_BASE_CA↩PS_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_umword_t value, unsigned char bits) L4_NOTHROW

  Round value up to the next alignment with bits size.

15.279.1 Detailed Description

Common constants.

Definition in file consts.h.

15.280 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
```

```
enum l4_cap_consts_t
{
  L4_CAP_SHIFT   = 12UL,
  L4_CAP_SIZE    = 1UL << L4_CAP_SHIFT,
  L4_CAP_OFFSET  = 1UL << L4_CAP_SHIFT,
  L4_CAP_MASK    = ~0UL << (L4_CAP_SHIFT - 1),
  L4_INVALID_CAP = ~0UL << (L4_CAP_SHIFT - 1),

  L4_INVALID_CAP_BIT = 1UL << (L4_CAP_SHIFT - 1),
};

enum l4_sched_consts_t
{
  L4_SCHED_MIN_PRIO = 0,
  L4_SCHED_MAX_PRIO = 255,
};

enum l4_unmap_flags_t
{
  L4_FP_ALL_SPACES   = 0x80000000UL,

  L4_FP_DELETE_OBJ   = 0xc0000000UL,

  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
00348 { return address & (~0UL << bits); }
00349
00360 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address)
      L4_NOTHROW;
00361 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW
00362 { return (address + L4_PAGESIZE - 1) & L4_PAGEMASK; }
00363
00371 L4_INLINE l4_addr_t l4_round_size(l4_umword_t value, unsigned char bits)
      L4_NOTHROW;
00372 L4_INLINE l4_addr_t l4_round_size(l4_umword_t value, unsigned char bits)
      L4_NOTHROW
00373 { return (value + (1UL << bits) - 1) & (~0UL << bits); }
00374
00379 enum l4_addr_consts_t {
00381   L4_INVALID_ADDR = ~0UL
00382 };
00383
00388 #define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)
00389
00390 #ifndef NULL
00391 #ifndef __cplusplus
00392 # define NULL ((void *)0)
00396 #else
00397 # define NULL 0
00398 #endif
00399 #endif
00400
00401 #endif /* ! __L4_SYS__INCLUDE__CONSTS_H__ */
```

15.281 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.

15.281.1 Detailed Description

Common L4 constants, arm version.

Definition in file consts.h.

15.282 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  *     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_CONSTS_H
00026 #define _L4_SYS_CONSTS_H
00027
00028 /* L4 includes */
00029 #include <l4/sys/l4int.h>
00037 #define L4_PAGESHIFT    12
00038
00042 #define L4_SUPERPAGESHIFT 21
00043
00046 #include_next <l4/sys/consts.h>
00047
00048 #endif /* !_L4_SYS_CONSTS_H */
```

15.283 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.

15.283.1 Detailed Description

Common L4 constants, amd64 version.

Definition in file consts.h.

15.284 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__ */
```

15.285 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.

15.285.1 Detailed Description

Common L4 constants, x86 version.

Definition in file consts.h.

15.286 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>,
 *               Lars Reuther <reuther@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_X86__CONSTS_H__
#define __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__

#define L4_PAGESHIFT    12

#define L4_SUPERPAGESHIFT 22

#include_next <l4/sys/consts.h>

#endif /* ! __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__ */
```

15.287 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:

15.287.1 Detailed Description

Constants.

Definition in file consts.h.

15.288 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
```

15.289 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.

15.289.1 Macro Definition Documentation

15.289.1.1 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.

15.290 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
```

```
 * 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/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) throw()
{
  return Rpc_inline_call<OP, C, R (ARGS...), FLAGS>::call(cap, a..., utcb);
}

} // namespace Msg
} // namespace Ipc
} // namespace L4

```

15.291 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).

15.291.1 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.

15.291.2 Macro Definition Documentation

15.291.2.1 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 457 of file ipc_iface.

Referenced by L4Re::Video::Goos::delete_view().

15.291.2.2 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 Rpc_inline_call 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 429 of file ipc_iface.

Referenced by L4::Factory::create(), L4Re::Video::Goos::create_view(), and L4Re::Inhibitor::next_lock_info().

15.291.2.3 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 Rpc_inline_call 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 442 of file ipc_iface.

15.291.2.4 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 472 of file ipc_iface.

Referenced by L4::Scheduler::info(), and L4::Icu::info().

15.291.2.5 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 515 of file ipc_iface.

Referenced by L4Re::Rm::find(), and L4Re::Rm::reserve_area().

15.291.2.6 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 Rpc_call 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 485 of file ipc_iface.

Referenced by L4Re::Rm::find(), and L4Re::Rm::reserve_area().

15.291.2.7 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 Rpc_call 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 500 of file ipc_iface.

Referenced by L4::Icu::bind(), L4::Icu::info(), L4::Icu::mask(), L4Re::Namespace::register_obj(), L4::Icu::set_mode(), and L4::Icu::unbind().

15.291.2.8 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 530 of file ipc_iface.

15.292 ipc_iface

```
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
00021 #include <l4/sys/cxx/ipc_basics>
00022 #include <l4/sys/cxx/ipc_types>
00023 #include <l4/sys/__typeinfo.h>
00024
00196 // TODO: add some more documentation
00197 namespace L4 { namespace Ipc {
00198
00215 struct L4_EXPORT Call
00216 {
00217   enum { Is_call = true };
00218   enum { Rights = 0 };
00219   static l4_timeout_t timeout() { return L4_IPC_NEVER; }
00220 };
00221
00225 struct L4_EXPORT Call_zero_send_timeout : Call
00226 {
00227   static l4_timeout_t timeout() { return L4_IPC_SEND_TIMEOUT_0; }
00228 };
00229
00245 template<unsigned RIGHTS>
00246 struct L4_EXPORT Call_t : Call
00247 {
00248   enum { Rights = RIGHTS };
00249 };
00250
00263 struct L4_EXPORT Send_only
00264 {
00265   enum { Is_call = false };
00266   enum { Rights = 0 };
00267   static l4_timeout_t timeout() { return L4_IPC_NEVER; }
00268 };
00269
00270 namespace Msg {
00271
00282 template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
00283 struct L4_EXPORT Rpc_inline_call;
00284
00289 template<typename OP, typename CLASS, typename FLAGS, typename R,
00290          typename ...ARGS>
00291 struct L4_EXPORT Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
00292 {
00293   template<typename T> struct Result { typedef T result_type; };
00294   enum
00295   {
00296     Return_tag = L4::Types::Same<R, l4_msgtag_t>::value
00297   };
00298
00300   typedef Rpc_inline_call type;
00302   typedef OP op_type;
00304   typedef CLASS class_type;
00306   typedef typename Result<R>::result_type result_type;
00308   typedef R ipc_type (ARGS...);
00310   typedef result_type func_type (typename _Elem<ARGS>::arg_type...);
00311
00313   typedef FLAGS flags_type;
00314
00315   template<typename RES>
00316   static typename L4::Types::Enable_if< Return_tag, RES >::type
00317   return_err(long err) { return l4_msgtag(err, 0, 0, 0); }
00318
00319   template<typename RES>
00320   static typename L4::Types::Enable_if< Return_tag, RES >::type
00321   return_ipc_err(l4_msgtag_t tag, l4_utcb_t const *) { return tag; }
00322
```

```
  template<typename RES>
  static typename L4::Types::Enable_if< Return_tag, RES >::type
  return_code(l4_msgtag_t tag) { return tag; }

  template<typename RES>
  static typename L4::Types::Enable_if< !Return_tag, RES >::type
  return_err(long err) { 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)
  { 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) { return tag.label(); }

  static R call(L4::Cap<class_type> cap,
                typename _Elem<ARGS>::arg_type ...a,
                l4_utcb_t *utcb = l4_utcb()) throw();
};

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
  {
    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 throw()
  { 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()) throw();
};

#define L4_INLINE_RPC_SRV_FORWARD(name)                                  \
  template<typename OBJ> struct fwd                                      \
  {                                                                      \
    OBJ *o;                                                              \
    fwd(OBJ *o) : o(o) {}                                                \
    template<typename ...ARGS> long call(ARGS ...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 Rpc_inline_call 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 Rpc_inline_call 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 Rpc_call 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 Rpc_call 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 int write(char *, int offset, int)
  { return offset; }

  template<typename DIR>
  static int read(char *, int offset, int, long)
  { 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)
  {
    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)
  {
    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)
  {
    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)
  {
    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)
  {
    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)
  {
    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) throw()
{
  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;
```

```
00769
00770   if (L4_UNLIKELY(err < 0 || item_limit > Mr_bytes))
00771     return return_err<R>(-L4_EMSGTOOSHORT);
00772
00773   // read the static out-items to the arguments
00774   err = Args::template read<Do_out_items>(&mrs->mr[t.words()], 0, item_limit,
00775                                           r, a...);
00776
00777   if (L4_UNLIKELY(err < 0))
00778     return return_err<R>(-L4_EMSGTOOSHORT);
00779
00780   return return_code<R>(t);
00781 }
00782
00783 } // namespace Msg
00784 } // namesapce Ipc
00785 } // namespace L4
00786
00787
```

15.293 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.

15.293.1 Detailed Description

IPC server loop.

Definition in file ipc_server.

15.294 ipc_server

```
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
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029 #include <l4/sys/typeinfo_svr>
00030 #include <l4/sys/err.h>
00031 #include <l4/cxx/ipc_stream>
00032 #include <l4/sys/cxx/ipc_epiface>
00033 #include <l4/sys/cxx/ipc_server_loop>
00034 #include <l4/cxx/type_traits>
00035 #include <l4/cxx/exceptions>
00036
00037 namespace L4 {
00038
00050 class Server_object : public Epiface
00051 {
00052 public:
00070   virtual int dispatch(unsigned long rights, Ipc::Iostream &ios) = 0;
00071
00072   l4_msgtag_t dispatch(l4_msgtag_t tag, unsigned rights,
      l4_utcb_t *utcb)
00073   {
00074     L4::Ipc::Iostream ios(utcb);
00075     ios.tag() = tag;
00076     int r = dispatch(rights, ios);
00077     return ios.prepare_ipc(r);
00078   }
00079
00080   Cap<Kobject> obj_cap() const
00081   { return cap_cast<Kobject>(Epiface::obj_cap()); }
00082 };
```

```
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> {};

}
```

15.295 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::Out< T >

  Mark an argument as a output value in an RPC signature.
- 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)

  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)

  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)

  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)

  Make an L4::IPC::Cap<T> for the given capability with full fpage and object-specific rights.

15.296 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() {}
```

```
  As_value(T v) : v(v) {}
  operator T () const { return v; }
  operator T & () { 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() : _valid(false) {}

  Opt(T value) : _value(value), _valid(true) {}

  Opt &operator = (T value)
  {
    this->_value = value;
    this->_valid = true;
    return *this;
  }

  void set_valid(bool valid = true) { _valid = valid; }

  T *operator -> () { return &this->_value; }
  T const *operator -> () const { return &this->_value; }
  T value() const { return this->_value; }
  T &value() { return this->_value; }
  bool is_valid() const { 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)
  {
    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)
  {
    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)
  {
    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)
  : _data(cap.cap() | L4_RCV_ITEM_SINGLE_CAP | flags) {}

  explicit Small_buf(l4_cap_idx_t cap, unsigned long flags = 0)
  : _data(cap | L4_RCV_ITEM_SINGLE_CAP | flags) {}

  l4_umword_t raw() const { return _data; }
private:
  l4_umword_t _data;
};

class Snd_item
{
public:
  Snd_item(l4_umword_t base, l4_umword_t data) : _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) : _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) : 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)
  : 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() : 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)
  : 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)
  : 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)
  {
    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)
  {
    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)
  {
    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)
  {
    return Gen_fpage<T>(Io, base << 12, order, rights, snd_base, map_type, None, cont);
  }

  unsigned order() const { return (T::_data >> 6) & 0x3f; }
  unsigned snd_order() const { return (T::_data >> 6) & 0x3f; }
  unsigned rcv_order() const { return (T::_base >> 6) & 0x3f; }
  l4_addr_t base() const { return T::_data & (~0UL << 12); }
  l4_addr_t snd_base() const { return T::_base & (~0UL << 10); }
  void snd_base(l4_addr_t b) { T::_base = (T::_base & ~(~0UL << 10)) | (b & (~0UL << 10)); }

  bool is_valid() const { return T::_base & L4_ITEM_MAP; }
```

```
00438   bool cap_received() const { return (T::_base & 0x3e) == 0x38; }
00450   bool id_received() const { return (T::_base & 0x3e) == 0x3c; }
00460   bool local_id_received() const { return (T::_base & 0x3e) == 0x3e; }
00461
00468   bool is_compound() const { return T::_base & 1; }
00470   l4_umword_t data() const { return T::_data; }
00472   l4_umword_t base_x() const { return T::_base; }
00473 };
00474
00475
00477 typedef Gen_fpage<Snd_item> Snd_fpage;
00479 typedef Gen_fpage<Buf_item> Rcv_fpage;
00480
00481 #ifdef L4_CXX_IPC_SUPPORT_STRINGS
00482 template <typename T, typename B>
00483 class Gen_string : public T
00484 {
00485 public:
00486   Gen_string() : T(0, 0) {}
00487   Gen_string(B buf, unsigned long size)
00488   : T(size << 10, l4_umword_t(buf))
00489   {}
00490
00491   unsigned long len() const { return T::_base >> 10; }
00492 };
00493
00494 typedef Gen_string<Snd_item, void const *> Snd_string;
00495 typedef Gen_string<Buf_item, void *> Rcv_string;
00496 #endif
00497
00498
00499 namespace Msg {
00500
00501 // Snd_fpage are out items
00502 template<> struct Class<L4::Ipc::Snd_fpage> : Cls_item {};
00503
00504 // Rcv_fpage are buffer items
00505 template<> struct Class<L4::Ipc::Rcv_fpage> : Cls_buffer {};
00506
00507 // Remove receive buffers from server-side arguments
00508 template<> struct Elem<L4::Ipc::Rcv_fpage>
00509 {
00510   typedef L4::Ipc::Rcv_fpage arg_type;
00511   typedef void svr_type;
00512   typedef void svr_arg_type;
00513   enum { Is_optional = false };
00514 };
00515
00516 // Rcv_fpage are buffer items
00517 template<> struct Class<L4::Ipc::Small_buf> : Cls_buffer {};
00518
00519 // Remove receive buffers from server-side arguments
00520 template<> struct Elem<L4::Ipc::Small_buf>
00521 {
00522   typedef L4::Ipc::Small_buf arg_type;
00523   typedef void svr_type;
00524   typedef void svr_arg_type;
00525   enum { Is_optional = false };
00526 };
00527 } // namespace Msg
00528
00529 // L4::Cap<> handling
00530
00541 template<typename T> class Cap
00542 {
00543   template<typename O> friend class Cap;
00544   l4_umword_t _cap_n_rights;
00545
00546 public:
00547   enum
00548   {
00554     Rights_mask = 0xff,
00555
00560     Cap_mask    = L4_CAP_MASK
00561   };
00562
00564   template<typename O>
00565   Cap(Cap<O> const &o) : _cap_n_rights(o._cap_n_rights)
00566   { T *x = (O*)1; (void)x; }
00567
00569   Cap(L4::Cap<T> cap)
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)
00576   : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
```

```
  { T *x = (O*)1; (void)x; }

  Cap() : _cap_n_rights(L4_INVALID_CAP) {}

  Cap(L4::Cap<T> cap, unsigned char rights)
  : _cap_n_rights((cap.cap() & Cap_mask) | (rights & Rights_mask)) {}

  static Cap from_ci(l4_cap_idx_t c)
  { return Cap(L4::Cap<T>(c & Cap_mask), c & Rights_mask); }

  L4::Cap<T> cap() const
  { return L4::Cap<T>(_cap_n_rights & Cap_mask); }

  unsigned rights() const
  { return _cap_n_rights & Rights_mask; }

  L4::Ipc::Snd_fpage fpage() const
  { return L4::Ipc::Snd_fpage(cap(), rights()); }

  bool is_valid() const throw()
  { return !(_cap_n_rights & L4_INVALID_CAP_BIT); }
};

template<typename T>
Cap<T> make_cap(L4::Cap<T> cap, unsigned rights)
{ return Cap<T>(cap, rights); }

template<typename T>
Cap<T> make_cap_rw(L4::Cap<T> cap)
{ return Cap<T>(cap, L4_CAP_FPAGE_RW); }

template<typename T>
Cap<T> make_cap_rws(L4::Cap<T> cap)
{ return Cap<T>(cap, L4_CAP_FPAGE_RWS); }

template<typename T>
Cap<T> make_cap_full(L4::Cap<T> cap)
{ return Cap<T>(cap, L4_CAP_FPAGE_RWSD |
L4_FPAGE_C_OBJ_RIGHTS); }

// caps are special the have an invalid representation
template<typename T> struct L4_EXPORT Opt< Cap<T> >
{
  Cap<T> _value;
  Opt() {}
  Opt(Cap<T> value) : _value(value) {}
  Opt(L4::Cap<T> value) : _value(value) {}
  Opt &operator = (Cap<T> value)
  { this->_value = value; }
  Opt &operator = (L4::Cap<T> value)
  { this->_value = value; }

  Cap<T> value() const { return this->_value; }
  bool is_valid() const { return this->_value.is_valid(); }
};


namespace Msg {
// prohibit L4::Cap as argument
template<typename A>
struct Is_valid_rpc_type< L4::Cap<A> > : L4::Types::False {};

template<typename A> struct Class< Cap<A> > : Cls_item {};
template<typename A> struct Elem< Cap<A> >
{
  enum { Is_optional = false };
  typedef Cap<A> arg_type;
  typedef L4::Ipc::Snd_fpage svr_type;
  typedef L4::Ipc::Snd_fpage svr_arg_type;
};


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)
  {
    // 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)
  {
    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)
  {
    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
```

15.297 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.

15.298 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
```

15.299 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::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++.

15.300 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

// very simple type traits for basic L4 functions, for a more complete set
// use <l4/cxx/type_traits> or the standard <type_traits>.

namespace L4 {

namespace Types {

  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; }
  };
```

```
00159
00165   template< bool V > struct Bool
00166   {
00167     typedef Bool<V> type;
00168     enum { value = V };
00169   };
00170
00173   struct False : Bool<false> {};
00174
00177   struct True : Bool<true> {};
00178
00179   /********************/
00188   template<typename A, typename B>
00189   struct Same : False {};
00190
00191   template<typename A>
00192   struct Same<A, A> : True {};
00193
00194   template<bool EXP, typename T = void> struct Enable_if {};
00195   template<typename T> struct Enable_if<true, T> { typedef T type; };
00196
00197   template<typename T1, typename T2, typename T = void>
00198   struct Enable_if_same : Enable_if<Same<T1, T2>::value, T> {};
00199
00200   template<typename T> struct Remove_const { typedef T type; };
00201   template<typename T> struct Remove_const<T const> { typedef T type; };
00202   template<typename T> struct Remove_volatile { typedef T type; };
00203   template<typename T> struct Remove_volatile<T volatile> { typedef T type; };
00204   template<typename T> struct Remove_cv
00205   { typedef typename Remove_const<typename Remove_volatile<T>::type>::type
      type; };
00206
00207   template<typename T> struct Remove_pointer { typedef T type; };
00208   template<typename T> struct Remove_pointer<T*> { typedef T type; };
00209   template<typename T> struct Remove_reference { typedef T type; };
00210   template<typename T> struct Remove_reference<T&> { typedef T type; };
00211   template<typename T> struct Remove_pr { typedef T type; };
00212   template<typename T> struct Remove_pr<T&> { typedef T type; };
00213   template<typename T> struct Remove_pr<T*> { typedef T type; };
00214 } // Types
00215 } // L4
```

15.301 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++ debugger interface.

Namespaces

- L4

  L4 low-level kernel interface.

15.301.1 Detailed Description

The debugger interface specifies common debugging related definitions.

Definition in file debugger.

15.302 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
```

```
 * 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/debugger.h>
#include <l4/sys/kobject>

namespace L4 {

class Debugger : public Kobject_t<Debugger, Kobject, L4_PROTO_DEBUGGER>
{
public:
  enum
  {
    Switch_log_on  = L4_DEBUGGER_SWITCH_LOG_ON,
    Switch_log_off = L4_DEBUGGER_SWITCH_LOG_OFF,
  };

  l4_msgtag_t set_object_name(const char *name,
                              l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_debugger_set_object_name_u(cap(), name, utcb); }

  unsigned long global_id(l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_debugger_global_id_u(cap(), utcb); }

  unsigned long kobj_to_id(l4_addr_t kobjp,
                           l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_debugger_kobj_to_id_u(cap(), kobjp, utcb); }

  int query_log_typeid(const char *name, unsigned idx,
                       l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_debugger_query_log_typeid_u(cap(), name, idx, utcb); }

  int query_log_name(unsigned idx,
                     char *name, unsigned namelen,
                     char *shortname, unsigned shortnamelen,
                     l4_utcb_t *utcb = l4_utcb()) throw()
  {
    return l4_debugger_query_log_name_u(cap(), idx, name, namelen,
                                        shortname, shortnamelen, utcb);
  }

  l4_msgtag_t switch_log(const char *name, unsigned on_off,
                         l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_debugger_switch_log_u(cap(), name, on_off, utcb); }

  l4_msgtag_t get_object_name(unsigned id, char *name, unsigned size,
                      l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_debugger_get_object_name_u(cap(), id, name, size, utcb); }
};
}
```

15.303 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.

- int 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.

- int 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.

15.303.1 Detailed Description

Debugger related definitions.

Definition in file debugger.h.

15.303.2 Function Documentation

15.303.2.1 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 366 of file debugger.h.

15.303.2.2 l4_debugger_query_log_name()

```
int 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 350 of file debugger.h.

15.303.2.3 l4_debugger_query_log_typeid()

```
int 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 343 of file debugger.h.

15.303.2.4 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 359 of file debugger.h.

15.304 debugger.h

```
00001 #pragma once
00002
00007 /*
00008  * (c) 2008-2011 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
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/utcb.h>
00028 #include <l4/sys/ipc.h>
00029
00053 L4_INLINE l4_msgtag_t
00054 l4_debugger_set_object_name(l4_cap_idx_t cap, const char *name)
      L4_NOTHROW;
00055
00059 L4_INLINE l4_msgtag_t
00060 l4_debugger_set_object_name_u(l4_cap_idx_t cap, const char *name,
      l4_utcb_t *utcb) L4_NOTHROW;
00061
00073 L4_INLINE l4_msgtag_t
00074 l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
00075                             char *name, unsigned size) L4_NOTHROW;
00076
00080 L4_INLINE l4_msgtag_t
00081 l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
00082                               char *name, unsigned size,
00083                               l4_utcb_t *utcb) L4_NOTHROW;
00084
00096 L4_INLINE unsigned long
00097 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 int
l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
                             unsigned idx) L4_NOTHROW;

L4_INLINE int
l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
                               unsigned idx, l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE int
l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
                           char *name, unsigned namelen,
                           char *shortname, unsigned shortnamelen) L4_NOTHROW;

L4_INLINE int
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 l4_msgtag_t
l4_debugger_set_object_name_u(unsigned long cap,
                              const char *name, l4_utcb_t *utcb) L4_NOTHROW
{
  unsigned int i;
  char *s = (char *)&l4_utcb_mr_u(utcb)->mr[1];
  l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_SET_OP;
  for (i = 0;
       *name && i < (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t) - 1;
       ++i, ++name, ++s)
    *s = *name;
  *s = 0;
  i = (i + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
  return l4_invoke_debugger(cap, l4_msgtag(0, i + 1, 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 int
l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
                               unsigned idx,
                               l4_utcb_t *utcb) L4_NOTHROW
{
  unsigned l;
  int e;
  l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_TYPEID_OP;
  l4_utcb_mr_u(utcb)->mr[1] = idx;
  l = __builtin_strlen(name);
  l = l > 31 ? 31 : l;
  __builtin_strncpy((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 31);
  l = (l + 1 + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
  e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2 + l, 0, 0), utcb), utcb);
  if (e < 0)
    return e;
  return l4_utcb_mr_u(utcb)->mr[0];
}

L4_INLINE int
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
{
  int e;
  char *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 *)&l4_utcb_mr_u(utcb)->mr[0];
  __builtin_strncpy(name, n, namelen);
  name[namelen - 1] = 0;
  __builtin_strncpy(shortname, n + __builtin_strlen(n) + 1, shortnamelen);
  shortname[shortnamelen - 1] = 0;
  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 l;
  l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_SWITCH_LOG_OP;
  l4_utcb_mr_u(utcb)->mr[1] = on_off;
  l = __builtin_strlen(name);
  l = l > 31 ? 31 : l;
  __builtin_strncpy((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 31);
  l = (l + 1 + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
  return l4_invoke_debugger(cap, l4_msgtag(0, 2 + l, 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);
  __builtin_strncpy(name, (char *)&l4_utcb_mr_u(utcb)->mr[0], size);
  name[size - 1] = 0;
  return t;
}


L4_INLINE l4_msgtag_t
l4_debugger_set_object_name(unsigned long 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 int
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 int
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());
}
```

15.305 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.

15.305.1 Detailed Description

Error codes.

Definition in file err.h.

15.306 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
```

15.307 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.

15.307.1 Detailed Description

Exception C++ interface.

Definition in file exception.

15.308 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
00035 class L4_EXPORT Exception :
00036   public Kobject_0t<Exception, L4_PROTO_EXCEPTION>
00037 {
00038 public:
00039   // TODO: pass a reference/pointer to the UTCB not copy the regs
00050   L4_INLINE_RPC(
00051       l4_msgtag_t, exception, (L4::Ipc::In_out<l4_exc_regs_t *>
      regs,
00052                                L4::Ipc::Rcv_fpage rwin,
00053                                L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00054
00055   typedef L4::Typeid::Rpc_nocode<exception_t>
      Rpcs;
00056 };
00057
00058 }
```

15.309 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 to create kernel objects.
- 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.

15.309.1 Detailed Description

Common factory related definitions.

Definition in file factory.

15.310 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
00055 class Factory : public Kobject_t<Factory, Kobject, L4_PROTO_FACTORY>
00056 {
00057 public:
00058
00059   typedef l4_mword_t Proto;
00060
00064   struct Nil {};
00065
00071   struct Lstr
00072   {
00076     char const *s;
00077
00081     int len;
00082
00087     Lstr(char const *s, int len) : s(s), len(len) {}
00088   };
00089
00096   class S
00097   {
00098   private:
00099     l4_utcb_t *u;
00100     l4_msgtag_t t;
00101     l4_cap_idx_t f;
00102
00103   public:
00109     S(S const &o)
00110     : u(o.u), t(o.t), f(o.f)
00111     { const_cast<S&>(o).t.raw = 0; }
00112
00124     S(l4_cap_idx_t f, long obj, L4::Cap<void> target,
00125       l4_utcb_t *utcb) throw()
00126     : u(utcb), t(l4_factory_create_start_u(obj, target.cap(), u)), f(f)
00127     {}
00128
00133     ~S()
00134     {
00135       if (t.raw)
00136   l4_factory_create_commit_u(f, t, u);
00137     }
00138
00144     operator l4_msgtag_t ()
00145     {
```

```
      l4_msgtag_t r = l4_factory_create_commit_u(f, t, u);
      t.raw = 0;
      return r;
    }

    S &operator  << (l4_mword_t i)
    {
      l4_factory_create_add_int_u(i, &t, u);
      return *this;
    }

    S &operator  << (l4_umword_t i)
    {
      l4_factory_create_add_uint_u(i, &t, u);
      return *this;
    }

    S &operator << (char const *s)
    {
      l4_factory_create_add_str_u(s, &t, u);
      return *this;
    }

    S &operator << (Lstr const &s)
    {
      l4_factory_create_add_lstr_u(s.s, s.len, &t, u);
      return *this;
    }

    S &operator << (Nil)
    {
      l4_factory_create_add_nil_u(&t, u);
      return *this;
    }

    S &operator << (l4_fpage_t d)
    {
      l4_factory_create_add_fpage_u(d, &t, u);
      return *this;
    }
  };


public:

  S create(Cap<void> target, long obj, l4_utcb_t *utcb = l4_utcb()) throw()
  {
    return S(cap(), obj, target, utcb);
  }

  template<typename OBJ>
  S create(Cap<OBJ> target, l4_utcb_t *utcb = l4_utcb()) throw()
  {
    return S(cap(), OBJ::Protocol, target, utcb);
  }

  L4_INLINE_RPC_NF(
      l4_msgtag_t, create, (L4::Ipc::Out< L4::Cap<void> > target, l4_mword_t obj,
                            L4::Ipc::Varg const *args),
      L4::Ipc::Call_t<L4_CAP_FPAGE_S>);

  l4_msgtag_t create_task(Cap<Task> const & target_cap,
                          l4_fpage_t const &utcb_area,
                          l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_factory_create_task_u(cap(), target_cap.cap(), utcb_area, utcb); }

  l4_msgtag_t create_thread(Cap<Thread> const &target_cap,
                            l4_utcb_t *utcb = l4_utcb()) throw()
    L4_DEPRECATED("Call create with Cap<Thread> as argument instead.")
  { return l4_factory_create_thread_u(cap(), target_cap.cap(), utcb); }

  l4_msgtag_t create_factory(Cap<Factory> const &target_cap,
                             unsigned long limit,
                             l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_factory_create_factory_u(cap(), target_cap.cap(), limit, utcb); }

  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()) throw()
  { return l4_factory_create_gate_u(cap(), target_cap.cap(), thread_cap.cap(), label, utcb); }

```

```
  l4_msgtag_t create_irq(Cap<Irq>const &target_cap,
                         l4_utcb_t *utcb = l4_utcb()) throw()
    L4_DEPRECATED("Call create with Cap<Irq> as argument instead.")
  { return l4_factory_create_irq_u(cap(), target_cap.
cap(), utcb); }

  l4_msgtag_t create_vm(Cap<Vm>const &target_cap,
                        l4_utcb_t *utcb = l4_utcb()) throw()
    L4_DEPRECATED("Call create with Cap<Vm> as argument instead.")
  { return l4_factory_create_vm_u(cap(), target_cap.cap(), utcb); }

  typedef L4::Typeid::Rpc_nocode<create_t> Rpcs;
};

}
```

15.311 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 const utcb_area) L4_NOTHROW

  Create a new task.
- l4_msgtag_t l4_factory_create_task_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_fpage_t const utcb_area, l4_utcb_t *utcb) 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_thread_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) 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_factory_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit, l4_utcb_t *utcb) 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_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

  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_irq_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW

  Create a new IRQ.
- 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_vm_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW

  Create a new virtual machine.

15.311.1 Detailed Description

Common factory related definitions.

Definition in file factory.h.

15.312 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  *
```

```
 * 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/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>

L4_INLINE l4_msgtag_t
l4_factory_create_task(l4_cap_idx_t factory,
                       l4_cap_idx_t target_cap, l4_fpage_t const utcb_area) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_factory_create_task_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
                         l4_fpage_t const utcb_area, l4_utcb_t *utcb) L4_NOTHROW;

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, int 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_FPAGE_RWX).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 const 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, int len, 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);
  char *c;
  int i;

  if (w + 1 + (len + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t)
      > 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 + (len + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);

  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());
}
```

15.313 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:

15.313.1 Detailed Description

Interrupt controller.

Definition in file icu.

15.314 icu

```
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 #pragma once
00025
00026 #include <l4/sys/irq>
```

15.315 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 capabilites of the ICU.
- 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 capabilities of the ICU.
- 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

  Acknowledge 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.

15.315.1 Detailed Description

Interrupt controller.

Definition in file icu.h.

15.316 icu.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  *               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/kernel_object.h>
00028 #include <l4/sys/ipc.h>
00029
00050 enum L4_icu_flags
00051 {
00059   L4_ICU_FLAG_MSI = 0x80000000,
00060 };
00061
00062
00067 enum L4_irq_mode
00068 {
00070   L4_IRQ_F_NONE         = 0,
00071   L4_IRQ_F_LEVEL        = 0x2,
00072   L4_IRQ_F_EDGE         = 0x0,
00073   L4_IRQ_F_POS          = 0x0,
00074   L4_IRQ_F_NEG          = 0x4,
00075   L4_IRQ_F_BOTH         = 0x8,
00076   L4_IRQ_F_LEVEL_HIGH   = 0x3,
00077   L4_IRQ_F_LEVEL_LOW    = 0x7,
00078   L4_IRQ_F_POS_EDGE     = 0x1,
00079   L4_IRQ_F_NEG_EDGE     = 0x5,
00080   L4_IRQ_F_BOTH_EDGE    = 0x9,
00081   L4_IRQ_F_MASK         = 0xf,
00084   L4_IRQ_F_SET_WAKEUP   = 0x10,
00085   L4_IRQ_F_CLEAR_WAKEUP = 0x20,
00086 };
00087
00088
00093 enum L4_icu_opcode
00094 {
00100   L4_ICU_OP_BIND = 0,
00101
00107   L4_ICU_OP_UNBIND = 1,
00108
00114   L4_ICU_OP_INFO = 2,
00115
00121   L4_ICU_OP_MSI_INFO = 3,
00122
00128   L4_ICU_OP_UNMASK   = 4,
00129
00135   L4_ICU_OP_MASK     = 5,
00136
00142   L4_ICU_OP_SET_MODE = 6,
00143 };
00144
00145 enum L4_icu_ctl_op
00146 {
00147   L4_ICU_CTL_UNMASK = 0,
00148   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_FPAGE_RWX).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_FPAGE_RWX).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());
}
```

15.317 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_RE↩MSGCUT = 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.

- 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_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

  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.

15.317.1 Detailed Description

Common IPC interface.

Definition in file ipc.h.

15.317.2 Function Documentation

15.317.2.1 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 441 of file ipc.h.

References L4_EIPC_LO.

15.318 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
00056 /******************************************************************************
00057  *** IPC result checking
00058  ******************************************************************************/
00059
00075 enum l4_ipc_tcr_error_t
00076 {
00077   L4_IPC_ERROR_MASK       = 0x1F,
00078   L4_IPC_SND_ERR_MASK     = 0x01,
00080   L4_IPC_ENOT_EXISTENT    = 0x04,
00083   L4_IPC_RETIMEOUT        = 0x03,
00086   L4_IPC_SETIMEOUT        = 0x02,
00089   L4_IPC_RECANCELED       = 0x07,
00092   L4_IPC_SECANCELED       = 0x06,
00095   L4_IPC_REMAPFAILED      = 0x11,
00099   L4_IPC_SEMAPFAILED      = 0x10,
00102   L4_IPC_RESNDPFTO        = 0x0b,
00106   L4_IPC_SESNDPFTO        = 0x0a,
00110   L4_IPC_RERCVPFTO        = 0x0d,
00114   L4_IPC_SERCVPFTO        = 0x0c,
00118   L4_IPC_REABORTED        = 0x0f,
00121   L4_IPC_SEABORTED        = 0x0e,
00124   L4_IPC_REMSGCUT         = 0x09,
00128   L4_IPC_SEMSGCUT         = 0x08,
00132 };
00133
00134
00145 L4_INLINE l4_umword_t
00146 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb)
      L4_NOTHROW;
00147
00148
00155 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__ */
```

15.319 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.

15.319.1 Detailed Description

L4 IPC System Calls, ARM.

Definition in file ipc.h.

15.320 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
```

```
#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__
```

15.321 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:

15.321.1 Detailed Description

L4 IPC System Calls, x86.

Definition in file ipc.h.

15.322 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__ */
```

15.323 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.

Namespaces

- L4

  L4 low-level kernel interface.

15.323.1 Detailed Description

The C++ IPC gate interface.

Definition in file ipc_gate.

15.324 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
00056 class L4_EXPORT Ipc_gate :
00057   public Kobject_t<Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT,
00058                    Type_info::Demand_t<1> >
00059 {
00060 public:
00068   L4_INLINE_RPC_OP(L4_IPC_GATE_GET_INFO_OP,
00069       l4_msgtag_t, get_infos, (l4_umword_t *label));
00070
00071   typedef L4::Typeid::Rpcs_sys<bind_thread_t, get_infos_t>
      Rpcs;
00072 };
00073
00074 }
```

15.325 l4/sys/ipc_gate.h File Reference

The C IPC gate 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_bind_thread (l4_cap_idx_t gate, l4_cap_idx_t thread, l4_umword_t label)

  Bind the IPC gate to a thread.
- l4_msgtag_t l4_ipc_gate_get_infos (l4_cap_idx_t gate, l4_umword_t *label)

  Get information about the IPC-gate.

15.325.1 Detailed Description

The C IPC gate interface.

IPC gates are used to create secure communication channels between threads. An IPC gate object can be created using the Factory interface. With l4_ipc_gate_bind_thread() a thread is bound to an IPC gate which then receives all messages sent to that IPC gate.

The l4_ipc_gate_bind_thread() call allows to assign each IPC gate a kernel protected, machine-word sized payload called a *label*. It securely identifies the gate. The lower two bits of the *label* can be used to encode rights bits. The kernel combines these bits with the capability rights, so a programmer usually should not pick the lower two bits for the *label*. The *label* is only visible in the task which is running the thread the IPC gate was bound to and cannot be altered by the sender.

Definition in file ipc_gate.h.

15.326 ipc_gate.h

```
00001
00018 /*
00019  * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00020  *               Alexander Warg <warg@os.inf.tu-dresden.de>
00021  *     economic rights: Technische Universität Dresden (Germany)
00022  *
00023  * This file is part of TUD:OS and distributed under the terms of the
00024  * GNU General Public License 2.
00025  * Please see the COPYING-GPL-2 file for details.
00026  *
00027  * As a special exception, you may use this file as part of a free software
00028  * library without restriction.  Specifically, if other files instantiate
00029  * templates or use macros or inline functions from this file, or you compile
00030  * this file and link it with other files to produce an executable, this
00031  * file does not by itself cause the resulting executable to be covered by
00032  * the GNU General Public License.  This exception does not however
00033  * invalidate any other reasons why the executable file might be covered by
00034  * the GNU General Public License.
00035  */
00036 #pragma once
00037
00038 #include <l4/sys/utcb.h>
00039 #include <l4/sys/types.h>
00040 #include <l4/sys/rcv_endpoint.h>
00041
00058 L4_INLINE l4_msgtag_t
00059 l4_ipc_gate_bind_thread(l4_cap_idx_t gate,
      l4_cap_idx_t thread,
00060                         l4_umword_t label)
00061   L4_DEPRECATED("Use l4_rcv_ep_bind_thread().");
00062
00067 L4_INLINE l4_msgtag_t
00068 l4_ipc_gate_bind_thread_u(l4_cap_idx_t gate, l4_cap_idx_t thread,
00069                           l4_umword_t label, l4_utcb_t *utcb)
00070   L4_DEPRECATED("Use l4_rcv_ep_bind_thread_u().");
00071
00078 L4_INLINE l4_msgtag_t
00079 l4_ipc_gate_get_infos(l4_cap_idx_t gate,
      l4_umword_t *label);
00080
00085 L4_INLINE l4_msgtag_t
00086 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label,
      l4_utcb_t *utcb);
```

```
00087
00094 enum L4_ipc_gate_ops
00095 {
00096   L4_IPC_GATE_BIND_OP      = 0x10,
00097   L4_IPC_GATE_GET_INFO_OP = 0x11,
00098 };
00099
00100
00101 /* IMPLEMENTATION ---------------------------------------------------------*/
00102
00103 #include <l4/sys/ipc.h>
00104
00105 L4_INLINE l4_msgtag_t
00106 l4_ipc_gate_bind_thread_u(l4_cap_idx_t gate,
00107                           l4_cap_idx_t thread, l4_umword_t label,
00108                           l4_utcb_t *utcb)
00109 {
00110   return l4_rcv_ep_bind_thread_u(gate, thread, label, utcb);
00111 }
00112
00113 L4_INLINE l4_msgtag_t
00114 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label,
      l4_utcb_t *utcb)
00115 {
00116   l4_msgtag_t tag;
00117   l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00118   m->mr[0] = L4_IPC_GATE_GET_INFO_OP;
00119   tag = l4_ipc_call(gate, utcb, l4_msgtag(L4_PROTO_KOBJECT, 1, 0, 0),
00120                     L4_IPC_NEVER);
00121   if (!l4_msgtag_has_error(tag) && l4_msgtag_label(tag) >= 0)
00122     *label = m->mr[0];
00123
00124   return tag;
00125 }
00126
00127
00128
00129 L4_INLINE l4_msgtag_t
00130 l4_ipc_gate_bind_thread(l4_cap_idx_t gate,
      l4_cap_idx_t thread,
00131                         l4_umword_t label)
00132 {
00133   return l4_rcv_ep_bind_thread_u(gate, thread, label, l4_utcb());
00134 }
00135
00136 L4_INLINE l4_msgtag_t
00137 l4_ipc_gate_get_infos(l4_cap_idx_t gate,
      l4_umword_t *label)
00138 {
00139   return l4_ipc_gate_get_infos_u(gate, label, l4_utcb());
00140 }
```

15.327 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 acknowledge message to an object.
- struct L4::Triggerable

  Interface that allows an object to be triggered by some source.
- class L4::Irq

  C++ Irq interface.
- struct L4::Irq_mux

  IRQ multiplexer for shared IRQs.
- class L4::Icu

  C++ Icu interface.
- class L4::Icu::Info

  This class encapsulates information about an ICU.

Namespaces

- L4

  L4 low-level kernel interface.

15.327.1 Detailed Description

C++ Irq interface.

Definition in file irq.

15.328 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
00041 class Irq_eoi : public Kobject_0t<Irq_eoi, L4::PROTO_EMPTY>
00042 {
00043 public:
00063   l4_msgtag_t unmask(unsigned irqnum, l4_umword_t *label = 0,
00064                      l4_timeout_t to = L4_IPC_NEVER,
00065                      l4_utcb_t *utcb = l4_utcb()) throw()
00066   {
00067     return l4_icu_control_u(cap(), irqnum, L4_ICU_CTL_UNMASK, label, to, utcb);
00068   }
00069 };
00070
00076 struct Triggerable : Kobject_t<Triggerable, Irq_eoi, L4_PROTO_IRQ>
00077 {
00091   l4_msgtag_t trigger(l4_utcb_t *utcb = l4_utcb()) throw()
00092   { return l4_irq_trigger_u(cap(), utcb); }
00093 };
00094
00112 class Irq : public Kobject_2t<Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER>
00113 {
00114 public:
00115   using Triggerable::unmask;
00116
00133   l4_msgtag_t attach(l4_umword_t label,
00134                      Cap<Thread> const &thread = Cap<Thread>::Invalid,
00135                      l4_utcb_t *utcb = l4_utcb()) throw()
00136     L4_DEPRECATED("Use bind_thread(thread, label).")
00137   {
00138 #pragma GCC diagnostic push
00139 #pragma GCC diagnostic ignored "-Wdeprecated-declarations"
00140     return l4_irq_attach_u(cap(), label, thread.cap(), utcb);
```

```
#pragma GCC diagnostic pop
  }

  l4_msgtag_t detach(l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_irq_detach_u(cap(), utcb); }


  l4_msgtag_t receive(l4_timeout_t timeout = L4_IPC_NEVER,
                      l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_irq_receive_u(cap(), timeout, utcb); }

  l4_msgtag_t wait(l4_umword_t *label, l4_timeout_t timeout = L4_IPC_NEVER,
                   l4_utcb_t *utcb = l4_utcb()) throw()
  { return unmask(-1, label, timeout, utcb); }

  l4_msgtag_t unmask(l4_utcb_t *utcb = l4_utcb()) throw()
  { return unmask(-1, 0, L4_IPC_NEVER, utcb); }
};

struct Irq_mux : Kobject_t<Irq_mux, Triggerable, L4_PROTO_IRQ_MUX>
{
  l4_msgtag_t chain(Cap<Triggerable> const &slave,
                    l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_irq_mux_chain_u(cap(), slave.cap(), utcb); }
};


class Icu :
  public Kobject_t<Icu, Irq_eoi, L4_PROTO_IRQ,
                   Type_info::Demand_t<1> >
{
public:
  enum Mode
  {
    F_none         = L4_IRQ_F_NONE,
    F_level_high   = L4_IRQ_F_LEVEL_HIGH,
    F_level_low    = L4_IRQ_F_LEVEL_LOW,
    F_pos_edge     = L4_IRQ_F_POS_EDGE,
    F_neg_edge     = L4_IRQ_F_NEG_EDGE,
    F_both_edge    = L4_IRQ_F_BOTH_EDGE,
    F_mask         = L4_IRQ_F_MASK,

    F_set_wakeup   = L4_IRQ_F_SET_WAKEUP,
    F_clear_wakeup = L4_IRQ_F_CLEAR_WAKEUP,
  };

  enum Flags
  {
    F_msi = L4_ICU_FLAG_MSI
  };

  class Info : public l4_icu_info_t
  {
  public:
    bool supports_msi() const { return features & F_msi; }
  };

  l4_msgtag_t bind(unsigned irqnum, L4::Cap<Triggerable> irq,
                   l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_icu_bind_u(cap(), irqnum, irq.cap(), utcb); }

  L4_RPC_NF_OP(L4_ICU_OP_BIND,
               l4_msgtag_t, bind, (l4_umword_t irqnum, Ipc::Cap<Irq> irq));

  l4_msgtag_t unbind(unsigned irqnum, L4::Cap<Triggerable> irq,
                     l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_icu_unbind_u(cap(), irqnum, irq.cap(), utcb); }

  L4_RPC_NF_OP(L4_ICU_OP_UNBIND,
               l4_msgtag_t, unbind, (l4_umword_t irqnum, Ipc::Cap<Irq> irq));

  l4_msgtag_t info(l4_icu_info_t *info, l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_icu_info_u(cap(), info, utcb); }

  struct _Info { l4_umword_t features, nr_irqs, nr_msis; };
  L4_RPC_NF_OP(L4_ICU_OP_INFO, l4_msgtag_t, info, (_Info *info));

  L4_INLINE_RPC_OP(L4_ICU_OP_MSI_INFO,
      l4_msgtag_t, msi_info, (l4_umword_t irqnum, l4_uint64_t source,
                              l4_icu_msi_info_t *msi_info));

```

```
  l4_msgtag_t control(unsigned irqnum, unsigned op, l4_umword_t *label,
                      l4_timeout_t to, l4_utcb_t *utcb =
l4_utcb()) throw()
  { return l4_icu_control_u(cap(), irqnum, op, label, to, utcb); }

  l4_msgtag_t mask(unsigned irqnum,
                   l4_umword_t *label = 0,
                   l4_timeout_t to = L4_IPC_NEVER,
                   l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_icu_mask_u(cap(), irqnum, label, to, utcb); }

  L4_RPC_NF_OP(L4_ICU_OP_MASK, l4_msgtag_t, mask, (
l4_umword_t irqnum),
               L4::Ipc::Send_only);


  L4_RPC_NF_OP(L4_ICU_OP_UNMASK, l4_msgtag_t,
unmask, (l4_umword_t irqnum),
               L4::Ipc::Send_only);

  l4_msgtag_t set_mode(unsigned irqnum, l4_umword_t mode,
                       l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_icu_set_mode_u(cap(), irqnum, mode, utcb); }

  L4_RPC_NF_OP(L4_ICU_OP_SET_MODE,
               l4_msgtag_t, set_mode, (l4_umword_t irqnum,
l4_umword_t mode));

  typedef L4::Typeid::Rpcs_sys<
    bind_t, unbind_t, info_t, msi_info_t, unmask_t, mask_t, set_mode_t
  > Rpcs;
};

}
```

15.329 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:

15.329.1 Detailed Description

Kernel object system calls.

Definition in file kernel_object.h.

15.330 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
```

```
 * 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__ */
```

15.331 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.

15.331.1 Detailed Description

L4::Kip class, memory descriptors.

Author

Alexander Warg `alexander.warg@os.inf.tu-dresden.de`

Definition in file kip.

15.332 kip

```
// vim:set ft=cpp: -*- Mode: C++ -*-
/*
 * (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.
 *
 * 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_KIP_H__
#define L4_SYS_KIP_H__

#include <l4/cxx/static_vector>

/* C++ version of memory descriptors */

namespace L4
{
  namespace Kip
  {
    class Mem_desc
    {
    public:
      enum Mem_type
      {
        Undefined    = 0x0,
        Conventional = 0x1,
        Reserved     = 0x2,
        Dedicated    = 0x3,
        Shared       = 0x4,

        Info         = 0xd,
        Bootloader   = 0xe,
        Arch         = 0xf
      };

      enum Info_sub_type
      {
        Info_acpi_rsdp = 0
      };

    private:
      unsigned long _l, _h;

      static unsigned long &memory_info(void *kip) throw()
      { return *((unsigned long *)kip + 21); }

      static unsigned long memory_info(void const *kip) throw()
      { return *((unsigned long const *)kip + 21); }

    public:
      static Mem_desc *first(void *kip) throw()
      {
        return (Mem_desc *)((char *)kip
               + (memory_info(kip) >> ((sizeof(unsigned long) / 2) * 8)));
      }

      static Mem_desc const *first(void const *kip) throw()
      {
        return (Mem_desc const *)((char const *)kip
               + (memory_info(kip) >> ((sizeof(unsigned long) / 2) * 8)));
      }

      static unsigned long count(void const *kip) throw()
      {
        return memory_info(kip)
               & ((1UL << ((sizeof(unsigned long) / 2) * 8)) - 1);
      }

      static void count(void *kip, unsigned count) throw()
      {
        unsigned long &mi = memory_info(kip);
        mi = (mi & ~((1UL << ((sizeof(unsigned long) / 2) * 8)) - 1)) | count;
      }

```

```
      static inline cxx::static_vector<Mem_desc const>
    all(void const *kip)
      {
        return cxx::static_vector<Mem_desc const>(
    Mem_desc::first(kip),
                                                Mem_desc::count(kip));
      }

      static inline cxx::static_vector<Mem_desc> all(void *kip)
      {
        return cxx::static_vector<Mem_desc>(
    Mem_desc::first(kip),
                                          Mem_desc::count(kip));
      }

      Mem_desc(unsigned long start, unsigned long end,
               Mem_type t, unsigned char st = 0, bool virt = false) throw()
      : _l((start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
           | (virt ? 0x0200 : 0x0)), _h(end | 0x3ffUL)
      {}

      unsigned long start() const throw() { return _l & ~0x3ffUL; }

      unsigned long end() const throw() { return _h | 0x3ffUL; }

      unsigned long size() const throw() { return end() + 1 - start(); }

      Mem_type type() const throw() { return (Mem_type)(_l & 0x0f); }

      unsigned char sub_type() const throw() { return (_l >> 4) & 0x0f; }

      unsigned is_virtual() const throw() { return _l & 0x200; }

      void set(unsigned long start, unsigned long end,
               Mem_type t, unsigned char st = 0, bool virt = false) throw()
      {
        _l = (start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
             | (virt?0x0200:0x0);

        _h = end | 0x3ffUL;
      }

    };
  };
};

#endif
```

15.333 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:

Data Structures

- struct l4_tracebuffer_status_window_t

  Trace-buffer status window descriptor.
- struct l4_tracebuffer_status_t

  Trace-buffer status.

Enumerations

- enum {
  LOG_EVENT_CONTEXT_SWITCH = 0, LOG_EVENT_IPC_SHORTCUT = 1, LOG_EVENT_IRQ_RAISED = 2, LOG_EVENT_TIMER_IRQ = 3,
  LOG_EVENT_THREAD_EX_REGS = 4, LOG_EVENT_MAX_EVENTS = 16 }

  Log event types.

Functions

- l4_tracebuffer_status_t ∗ fiasco_tbuf_get_status (void)

  Return trace-buffer status.
- l4_addr_t fiasco_tbuf_get_status_phys (void)

  Return the physical address of the trace-buffer status struct.
- 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.

15.333.1 Detailed Description

L4 kernel event tracing.

Definition in file ktrace.h.

15.334 ktrace.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  *     2015      Adam Lackorzynski <adam@l4re.org>
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_KTRACE_H__
00028 #define __L4_KTRACE_H__
00029
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/ktrace_events.h>
00032
00037 enum
00038 {
00039   LOG_EVENT_CONTEXT_SWITCH = 0,
00040   LOG_EVENT_IPC_SHORTCUT   = 1,
00041   LOG_EVENT_IRQ_RAISED     = 2,
00042   LOG_EVENT_TIMER_IRQ      = 3,
00043   LOG_EVENT_THREAD_EX_REGS = 4,
00044   LOG_EVENT_MAX_EVENTS     = 16,
00045 };
00046
00051 // keep in sync with fiasco/src/jabi/jdb_ktrace.cpp
00052 typedef struct
00053 {
00055   l4_tracebuffer_entry_t *tracebuffer;
00057   l4_umword_t size;
00059   volatile l4_uint64_t version;
00060 } l4_tracebuffer_status_window_t;
00061
00066 // keep in sync with fiasco/src/jabi/jdb_ktrace.cpp
00067 typedef struct
00068 {
00069   l4_tracebuffer_status_window_t window[2];
00071   volatile l4_tracebuffer_entry_t * current_entry;
00073   l4_uint32_t logevents[LOG_EVENT_MAX_EVENTS];
```

```
00074
00076   l4_uint32_t scaler_tsc_to_ns;
00078   l4_uint32_t scaler_tsc_to_us;
00080   l4_uint32_t scaler_ns_to_tsc;
00081
00083   volatile l4_uint32_t cnt_context_switch;
00085   volatile l4_uint32_t cnt_addr_space_switch;
00087   volatile l4_uint32_t cnt_shortcut_failed;
00089   volatile l4_uint32_t cnt_shortcut_success;
00091   volatile l4_uint32_t cnt_irq;
00093   volatile l4_uint32_t cnt_ipc_long;
00095   volatile l4_uint32_t cnt_page_fault;
00098   volatile l4_uint32_t cnt_io_fault;
00100   volatile l4_uint32_t cnt_task_create;
00102   volatile l4_uint32_t cnt_schedule;
00106   volatile l4_uint32_t cnt_iobmap_tlb_flush;
00107
00108 } l4_tracebuffer_status_t;
00109
00116 L4_INLINE l4_tracebuffer_status_t *
00117 fiasco_tbuf_get_status(void);
00118
00125 L4_INLINE l4_addr_t
00126 fiasco_tbuf_get_status_phys(void);
00127
00135 L4_INLINE l4_umword_t
00136 fiasco_tbuf_log(const char *text);
00137
00149 L4_INLINE l4_umword_t
00150 fiasco_tbuf_log_3val(const char *text, l4_umword_t v1,
      l4_umword_t v2, l4_umword_t v3);
00151
00159 L4_INLINE l4_umword_t
00160 fiasco_tbuf_log_binary(const unsigned char *data);
00161
00166 L4_INLINE void
00167 fiasco_tbuf_clear(void);
00168
00173 L4_INLINE void
00174 fiasco_tbuf_dump(void);
00175
00176 #include <l4/sys/__ktrace-impl.h>
00177
00178 #endif
```

15.335 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.

15.335.1 Detailed Description

Fixed sized integer types, generic version.

Definition in file l4int.h.

15.336 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;
00046 //do-we-need-this?//typedef unsigned long           l4_offs_t;    /**< Address offset type \ingroup
       l4_basic_types */
00047
00048
00049 typedef signed long             l4_mword_t;
00052 typedef unsigned long           l4_umword_t;
00059 typedef l4_uint64_t l4_cpu_time_t;
00060
00065 typedef l4_uint64_t l4_kernel_clock_t;
00066
00067 #endif /* !__L4_SYS_L4INT_H__ */
```

15.337 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.

15.337.1 Detailed Description

Fixed sized integer types, arm version.

Definition in file l4int.h.

15.338 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
00032
00033 #define L4_MWORD_BITS           32
00035 typedef unsigned int            l4_size_t;
00036 typedef signed int              l4_ssize_t;
00038
```

15.339 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.

15.339.1 Detailed Description

Fixed sized integer types, amd64 version.

Definition in file l4int.h.

15.340 l4int.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_next <l4/sys/l4int.h>


#define L4_MWORD_BITS           64
typedef unsigned long           l4_size_t;
typedef signed long             l4_ssize_t;

```

15.341 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.

15.341.1 Detailed Description

Fixed sized integer types, x86 version.

Definition in file l4int.h.

15.342 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
00032
00033 #define L4_MWORD_BITS           32
00035 typedef unsigned int            l4_size_t;
00036 typedef signed int              l4_ssize_t;
00038
```

15.343 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.

15.343.1 Detailed Description

Memory description functions.

Definition in file memdesc.h.

15.344 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,
```

```
};

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__ */
```

15.345 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.

15.345.1 Detailed Description

Meta interface for getting dynamic type information about objects behind capabilities.

Definition in file meta.

15.346 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 }
```

15.347 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.

15.347.1 Detailed Description

Pager and Io_pager C++ interface.

Definition in file pager.

15.348 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
00035 class L4_EXPORT Io_pager :
00036   public Kobject_0t<Io_pager, L4_PROTO_IO_PAGE_FAULT>
00037 {
00038 public:
00055   L4_INLINE_RPC(
00056       l4_msgtag_t, io_page_fault, (l4_fpage_t io_pfa,
      l4_umword_t pc,
00057                                    L4::Ipc::Opt<l4_mword_t &> result,
00058                                    L4::Ipc::Rcv_fpage rwin,
00059                                    L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp)
      );
00060
00061   typedef L4::Typeid::Rpc_nocode<io_page_fault_t>
      Rpcs;
00062 };
00063
00067 class L4_EXPORT Pager :
00068   public Kobject_t<Pager, Io_pager, L4_PROTO_PAGE_FAULT>
00069 {
00070 public:
00103   L4_INLINE_RPC(
00104       l4_msgtag_t, page_fault, (l4_umword_t pfa,
      l4_umword_t pc,
00105                                 L4::Ipc::Opt<l4_mword_t &> result,
00106                                 L4::Ipc::Rcv_fpage rwin,
00107                                 L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00108
00109   typedef L4::Typeid::Rpc_nocode<page_fault_t>
      Rpcs;
00110 };
00111
00112 }
```

15.349 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.

Namespaces

- L4

  L4 low-level kernel interface.

15.349.1 Detailed Description

Platform control object.

Definition in file platform_control.

15.350 platform_control

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00008  *          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/sys/capability>
00029 #include <l4/sys/platform_control.h>
00030 #include <l4/sys/cxx/ipc_iface>
00031
00032 namespace L4 {
00033
00046 class L4_EXPORT Platform_control
00047 : public Kobject_t<Platform_control, Kobject, L4_PROTO_PLATFORM_CTL>
00048 {
00049 public:
00051   enum Opcode
00052   {
00053     Suspend  = L4_PLATFORM_CTL_SYS_SUSPEND_OP,
00054     Shutdown = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
00055     Cpu_enable  = L4_PLATFORM_CTL_CPU_ENABLE_OP,
00056     Cpu_disable = L4_PLATFORM_CTL_CPU_DISABLE_OP
00057   };
00058
00065   L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SUSPEND_OP,
00066                    l4_msgtag_t, system_suspend, (l4_umword_t extras));
00067
00073   L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
00074                    l4_msgtag_t, system_shutdown, (l4_umword_t reboot));
00075
00083   L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ENABLE_OP,
00084                    l4_msgtag_t, cpu_enable, (l4_umword_t phys_id));
00085
00093   L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_DISABLE_OP,
00094                    l4_msgtag_t, cpu_disable, (l4_umword_t phys_id));
00095
00096   typedef L4::Typeid::Rpcs_sys<system_suspend_t, system_shutdown_t,
00097                                cpu_enable_t, cpu_disable_t> Rpcs;
00098 };
00099
00100 }
00101
```

15.351 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_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_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.

15.351.1 Detailed Description

Platform control object.

Definition in file platform_control.h.

15.352 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
00051 L4_INLINE l4_msgtag_t
00052 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00053                                l4_umword_t extras) L4_NOTHROW;
00054
00058 L4_INLINE l4_msgtag_t
00059 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00060                                   l4_umword_t extras,
00061                                   l4_utcb_t *utcb) L4_NOTHROW;
00062
00063
00072 L4_INLINE l4_msgtag_t
00073 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00074                                 l4_umword_t reboot) L4_NOTHROW;
00075
00079 L4_INLINE l4_msgtag_t
00080 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00081                                    l4_umword_t reboot,
00082                                    l4_utcb_t *utcb) L4_NOTHROW;
00083
00092 L4_INLINE l4_msgtag_t
00093 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00094                            l4_umword_t phys_id) L4_NOTHROW;
00095
00099 L4_INLINE l4_msgtag_t
00100 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00101                              l4_umword_t phys_id,
00102                              l4_utcb_t *utcb) L4_NOTHROW;
00103
00112 L4_INLINE l4_msgtag_t
00113 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00114                             l4_umword_t phys_id) L4_NOTHROW;
00115
00119 L4_INLINE l4_msgtag_t
00120 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_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_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());
}
```

15.353 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.

15.353.1 Detailed Description

The C++ Receive endpoint interface.

Definition in file rcv_endpoint.

15.354 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:
00059   L4_INLINE_RPC_OP(L4_RCV_EP_BIND_OP,
00060       l4_msgtag_t, bind_thread, (Ipc::Opt<Ipc::Cap<Thread> > t,
      l4_umword_t label));
00061
00062   typedef L4::Typeid::Rpcs_sys<bind_thread_t>
      Rpcs;
00063 };
00064
00065 }
```

15.355 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 gate to a thread.

15.355.1 Detailed Description

Receive endpoint C interface.

Definition in file rcv_endpoint.h.

15.355.2 Enumeration Type Documentation

15.355.2.1 L4_rcv_ep_ops

```
enum L4_rcv_ep_ops
```

Receive endpoint operations.

Enumerator

L4_RCV_EP_BIND_OP	Bind operation.

Definition at line 56 of file rcv_endpoint.h.

15.356 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
00043 L4_INLINE l4_msgtag_t
00044 l4_rcv_ep_bind_thread(l4_cap_idx_t ep,
      l4_cap_idx_t thread,
00045                       l4_umword_t label);
00046
00051 L4_INLINE l4_msgtag_t
00052 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep, l4_cap_idx_t thread,
00053                         l4_umword_t label, l4_utcb_t *utcb);
00054
00056 enum L4_rcv_ep_ops
00057 {
00058   L4_RCV_EP_BIND_OP     = 0x10,
00059 };
00060
00061 /* IMPLEMENTATION ----------------------------------------------------------*/
00062
00063 #include <l4/sys/ipc.h>
00064
00065 L4_INLINE l4_msgtag_t
00066 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep,
00067                         l4_cap_idx_t thread, l4_umword_t label,
00068                         l4_utcb_t *utcb)
00069 {
00070   l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00071   m->mr[0] = L4_RCV_EP_BIND_OP;
00072   m->mr[1] = label;
00073   m->mr[2] = l4_map_obj_control(0, 0);
00074   m->mr[3] = l4_obj_fpage(thread, 0, L4_FPAGE_RWX).raw;
00075   return l4_ipc_call(ep, utcb, l4_msgtag(L4_PROTO_KOBJECT, 2, 1, 0),
00076                      L4_IPC_NEVER);
00077 }
00078
00079 L4_INLINE l4_msgtag_t
00080 l4_rcv_ep_bind_thread(l4_cap_idx_t ep,
      l4_cap_idx_t thread,
00081                       l4_umword_t label)
00082 {
00083   return l4_rcv_ep_bind_thread_u(ep, thread, label, l4_utcb());
00084 }
00085
00086
```

15.357 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.

Namespaces

- L4

  L4 low-level kernel interface.

15.357.1 Detailed Description

Scheduler object functions.

Definition in file scheduler.

15.358 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
00042 class L4_EXPORT Scheduler :
00043   public Kobject_t<Scheduler, Icu, L4_PROTO_SCHEDULER,
00044                    Type_info::Demand_t<1> >
00045 {
00046 public:
00047   // ABI function for 'info' call
00048   L4_INLINE_RPC_NF_OP(L4_SCHEDULER_INFO_OP,
00049       l4_msgtag_t, info, (l4_umword_t gran_offset,
      l4_umword_t *map,
00050                           l4_umword_t *cpu_max));
00051
00066   l4_msgtag_t info(l4_umword_t *cpu_max,
      l4_sched_cpu_set_t *cpus,
00067                    l4_utcb_t *utcb = l4_utcb()) const throw()
00068   {
00069     l4_umword_t max = 0;
00070     l4_msgtag_t t =
00071       info_t::call(c(), cpus->gran_offset, &cpus->map, &max, utcb);
00072     if (cpu_max) *cpu_max = max;
00073     return t;
00074   }
00075
00097   L4_INLINE_RPC_OP(L4_SCHEDULER_RUN_THREAD_OP,
00098       l4_msgtag_t, run_thread, (Ipc::Cap<Thread> thread,
      l4_sched_param_t const &sp));
00099
00126   L4_INLINE_RPC_OP(L4_SCHEDULER_IDLE_TIME_OP,
00127       l4_msgtag_t, idle_time, (l4_sched_cpu_set_t const &cpus,
00128                                l4_kernel_clock_t *us));
00129
00139   bool is_online(l4_umword_t cpu, l4_utcb_t *utcb =
      l4_utcb()) const throw()
00140   { return l4_scheduler_is_online_u(cap(), cpu, utcb); }
00141
00142   typedef L4::Typeid::Rpcs_sys<info_t, run_thread_t, idle_time_t>
       Rpcs;
00143 };
00144 }
```

15.359 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_ops { L4_SCHEDULER_INFO_OP = 0UL, L4_SCHEDULER_RUN_THREAD_OP = 1↩ UL, 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_cpu_time_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.

15.359.1 Detailed Description

Scheduler object functions.

Definition in file scheduler.h.

15.360 scheduler.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
```

```
 * 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>

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_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_cpu_time_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_cpu_time_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_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];

  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_FPAGE_RWX).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, 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, 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());
}
```

15.361 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

  Kernel-provided semaphore object.

Namespaces

- L4

  L4 low-level kernel interface.

15.361.1 Detailed Description

Semaphore class definition.

Definition in file semaphore.

15.362 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
00051 struct Semaphore : Kobject_t<Semaphore, Triggerable, L4_PROTO_SEMAPHORE>
00052 {
00065   l4_msgtag_t up(l4_utcb_t *utcb = l4_utcb()) throw()
00066   { return trigger(utcb); }
00067
00084   l4_msgtag_t down(l4_timeout_t timeout =
      L4_IPC_NEVER,
00085                    l4_utcb_t *utcb = l4_utcb()) throw()
00086   { return l4_semaphore_down_u(cap(), timeout, utcb);  }
00087 };
00088
00089 }
```

15.363 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) throw ()

  `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) throw ()

  `reinterpret_cast` for (smart) capabilities.

15.363.1 Detailed Description

L4::Capability class.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file smart_capability.

15.364 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 { return *this; }
00041
00042   void _delete() throw()
00043   {
00044     SMART::free(const_cast<Smart_cap<T,SMART>&>(*this));
00045   }
00046
00047   Cap<T> release() const throw()
00048   {
00049     l4_cap_idx_t r = cap();
00050     SMART::invalidate(const_cast<Smart_cap<T,SMART>&>(*this));
00051
00052     return Cap<T>(r);
00053   }
00054
00055   void reset()
00056   {
00057     _c = L4_INVALID_CAP;
```

```
  }

  Smart_cap() throw() : Cap_base(Invalid) {}

  Smart_cap(Cap_base::Cap_type t) throw() : Cap_base(t) {}

  template< typename O >
  Smart_cap(Cap<O> const &p) throw() : Cap_base(p.cap())
  { T* __t = ((O*)100); (void)__t; }

  template< typename O >
  Smart_cap(Cap<O> const &p, SMART const &smart) throw()
  : Cap_base(p.cap()), SMART(smart)
  { T* __t = ((O*)100); (void)__t; }

  template< typename O >
  Smart_cap(Smart_cap<O, SMART> const &o) throw()
  : Cap_base(SMART::copy(o)), SMART(o.smart())
  { T* __t = ((O*)100); (void)__t; }

  Smart_cap(Smart_cap const &o) throw()
  : Cap_base(SMART::copy(o)), SMART(o.smart())
  { }

  template< typename O >
  Smart_cap(typename Cap<O>::Cap_type cap) throw() : Cap_base(cap)
  { T* __t = ((O*)100); (void)__t; }

  void operator = (typename Cap<T>::Cap_type cap) throw()
  {
    _delete();
    _c = cap;
  }

  template< typename O >
  void operator = (Smart_cap<O, SMART> const &o) throw()
  {
    _delete();
    _c = this->SMART::copy(o).cap();
    this->SMART::operator = (o.smart());
    // return *this;
  }

  Smart_cap const &operator = (Smart_cap const &o) throw()
  {
    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) throw()
  : Cap_base(o.release()), SMART(o.smart())
  { T* __t = ((O*)100); (void)__t; }

  Smart_cap(Smart_cap &&o) throw()
  : Cap_base(o.release()), SMART(o.smart())
  { }

  template< typename O >
  void operator = (Smart_cap<O, SMART> &&o) throw()
  {
    _delete();
    _c = o.release().cap();
    this->SMART::operator = (o.smart());
    // return *this;
  }

  Smart_cap const &operator = (Smart_cap &&o) throw()
  {
    if (&o == this)
      return *this;

    _delete();
    _c = o.release().cap();
    this->SMART::operator = (o.smart());
    return *this;
  }
#endif

  Cap<T> operator -> () const throw() { return Cap<T>(_c); }
```

```
00155
00156   Cap<T> get() const throw() { return Cap<T>(_c); }
00157
00158   ~Smart_cap() throw() { _delete(); }
00159 };
00160
00161 template< typename T >
00162 class Weak_cap : public Cap_base
00163 {
00164 public:
00165   Weak_cap() : Cap_base(Invalid) {}
00166
00167   template< typename O >
00168   Weak_cap(typename Cap<O>::Cap_type t) : Cap_base(t)
00169   { T* __t = ((O*)100); (void)__t; }
00170
00171   template< typename O, typename S >
00172   Weak_cap(Smart_cap<O, S> const &c) : Cap_base(c.cap())
00173   { T* __t = ((O*)100); (void)__t; }
00174
00175   Weak_cap(Weak_cap const &o) : Cap_base(o) {}
00176
00177   template< typename O >
00178   Weak_cap(Weak_cap<O> const &o) : Cap_base(o)
00179   { T* __t = ((O*)100); (void)__t; }
00180
00181 };
00182
00183 namespace Cap_traits {
00184   template< typename T1, typename T2 >
00185   struct Type { enum { Equal = false }; };
00186
00187   template< typename T1 >
00188   struct Type<T1,T1> { enum { Equal = true }; };
00189 };
00190
00201 template< typename T, typename F, typename SMART >
00202 inline
00203 Smart_cap<T, SMART> cap_cast(Smart_cap<F, SMART> const &c)
      throw()
00204 {
00205   (void)static_cast<T const *>(reinterpret_cast<F const *>(100));
00206   return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00207 }
00208
00209
00220 template< typename T, typename F, typename SMART >
00221 inline
00222 Smart_cap<T, SMART> cap_reinterpret_cast(
      Smart_cap<F, SMART> const &c) throw()
00223 {
00224   return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00225 }
00226
00227
00228 }
00229
```

15.365 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.

Namespaces

- L4

  L4 low-level kernel interface.

15.365.1 Detailed Description

Common task related definitions.

Definition in file task.

15.366 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
00043 class Task :
00044   public Kobject_t<Task, Kobject, L4_PROTO_TASK,
00045                    Type_info::Demand_t<2> >
00046 {
00047 public:
00067   l4_msgtag_t map(Cap<Task> const &src_task,
00068                   l4_fpage_t const &snd_fpage, l4_addr_t snd_base,
00069                   l4_utcb_t *utcb = l4_utcb()) throw()
00070   { return l4_task_map_u(cap(), src_task.cap(), snd_fpage, snd_base, utcb); }
00071
00090   l4_msgtag_t unmap(l4_fpage_t const &fpage,
00091                     l4_umword_t map_mask,
00092                     l4_utcb_t *utcb = l4_utcb()) throw()
00093   { return l4_task_unmap_u(cap(), fpage, map_mask, utcb); }
00094
00115   l4_msgtag_t unmap_batch(l4_fpage_t const *fpages,
00116                           unsigned num_fpages,
00117                           l4_umword_t map_mask,
00118                           l4_utcb_t *utcb = l4_utcb()) throw()
00119   { return l4_task_unmap_batch_u(cap(), fpages, num_fpages, map_mask, utcb); }
00120
00133   l4_msgtag_t delete_obj(L4::Cap<void> obj,
00134                          l4_utcb_t *utcb = l4_utcb()) throw()
00135   { return l4_task_delete_obj_u(cap(), obj.cap(), utcb); }
00136
00147   l4_msgtag_t release_cap(L4::Cap<void> cap,
00148                           l4_utcb_t *utcb = l4_utcb()) throw()
00149   { return l4_task_release_cap_u(this->cap(), cap.cap(), utcb); }
00150
00163   l4_msgtag_t cap_valid(Cap<void> const &cap,
00164                         l4_utcb_t *utcb = l4_utcb()) throw()
00165   { return l4_task_cap_valid_u(this->cap(), cap.cap(), utcb); }
00166
00179   l4_msgtag_t cap_has_child(Cap<void> const &cap,
00180                             l4_utcb_t *utcb = l4_utcb()) throw()
00181     L4_DEPRECATED("Do not use. Future uncertain.")
00182   { return l4_task_cap_has_child_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()) throw()
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,
```

```
                                l4_utcb_t *utcb = l4_utcb()) throw()
  { return l4_task_add_ku_mem_u(cap(), fpage, utcb); }

};
}
```

15.367 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 }

  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_addr_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, unsigned long 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 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_has_child (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW

  Test whether a capability has child mappings (in another task).
- 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.

15.367.1 Detailed Description

Common task related definitions.

Definition in file task.h.

15.368 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
00060 L4_INLINE l4_msgtag_t
00061 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00062             l4_fpage_t snd_fpage, l4_addr_t snd_base)
      L4_NOTHROW;
```

```
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_addr_t snd_base,
l4_utcb_t *utcb) L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
              l4_umword_t map_mask) L4_NOTHROW;

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 *utcb)
L4_NOTHROW;

L4_INLINE l4_msgtag_t
l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
                    unsigned num_fpages, unsigned long 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, unsigned long 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_has_child(l4_cap_idx_t task,
l4_cap_idx_t cap) L4_NOTHROW
  L4_DEPRECATED("Do not use. Future uncertain.");

L4_INLINE l4_msgtag_t
l4_task_cap_has_child_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,
};


/* 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, unsigned long 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_FPAGE_RWX).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,
                unsigned long 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, unsigned long 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_has_child_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, unsigned long 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,
              unsigned long 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, unsigned long 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
l4_task_release_cap(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
{
  return l4_task_release_cap_u(task, cap, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
{
  return l4_task_cap_valid_u(task, cap, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_task_cap_has_child(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
{
  return l4_task_cap_has_child_u(task, cap, l4_utcb());
}

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
{
  return l4_task_cap_equal_u(task, cap_a, cap_b, l4_utcb());
}

L4_INLINE l4_msgtag_t
l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW
{
  return l4_task_add_ku_mem_u(task, ku_mem, l4_utcb());
```

```
00439 }
```

15.369 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.
- class L4::Thread::Attr

  Thread attributes used for control_commit().
- class L4::Thread::Modify_senders

  Wrapper class for modifying senders.

Namespaces

- L4

  L4 low-level kernel interface.

15.369.1 Detailed Description

Common thread related definitions.

Definition in file thread.

15.370 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:
00085   l4_msgtag_t ex_regs(l4_addr_t ip, l4_addr_t sp,
00086                       l4_umword_t flags,
00087                       l4_utcb_t *utcb = l4_utcb()) throw()
00088   { return l4_thread_ex_regs_u(cap(), ip, sp, flags, utcb); }
00089
00111   l4_msgtag_t ex_regs(l4_addr_t *ip, l4_addr_t *sp,
00112                       l4_umword_t *flags,
00113                       l4_utcb_t *utcb = l4_utcb()) throw()
00114   { return l4_thread_ex_regs_ret_u(cap(), ip, sp, flags, utcb); }
00115
00116
00125   class Attr
```

```
{
private:
   friend class L4::Thread;
   l4_utcb_t *_u;

 public:
   explicit Attr(l4_utcb_t *utcb = l4_utcb()) throw() : _u(utcb)
   { l4_thread_control_start_u(utcb); }

   void pager(Cap<void> const &pager) throw()
   { l4_thread_control_pager_u(pager.cap(), _u); }

   Cap<void> pager() throw()
   { return Cap<void>(l4_utcb_mr_u(_u)->mr[1]); }

   void exc_handler(Cap<void> const &exc_handler) throw()
   { l4_thread_control_exc_handler_u(exc_handler.cap(), _u); }

   Cap<void> exc_handler() throw()
   { return Cap<void>(l4_utcb_mr_u(_u)->mr[2]); }

   void bind(l4_utcb_t *thread_utcb, Cap<Task> const &task) throw()
   { l4_thread_control_bind_u(thread_utcb, task.cap(), _u); }

   void alien(int on) throw()
   { l4_thread_control_alien_u(_u, on); }

   void ux_host_syscall(int on) throw()
   { l4_thread_control_ux_host_syscall_u(_u, on); }

 };

l4_msgtag_t control(Attr const &attr) throw()
{ return l4_thread_control_commit_u(cap(), attr._u); }

l4_msgtag_t switch_to(l4_utcb_t *utcb = l4_utcb()) throw()
{ return l4_thread_switch_u(cap(), utcb); }

l4_msgtag_t stats_time(l4_kernel_clock_t *us,
                       l4_utcb_t *utcb = l4_utcb()) throw()
{ return l4_thread_stats_time_u(cap(), us, utcb); }

l4_msgtag_t vcpu_resume_start(l4_utcb_t *utcb = l4_utcb()) throw()
{ return l4_thread_vcpu_resume_start_u(utcb); }

l4_msgtag_t vcpu_resume_commit(l4_msgtag_t tag,
                               l4_utcb_t *utcb = l4_utcb()) throw()
{ return l4_thread_vcpu_resume_commit_u(cap(), tag, utcb); }

 l4_msgtag_t vcpu_control(l4_addr_t vcpu_state, l4_utcb_t *utcb = l4_utcb())
    throw()
 { return l4_thread_vcpu_control_u(cap(), vcpu_state, utcb); }

 l4_msgtag_t vcpu_control_ext(l4_addr_t ext_vcpu_state,
                              l4_utcb_t *utcb = l4_utcb()) throw()
 { return l4_thread_vcpu_control_ext_u(cap(), ext_vcpu_state, utcb); }

l4_msgtag_t register_del_irq(Cap<Irq> irq, l4_utcb_t *u = l4_utcb()) throw()
{ return l4_thread_register_del_irq_u(cap(), irq.cap(), u); }

class Modify_senders
{
private:
  friend class Thread;
  l4_utcb_t *utcb;
  unsigned cnt;

public:
  explicit Modify_senders(l4_utcb_t *u = l4_utcb()) throw()
  : utcb(u), cnt(1)
  {
    l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
  }

  int add(l4_umword_t match_mask, l4_umword_t match,
          l4_umword_t del_bits, l4_umword_t add_bits) throw()
  {
    l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
    if (cnt >= L4_UTCB_GENERIC_DATA_SIZE - 4)
      return -L4_ENOMEM;
    m->mr[cnt++] = match_mask;
    m->mr[cnt++] = match;
    m->mr[cnt++] = del_bits;
    m->mr[cnt++] = add_bits;
```

```
      return 0;
    }
  };

  l4_msgtag_t modify_senders(Modify_senders const &todo) throw()
  {
    return l4_ipc_call(cap(), todo.utcb, l4_msgtag(
L4_PROTO_THREAD, todo.cnt, 0, 0), L4_IPC_NEVER);
  }
};
}
```

15.371 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.

15.371.1 Detailed Description

Thread implementation.

Definition in file thread.

15.372 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");
```

```
    virtual void run() = 0;
    virtual void shutdown() asm ("L4_Thread_shutdown");
    void start();
    void stop();

    L4::Cap<L4::Thread> self() const throw()
    { return _cap; }

    State state() const
    { return _state; }

    static void start_cxx_thread(Thread *_this)
      asm ("L4_Thread_start_cxx_thread");

    static void kill_cxx_thread(Thread *_this)
      asm ("L4_Thread_kill_cxx_thread");

    static void set_pager(L4::Cap<void>const &p) throw()
    { _pager = p; }

  private:
    int create();

    L4::Cap<L4::Thread> _cap;
    State _state;

  protected:
    void *_stack;

  private:
    static L4::Cap<void> _pager;
    static L4::Cap<void> _master;
  };

};

#endif  /* CXX_THREAD_H__ */
```

15.373 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.

15.373.1 Detailed Description

Type information server template.

Definition in file typeinfo_svr.

15.374 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  */
00023
00024 #pragma once
00025
00026 #include <l4/sys/meta>
00027 #include <l4/sys/cxx/ipc_epiface>
00028
00029 namespace L4 { namespace Util {
00030
00031 template<typename KO, typename IOS>
00032 long handle_meta_request(IOS &ios)
00033 {
00034   using L4::Ipc::Msg::dispatch_call;
00035   typedef L4::Ipc::Detail::Meta_svr<KO> Msvr;
00036   l4_msgtag_t tag = dispatch_call<L4::Meta::Rpcs>((Msvr *)0, ios.utcb(),
00037                                                   ios.tag(), 0);
00038   ios.set_ipc_params(tag);
00039   return tag.label();
00040 }
00041
00042 }}
```

15.375 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.

15.375.1 Detailed Description

UTCB definitions.

Definition in file utcb.h.

15.376 utcb.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_UTCB_H
#define _L4_SYS_UTCB_H

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

typedef struct l4_utcb_t l4_utcb_t;

typedef union l4_msg_regs_t
{
  l4_umword_t mr[L4_UTCB_GENERIC_DATA_SIZE];
  l4_uint64_t mr64[L4_UTCB_GENERIC_DATA_SIZE / (sizeof(l4_uint64_t)/sizeof(l4_umword_t))];
} l4_msg_regs_t;

typedef struct l4_buf_regs_t
{
  l4_umword_t bdr;

  l4_umword_t br[L4_UTCB_GENERIC_BUFFERS_SIZE];
} l4_buf_regs_t;

typedef struct l4_thread_regs_t
{
  l4_umword_t  error;
  l4_timeout_t xfer;
  l4_umword_t  user[3];
} l4_thread_regs_t;

__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
{ return l4_utcb_mr_u(l4_utcb()); }

L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW
{ return l4_utcb_br_u(l4_utcb()); }

L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void) L4_NOTHROW
{ return l4_utcb_tcr_u(l4_utcb()); }

L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW
{ return l4_utcb_exc_u(l4_utcb()); }

L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW
{ l4_utcb_inherit_fpu_u(l4_utcb(), switch_on); }

L4_INLINE
l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t val, int pos,
                              l4_utcb_t *utcb) L4_NOTHROW
{
  union T
  {
    l4_kernel_clock_t t;
    l4_umword_t m[sizeof(l4_kernel_clock_t)/sizeof(
l4_umword_t)];
  };
  l4_timeout_s to;
  to.t = 0x8000 | pos;
  ((union T*)(l4_utcb_br_u(utcb)->br + pos))->t = val;
  return to;
}

L4_INLINE
l4_timeout_s l4_timeout_abs(l4_kernel_clock_t val, int pos)
L4_NOTHROW
{ return l4_timeout_abs_u(val, pos, l4_utcb()); }

L4_INLINE unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW
{ return idx / (sizeof(l4_uint64_t) / sizeof(l4_umword_t)); }

__END_DECLS
```

```
00391 #endif /* ! _L4_SYS_UTCB_H */
```

15.377 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().

15.377.1 Detailed Description

UTCB definitions for ARM.

Definition in file utcb.h.

15.378 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 #ifndef __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
00025 #define __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
00026
00027 #include <l4/sys/types.h>
00028
00038 typedef struct l4_exc_regs_t
00039 {
00040   l4_umword_t pfa;
00041   l4_umword_t err;
00043   l4_umword_t r[13];
00044   l4_umword_t sp;
00045   l4_umword_t ulr;
00046   l4_umword_t _dummy1;
00047   l4_umword_t pc;
00048   l4_umword_t cpsr;
00049   l4_umword_t tpidruro;
00050 } l4_exc_regs_t;
00051
00057 enum L4_utcb_consts_arm
00058 {
00059   L4_UTCB_EXCEPTION_REGS_SIZE    = sizeof(
      l4_exc_regs_t) / sizeof(l4_umword_t),
00060   L4_UTCB_GENERIC_DATA_SIZE      = 63,
00061   L4_UTCB_GENERIC_BUFFERS_SIZE   = 58,
00062
```

```
  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__ */
```

15.379 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().

15.379.1 Detailed Description

UTCB definitions for amd64.

Definition in file utcb.h.

15.380 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
00049   L4_UTCB_INHERIT_FPU            = 1UL << 24,
00050   L4_UTCB_OFFSET                 = 1024,
00051 };
00052
00057 typedef struct l4_exc_regs_t
00058 {
00059   l4_umword_t r15;
00060   l4_umword_t r14;
00061   l4_umword_t r13;
00062   l4_umword_t r12;
00063   l4_umword_t r11;
00064   l4_umword_t r10;
00065   l4_umword_t r9;
00066   l4_umword_t r8;
00067   l4_umword_t rdi;
00068   l4_umword_t rsi;
00069   l4_umword_t rbp;
00070   l4_umword_t pfa;
00071   l4_umword_t rbx;
00072   l4_umword_t rdx;
00073   l4_umword_t rcx;
00074   l4_umword_t rax;
00076   l4_umword_t trapno;
00077   l4_umword_t err;
00078   l4_umword_t ip;
00079   l4_umword_t dummy1;
00080   l4_umword_t flags;
00081   l4_umword_t sp;
00082   l4_umword_t ss;
00083   l4_umword_t fs_base;
00084   l4_umword_t gs_base;
00085   l4_uint16_t ds, es, fs, gs;
00086 } l4_exc_regs_t;
00087
```

```
#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__ */
```

15.381 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_GENERI↩
  C_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().

15.381.1 Detailed Description

UTCB definitions for X86.

Definition in file utcb.h.

15.382 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
00073 {
00074   l4_umword_t es;
00075   l4_umword_t ds;
00076   l4_umword_t gs;
00077   l4_umword_t fs;
00079   l4_umword_t edi;
00080   l4_umword_t esi;
00081   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__ */
```

15.383 l4/sys/vcon File Reference

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.

Namespaces

- L4

  L4 low-level kernel interface.

15.383.1 Detailed Description

Virtual console interface.

Definition in file vcon.

15.384 vcon

```
// 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/icu>
#include <l4/sys/vcon.h>
#include <l4/sys/capability>

namespace L4 {

class Vcon :
  public Kobject_t<Vcon, Icu, L4_PROTO_LOG>
{
public:
  l4_msgtag_t
  send(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const throw()
  { return l4_vcon_send_u(cap(), buf, size, utcb); }

  long
  write(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const throw()
  { return l4_vcon_write_u(cap(), buf, size, utcb); }

  int
  read(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const throw()
  { return l4_vcon_read_u(cap(), buf, size, utcb); }

  int
  read_with_flags(char *buf, unsigned size, l4_utcb_t *utcb =
l4_utcb()) const throw()
  { return l4_vcon_read_with_flags_u(cap(), buf, size, utcb); }

  l4_msgtag_t
  set_attr(l4_vcon_attr_t const *attr, l4_utcb_t *utcb =
l4_utcb()) const throw()
  { return l4_vcon_set_attr_u(cap(), attr, utcb); }

  l4_msgtag_t
  get_attr(l4_vcon_attr_t *attr, l4_utcb_t *utcb =
l4_utcb()) const throw()
  { return l4_vcon_get_attr_u(cap(), attr, utcb); }

  typedef L4::Typeid::Raw_ipc<Vcon> Rpcs;
};

}
```

15.385 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.

15.385.1 Detailed Description

Virtual console interface.

Definition in file vcon.h.

15.385.2 Enumeration Type Documentation

15.385.2.1 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 167 of file vcon.h.

15.386 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
00056 L4_INLINE l4_msgtag_t
00057 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size)
      L4_NOTHROW;
00058
00065 L4_INLINE l4_msgtag_t
00066 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size,
      l4_utcb_t *utcb) L4_NOTHROW;
00067
00079 L4_INLINE long
00080 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size)
      L4_NOTHROW;
00081
00088 L4_INLINE long
00089 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size,
      l4_utcb_t *utcb) L4_NOTHROW;
00090
00095 enum L4_vcon_size_consts
00096 {
00098   L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(
      l4_umword_t),
00100   L4_VCON_READ_SIZE  = (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(
      l4_umword_t),
00101 };
```

```
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;
} 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;


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 + (size + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t),
                               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());
}
```

15.387 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.

15.387.1 Detailed Description

Descriptors for virtual hardware (under UX).

Definition in file vhw.h.

15.388 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;
```

```
00111
00112   return NULL;
00113 }
00114
00115 #endif /* ! _L4_SYS_VHW_H */
```

15.389 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.

Namespaces

- L4

  L4 low-level kernel interface.

15.389.1 Detailed Description

Virtualization interface.

Definition in file vm.

15.390 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
00035 class Vm : public Kobject_t<Vm, Task, L4_PROTO_VM>
00036 {
00037 protected:
00038   Vm();
00039
00040 private:
00041   Vm(Vm const &);
00042   void operator = (Vm const &);
00043 };
00044
00045 };
```

15.391 l4/util/alloc.h File Reference

Allocator using a bit-array.

```
#include <l4/sys/l4int.h>
#include <l4/util/bitops.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for alloc.h:

15.391.1 Detailed Description

Allocator using a bit-array.

Date

09/14/2004

Author

Jork Loeser jork.loeser@inf.tu-dresden.de

Definition in file alloc.h.

15.392 alloc.h

```
00001
00009 /*
00010  * (c) 2004-2009 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 #ifndef __UTIL_INCLUDE_ALLOC_H_
00017 #define __UTIL_INCLUDE_ALLOC_H_
00018 #include <l4/sys/l4int.h>
00019 #include <l4/util/bitops.h>
00020 #include <l4/sys/compiler.h>
00021
00022 EXTERN_C_BEGIN
```

```
00023
00024 typedef struct {
00025     int base, count, next_elem;
00026     l4_umword_t *bits;
00027 } l4util_alloc_t;
00028
00029 #define L4UTIL_ALLOC_BITS_SIZE (8 * sizeof(l4_umword_t))
00030
00031 L4_CV l4util_alloc_t *l4util_alloc_init(int count, int base);
00032 L4_CV int l4util_alloc_avail(l4util_alloc_t *alloc, int elem);
00033 L4_CV int l4util_alloc_occupy(l4util_alloc_t *alloc, int elem);
00034 L4_CV int l4util_alloc_alloc(l4util_alloc_t *alloc);
00035 L4_CV int l4util_alloc_free(l4util_alloc_t *alloc, int elem);
00036
00037 EXTERN_C_END
00038 #endif
```

15.393 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.

15.393.1 Detailed Description

Alloc list.

Definition in file alloc.h.

15.394 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
00024 #ifndef L4_CXX_ALLOC_H__
```

```
#define L4_CXX_ALLOC_H__

namespace L4 {

  class Alloc_list
  {
  public:
    Alloc_list() : _free(0) {}
    Alloc_list( void *blk, unsigned long size ) : _free(0)
    { free( blk, size ); }

    void free( void *blk, unsigned long size );
    void *alloc( unsigned long size );

  private:
    struct Elem
    {
      Elem *next;
      unsigned long size;
    };

    Elem *_free;
  };
};

#endif // L4_CXX_ALLOC_H__
```

15.395 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:

15.395.1 Detailed Description

Some useful assert-style macros.

Date

09/2009

Author

Bjoern Doebel doebel@tudos.org

Definition in file assert.h.

15.396 assert.h

```
/*****************************************************************************/
/*
 * (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.
 */

/*****************************************************************************/
#pragma once

#ifdef NDEBUG

#define DO_NOTHING               do {} while (0)
#define ASSERT_VALID(c)          DO_NOTHING
#define ASSERT_EQUAL(a,b)        DO_NOTHING
#define ASSERT_NOT_EQUAL(a,b)    DO_NOTHING
#define ASSERT_LOWER_EQ(a,b)     DO_NOTHING
#define ASSERT_GREATER_EQ(a,b)   DO_NOTHING
#define ASSERT_BETWEEN(a,b,c)    DO_NOTHING
#define ASSERT_IPC_OK(i)         DO_NOTHING
#define ASSERT_OK(e)             do { (void)e; } while (0)
#define ASSERT_NOT_NULL(p)       DO_NOTHING
#ifndef assert
#define assert(cond)             DO_NOTHING
#endif

#else // NDEBUG

#ifndef ASSERT_PRINTF
#include <stdio.h>
#define ASSERT_PRINTF printf
#endif
#ifndef ASSERT_ASSERT
#include <assert.h>
#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
```

15.397 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.

15.397.1 Detailed Description

Low-level assert implementation.

Definition in file assert.h.

15.397.2 Macro Definition Documentation

15.397.2.1 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.

Referenced by L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free(), L4Re::Util::Counting_cap_alloc< C↩OUNTERTYPE >::release(), and L4Re::Util::Br_manager::set_rcv_cap_flags().

15.398 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>
00033
00043 #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 */
```

15.399 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)

15.399.1 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.

15.400 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__ */
```

15.401 arm/l4/sys/atomic.h File Reference

Atomic memory modifications.

`#include <l4/sys/compiler.h>`

Include dependency graph for atomic.h:

15.401.1 Detailed Description

Atomic memory modifications.

Definition in file atomic.h.

15.402 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;
```

15.403 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.

15.403.1 Detailed Description

Atomic template.

Definition in file atomic.h.

15.404 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
00024 #ifndef L4_CXX_ATOMIC_H__
00025 #define L4_CXX_ATOMIC_H__
00026
00027 #include <l4/util/atomic.h>
00028
00029 extern "C" void  ____error_compare_and_swap_does_not_support_3_bytes____();
00030 extern "C" void  ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00031
00032 namespace L4
00033 {
00034   template< typename X >
00035   inline int compare_and_swap(X volatile *dst, X old_val, X new_val)
00036   {
00037     switch (sizeof(X))
00038       {
00039       case 1:
00040   return l4util_cmpxchg8((l4_uint8_t volatile*)dst, old_val, new_val);
00041       case 2:
00042   return l4util_cmpxchg16((l4_uint16_t volatile *)dst, old_val, new_val);
00043       case 3: ____error_compare_and_swap_does_not_support_3_bytes____();
00044       case 4:
00045   return l4util_cmpxchg32((l4_uint32_t volatile*)dst, old_val, new_val);
00046       default:
00047         ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00048       }
00049     return 0;
00050   }
00051 };
00052
00053 #endif // L4_CXX_ATOMIC_H__
```

15.405 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.

15.405.1 Detailed Description

Backtrace.

Definition in file backtrace.h.

15.405.2 Function Documentation

15.405.2.1 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

15.406 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
```

15.407 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

15.407.1 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.

15.408 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
00030
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
```

15.409 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 (const unsigned long val)

  Find the next power of 2 for a given number.

15.409.1 Detailed Description

bit manipulation functions

Date

07/03/2001

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file bitops.h.

15.410 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);
```

```
L4_INLINE int
l4util_find_first_zero_bit(const void * dest,
l4_size_t size);


L4_INLINE int
l4util_next_power2(const 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__ */
```

15.411 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 - figure 1-9, page 1-9.
- 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_Sym

  ELF32 symbol table entry.
- struct Elf64_Sym

  ELF64 symbol table entry.

Macros

- #define EI_NIDENT 16

  number of characters
- #define EI_CLASS 4

  ELF class byte index.
- #define ELFCLASSNONE 0

  Invalid ELF class.
- #define ELFCLASS32 1

  32-bit objects
- #define ELFCLASS64 2

  64-bit objects
- #define ELFCLASSNUM 3

  Mask for 32-bit or 64-bit class.
- #define EI_DATA 5

  Data encoding byte index.
- #define ELFDATANONE 0

  Invalid data encoding.
- #define ELFDATA2LSB 1

  2's complement, little endian
- #define ELFDATA2MSB 2

  2's complement, big endian
- #define EI_VERSION 6

  File version byte index.
- #define EI_OSABI 7

  OS ABI identification.
- #define ELFOSABI_NONE 0

UNIX System V ABI.

- #define ELFOSABI_SYSV 0

  Alias.
- #define ELFOSABI_HPUX 1

  HP-UX.
- #define ELFOSABI_NETBSD 2

  NetBSD.
- #define ELFOSABI_LINUX 3

  Linux.
- #define ELFOSABI_SOLARIS 6

  Sun Solaris.
- #define ELFOSABI_AIX 7

  IBM AIX.
- #define ELFOSABI_IRIX 8

  SGI Irix.
- #define ELFOSABI_FREEBSD 9

  FreeBSD.
- #define ELFOSABI_TRU64 10

  Compaq TRU64 UNIX.
- #define ELFOSABI_MODESTO 11

  Novell Modesto.
- #define ELFOSABI_OPENBSD 12

  OpenBSD.
- #define ELFOSABI_ARM 97

  ARM.
- #define ELFOSABI_STANDALONE 255

  Standalone (embedded) application.
- #define EI_ABIVERSION 8

  ABI version.
- #define EI_PAD 9

  Byte index of padding bytes.
- #define ET_NONE 0

  no file type
- #define ET_REL 1

  relocatable file
- #define ET_EXEC 2

  executable file
- #define ET_DYN 3

  shared object file
- #define ET_CORE 4

  core file
- #define ET_LOPROC 0xff00

  processor-specific
- #define ET_HIPROC 0xffff

  processor-specific
- #define EM_NONE 0

  no machine
- #define EM_M32 1

  AT&T WE 32100.
- #define EM_SPARC 2

  SPARC.

- #define EM_386 3

  Intel 80386.
- #define EM_68K 4

  Motorola 68000.
- #define EM_88K 5

  Motorola 88000.
- #define EM_860 7

  Intel 80860.
- #define EM_MIPS 8

  MIPS RS3000 big-endian.
- #define EM_MIPS_RS4_BE 10

  MIPS RS4000 big-endian.
- #define EM_SPARC64 11

  SPARC 64-bit.
- #define EM_PARISC 15

  HP PA-RISC.
- #define EM_VPP500 17

  Fujitsu VPP500.
- #define EM_SPARC32PLUS 18

  Sun's V8plus.
- #define EM_960 19

  Intel 80960.
- #define EM_PPC 20

  PowerPC.
- #define EM_V800 36

  NEC V800.
- #define EM_FR20 37

  Fujitsu FR20.
- #define EM_RH32 38

  TRW RH-32.
- #define EM_RCE 39

  Motorola RCE.
- #define EM_ARM 40

  Advanced RISC Machines ARM.
- #define EM_ALPHA 41

  Digital Alpha.
- #define EM_SH 42

  Hitachi SuperH.
- #define EM_SPARCV9 43

  SPARC v9 64-bit.
- #define EM_TRICORE 44

  Siemens Tricore embedded processor.
- #define EM_ARC 45

  Argonaut RISC Core, Argonaut Techn Inc.
- #define EM_H8_300 46

  Hitachi H8/300.
- #define EM_H8_300H 47

  Hitachi H8/300H.
- #define EM_H8S 48

  Hitachi H8/S.
- #define EM_H8_500 49

Hitachi H8/500.

- #define EM_IA_64 50

  HP/Intel IA-64.
- #define EM_MIPS_X 51

  Stanford MIPS-X.
- #define EM_COLDFIRE 52

  Motorola Coldfire.
- #define EM_68HC12 53

  Motorola M68HC12.
- #define EM_X86_64 62

  Advanced Micro Devices x86-64.
- #define EM_PDSP 63

  Sony DSP Processor.
- #define EM_FX66 66

  Siemens FX66 microcontroller.
- #define EM_ST9PLUS 67

  STMicroelectronics ST9+ 8/16 mc.
- #define EM_ST7 68

  STmicroelectronics ST7 8 bit mc.
- #define EM_68HC16 69

  Motorola MC68HC16 microcontroller.
- #define EM_68HC11 70

  Motorola MC68HC11 microcontroller.
- #define EM_68HC08 71

  Motorola MC68HC08 microcontroller.
- #define EM_68HC05 72

  Motorola MC68HC05 microcontroller.
- #define EM_SVX 73

  Silicon Graphics SVx.
- #define EM_ST19 74

  STMicroelectronics ST19 8 bit mc.
- #define EM_VAX 75

  Digital VAX.
- #define EM_CRIS 76

  Axis Communications 32-bit embedded processor.
- #define EM_JAVELIN 77

  Infineon Technologies 32-bit embedded processor.
- #define EM_FIREPATH 78

  Element 14 64-bit DSP Processor.
- #define EM_ZSP 79

  LSI Logic 16-bit DSP Processor.
- #define EM_MMIX 80

  Donald Knuth's educational 64-bit processor.
- #define EM_HUANY 81

  Harvard University machine-independent object files.
- #define EM_PRISM 82

  SiTera Prism.
- #define EM_AVR 83

  Atmel AVR 8-bit microcontroller.
- #define EM_FR30 84

  Fujitsu FR30.

- #define EM_D10V 85

  Mitsubishi D10V.
- #define EM_D30V 86

  Mitsubishi D30V.
- #define EM_V850 87

  NEC v850.
- #define EM_M32R 88

  Mitsubishi M32R.
- #define EM_MN10300 89

  Matsushita MN10300.
- #define EM_MN10200 90

  Matsushita MN10200.
- #define EM_PJ 91

  picoJava
- #define EM_OPENRISC 92

  OpenRISC 32-bit embedded processor.
- #define EM_ARC_A5 93

  ARC Cores Tangent-A5.
- #define EM_XTENSA 94

  Tensilica Xtensa Architecture.
- #define EM_ALTERA_NIOS2 113

  Altera Nios II.
- #define EM_AARCH64 183

  ARM AARCH64.
- #define EM_TILEPRO 188

  Tilera TILEPro.
- #define EM_MICROBLAZE 189

  Xilinx MicroBlaze.
- #define EM_TILEGX 191

  Tilera TILE-Gx.
- #define EV_NONE 0

  Invalid version.
- #define EV_CURRENT 1

  Current version.
- #define EI_MAG0 0

  file id
- #define EI_MAG1 1

  file id
- #define EI_MAG2 2

  file id
- #define EI_MAG3 3

  file id
- #define EI_CLASS 4

  ELF class byte index.
- #define EI_DATA 5

  Data encoding byte index.
- #define EI_VERSION 6

  File version byte index.
- #define EI_OSABI 7

  OS ABI identification.
- #define EI_ABIVERSION 8

ABI version.

- #define EI_PAD 9

  Byte index of padding bytes.
- #define ELFMAG0 0x7f

  e_ident[EI_MAG0]
- #define ELFMAG1 'E'

  e_ident[EI_MAG1]
- #define ELFMAG2 'L'

  e_ident[EI_MAG2]
- #define ELFMAG3 'F'

  e_ident[EI_MAG3]
- #define ELFCLASSNONE 0

  Invalid ELF class.
- #define ELFCLASSS32 1

  32-bit object
- #define ELFCLASSS64 2

  64-bit object
- #define ELFDATANONE 0

  Invalid data encoding.
- #define ELFDATA2LSB 1

  2's complement, little endian
- #define ELFDATA2MSB 2

  2's complement, big endian
- #define ELFOSABI_SYSV 0

  Alias.
- #define ELFOSABI_HPUX 1

  HP-UX.
- #define ELFOSABI_STANDALONE 255

  Standalone (embedded) application.
- #define SHN_UNDEF 0

  undefined section header entry
- #define SHN_LORESERVE 0xff00

  lower bound of reserved indexes
- #define SHN_LOPROC 0xff00

  lower bound of proc spec entr
- #define SHN_HIPROC 0xff1f

  upper bound of proc spec entr
- #define SHN_ABS 0xfff1

  absolute values for ref
- #define SHN_COMMON 0xfff2

  common symbols
- #define SHN_HIRESERVE 0xffff

  upper bound of reserved indexes
- #define SHT_INIT_ARRAY 14

  Array of constructors.
- #define SHT_FINI_ARRAY 15

  Array of destructors.
- #define SHT_PREINIT_ARRAY 16

  Array of pre-constructors.
- #define SHT_GROUP 17

  Section group.

- #define SHT_SYMTAB_SHNDX 18

  Extended section indeces.
- #define SHT_NUM 19

  Number of defined types.
- #define SHF_WRITE 0x1

  writeable during execution
- #define SHF_ALLOC 0x2

  section occupies virt memory
- #define SHF_EXECINSTR 0x4

  code section
- #define SHF_MERGE 0x10

  Might be merged.
- #define SHF_STRINGS 0x20

  Contains nul-terminated strings.
- #define SHF_INFO_LINK 0x40

  'sh_info' contains SHT index
- #define SHF_LINK_ORDER 0x80

  Preserve order after combining.
- #define SHF_OS_NONCONFORMING 0x100

  Non-standard OS specific handling required.
- #define SHF_GROUP 0x200

  Section is member of a group.
- #define SHF_TLS 0x400

  Section hold thread-local data.
- #define SHF_MASKOS 0x0ff00000

  OS-specific.
- #define SHF_MASKPROC 0xf0000000

  proc spec mask
- #define PT_NULL 0

  array is unused
- #define PT_LOAD 1

  loadable
- #define PT_DYNAMIC 2

  dynamic linking information
- #define PT_INTERP 3

  path to interpreter
- #define PT_NOTE 4

  auxiliary information
- #define PT_SHLIB 5

  reserved
- #define PT_PHDR 6

  location of the pht itself
- #define PT_TLS 7

  Thread-local storage segment.
- #define PT_NUM 8

  Number of defined types.
- #define PT_LOOS 0x60000000

  os spec.
- #define PT_HIOS 0x6fffffff

  os spec.
- #define PT_LOPROC 0x70000000

processor spec.

- #define PT_HIPROC 0x7fffffff

  processor spec.
- #define PT_GNU_EH_FRAME (PT_LOOS + 0x474e550)

  EH frame information.
- #define PT_GNU_STACK (PT_LOOS + 0x474e551)

  Flags for stack.
- #define PT_GNU_RELRO (PT_LOOS + 0x474e552)

  Read only after reloc.
- #define PT_L4_STACK (PT_LOOS + 0x12)

  Address of the stack.
- #define PT_L4_KIP (PT_LOOS + 0x13)

  Address of the KIP.
- #define PT_L4_AUX (PT_LOOS + 0x14)

  Address of the AUX strcutures.
- #define NT_PRSTATUS 1

  Contains copy of prstatus struct.
- #define NT_FPREGSET 2

  Contains copy of fpregset struct.
- #define NT_PRPSINFO 3

  Contains copy of prpsinfo struct.
- #define NT_PRXREG 4

  Contains copy of prxregset struct.
- #define NT_TASKSTRUCT 4

  Contains copy of task structure.
- #define NT_PLATFORM 5

  String from sysinfo(SI_PLATFORM)
- #define NT_AUXV 6

  Contains copy of auxv array.
- #define NT_GWINDOWS 7

  Contains copy of gwindows struct.
- #define NT_ASRS 8

  Contains copy of asrset struct.
- #define NT_PSTATUS 10

  Contains copy of pstatus struct.
- #define NT_PSINFO 13

  Contains copy of psinfo struct.
- #define NT_PRCRED 14

  Contains copy of prcred struct.
- #define NT_UTSNAME 15

  Contains copy of utsname struct.
- #define NT_LWPSTATUS 16

  Contains copy of lwpstatus struct.
- #define NT_LWPSINFO 17

  Contains copy of lwpinfo struct.
- #define NT_PRFPXREG 20

  Contains copy of fprxregset struct.
- #define NT_VERSION 1

  Contains a version string.
- #define DT_NULL 0

  Dynamic Array Tags, d_tag - figure 2-10, page 2-12.

- #define DT_NEEDED 1

  name of a needed library
- #define DT_PLTRELSZ 2

  total size of relocation entry
- #define DT_PLTGOT 3

  address assoc with prog link table
- #define DT_HASH 4

  address of symbol hash table
- #define DT_STRTAB 5

  address of string table
- #define DT_SYMTAB 6

  address of symbol table
- #define DT_RELA 7

  address of relocation table
- #define DT_RELASZ 8

  total size of relocation table
- #define DT_RELAENT 9

  size of DT_RELA relocation entry
- #define DT_STRSZ 10

  size of the string table
- #define DT_SYMENT 11

  size of a symbol table entry
- #define DT_INIT 12

  address of initialization function
- #define DT_FINI 13

  address of termination function
- #define DT_SONAME 14

  name of the shared object
- #define DT_RPATH 15

  search library path
- #define DT_SYMBOLIC 16

  alter symbol resolution algorithm
- #define DT_REL 17

  address of relocation table
- #define DT_RELSZ 18

  total size of DT_REL relocation table
- #define DT_RELENT 19

  size of the DT_REL relocation entry
- #define DT_PTRREL 20

  type of relocation entry
- #define DT_DEBUG 21

  for debugging purposes
- #define DT_TEXTREL 22

  at least on entry changes r/o section
- #define DT_JMPREL 23

  address of relocation entries
- #define DT_BIND_NOW 24

  Process relocations of object.
- #define DT_INIT_ARRAY 25

  Array with addresses of init fct.
- #define DT_FINI_ARRAY 26

Array with addresses of fini fct.

- #define DT_INIT_ARRAYSZ 27

  Size in bytes of DT_INIT_ARRAY.
- #define DT_FINI_ARRAYSZ 28

  Size in bytes of DT_FINI_ARRAY.
- #define DT_RUNPATH 29

  Library search path.
- #define DT_FLAGS 30

  Flags for the object being loaded.
- #define DT_ENCODING 32

  Start of encoded range.
- #define DT_PREINIT_ARRAY 32

  Array with addresses of preinit fct.
- #define DT_PREINIT_ARRAYSZ 33

  size in bytes of DT_PREINIT_ARRAY
- #define DT_NUM 34

  Number used.
- #define DT_LOOS 0x6000000d

  Start of OS-specific.
- #define DT_HIOS 0x6ffff000

  End of OS-specific.
- #define DT_LOPROC 0x70000000

  processor spec.
- #define DT_HIPROC 0x7fffffff

  processor spec.
- #define DF_ORIGIN 0x00000001

  Object may use DF_ORIGIN.
- #define DF_SYMBOLIC 0x00000002

  Symbol resolutions starts here.
- #define DF_TEXTREL 0x00000004

  Object contains text relocations.
- #define DF_BIND_NOW 0x00000008

  No lazy binding for this object.
- #define DF_STATIC_TLS 0x00000010

  Module uses the static TLS model.
- #define DF_1_NOW 0x00000001

  Set RTLD_NOW for this object.
- #define DF_1_GLOBAL 0x00000002

  Set RTLD_GLOBAL for this object.
- #define DF_1_GROUP 0x00000004

  Set RTLD_GROUP for this object.
- #define DF_1_NODELETE 0x00000008

  Set RTLD_NODELETE for this object.
- #define DF_1_LOADFLTR 0x00000010

  Trigger filtee loading at runtime.
- #define DF_1_INITFIRST 0x00000020

  Set RTLD_INITFIRST for this object.
- #define DF_1_NOOPEN 0x00000040

  Set RTLD_NOOPEN for this object.
- #define DF_1_ORIGIN 0x00000080

  $ORIGIN must be handled.

- #define DF_1_DIRECT 0x00000100

  Direct binding enabled.
- #define DF_1_INTERPOSE 0x00000400

  Object is used to interpose.
- #define DF_1_NODEFLIB 0x00000800

  Ignore default lib search path.
- #define DF_1_NODUMP 0x00001000

  Object can't be dldump'ed.
- #define DF_1_CONFALT 0x00002000

  Configuration alternative created.
- #define DF_1_ENDFILTEE 0x00004000

  Filtee terminates filters search.
- #define DF_1_DISPRELDNE 0x00008000

  Disp reloc applied at build time.
- #define DF_1_DISPRELPND 0x00010000

  Disp reloc applied at run-time.
- #define DF_P1_LAZYLOAD 0x00000001

  Lazyload following object.
- #define DF_P1_GROUPPERM 0x00000002

  Symbols from next object are not generally available.
- #define R_386_NONE 0

  none
- #define R_386_32 1

  S + A.
- #define R_386_PC32 2

  S + A - P.
- #define R_386_GOT32 3

  G + A - P.
- #define R_386_PLT32 4

  L + A - P.
- #define R_386_COPY 5

  none
- #define R_386_GLOB_DAT 6

  S.
- #define R_386_JMP_SLOT 7

  S.
- #define R_386_RELATIVE 8

  B + A.
- #define R_386_GOTOFF 9

  S + A - GOT.
- #define R_386_GOTPC 10

  GOT + A - P.
- #define STB_LOCAL 0

  not visible outside object file
- #define STB_GLOBAL 1

  visible to all objects beeing combined
- #define STB_WEAK 2

  resemble global symbols
- #define STB_LOOS 10

  os specific
- #define STB_HIOS 12

os specific

- #define STB_LOPROC 13

  proc specific

- #define STB_HIPROC 15

  proc specific

- #define STT_NOTYPE 0

  symbol's type not specified

- #define STT_OBJECT 1

  associated with a data object

- #define STT_FUNC 2

  associated with a function or other code

- #define STT_SECTION 3

  associated with a section

- #define STT_FILE 4

  source file name associated with object

- #define STT_LOOS 10

  os specific

- #define STT_HIOS 12

  os specific

- #define STT_LOPROC 13

  proc specific

- #define STT_HIPROC 15

  proc specific

Typedefs

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

15.411.1 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.

15.412 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 #ifndef _L4_EXEC_ELF_H
00033 #define _L4_EXEC_ELF_H
00034
00035 #include <l4/sys/l4int.h>
00036
00047 typedef l4_uint32_t  Elf32_Addr;
00048 typedef l4_uint32_t  Elf32_Off;
00049 typedef l4_uint16_t  Elf32_Half;
00050 typedef l4_uint32_t  Elf32_Word;
00051 typedef l4_int32_t  Elf32_Sword;
00052 typedef l4_uint64_t  Elf64_Addr;
00053 typedef l4_uint64_t  Elf64_Off;
00054 typedef l4_uint16_t  Elf64_Half;
00055 typedef l4_uint32_t  Elf64_Word;
00056 typedef l4_int32_t  Elf64_Sword;
00057 typedef l4_uint64_t  Elf64_Xword;
00058 typedef l4_int64_t  Elf64_Sxword;
00060
00061 #if L4_MWORD_BITS == 64
00062
00066 #define ElfW(type)      _ElfW(Elf, 64, type)
00067 #else
00068 #define ElfW(type)      _ElfW(Elf, 32, type)
00069 #endif
00070 #define _ElfW(e,w,t)    __ElfW(e, w, _##t)
00071 #define __ElfW(e,w,t)   e##w##t
00072
00073 #if defined(ARCH_x86)
00074 #define L4_ARCH_EI_DATA      ELFDATA2LSB
00075 #define L4_ARCH_E_MACHINE    EM_386
00076 #define L4_ARCH_EI_CLASS     ELFCLASS32
00077 #elif defined(ARCH_amd64)
00078 #define L4_ARCH_EI_DATA      ELFDATA2LSB
00079 #define L4_ARCH_E_MACHINE    EM_X86_64
00080 #define L4_ARCH_EI_CLASS     ELFCLASS64
00081 #elif defined(ARCH_arm)
00082 #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


/*************************************/
/* ELF Header - figure 1-3, page 1-3 */
/*************************************/

#define 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;

#define EI_CLASS  4
#define ELFCLASSNONE  0
#define ELFCLASS32  1
#define ELFCLASS64  2
#define ELFCLASSNUM 3
#define EI_DATA   5
#define ELFDATANONE 0
#define ELFDATA2LSB 1
#define ELFDATA2MSB 2
#define ELFDATANUM  3

#define EI_VERSION  6
#define EI_OSABI  7
#define ELFOSABI_NONE   0
#define ELFOSABI_SYSV   0
#define ELFOSABI_HPUX   1
#define ELFOSABI_NETBSD   2
#define ELFOSABI_LINUX    3
#define ELFOSABI_SOLARIS  6
#define ELFOSABI_AIX    7
```

```
#define ELFOSABI_IRIX   8
#define ELFOSABI_FREEBSD  9
#define ELFOSABI_TRU64    10
#define ELFOSABI_MODESTO  11
#define ELFOSABI_OPENBSD  12
#define ELFOSABI_ARM    97
#define ELFOSABI_STANDALONE 255
#define EI_ABIVERSION 8
#define EI_PAD    9
/* object file type - page 1-3 (e_type) */

#define ET_NONE   0
#define ET_REL    1
#define ET_EXEC   2
#define ET_DYN    3
#define ET_CORE   4
#define ET_LOPROC 0xff00
#define ET_HIPROC 0xffff
/* required architecture - page 1-4 (e_machine) */

#define EM_NONE   0
#define EM_M32    1
#define EM_SPARC  2
#define EM_386    3
#define EM_68K    4
#define EM_88K    5
#define EM_860    7
#define EM_MIPS   8
#define EM_MIPS_RS4_BE  10
#define EM_SPARC64  11
#define EM_PARISC 15
#define EM_VPP500 17
#define EM_SPARC32PLUS  18
#define EM_960    19
#define EM_PPC    20
#define EM_V800   36
#define EM_FR20   37
#define EM_RH32   38
#define EM_RCE    39
#define EM_ARM    40
#define EM_ALPHA  41
#define EM_SH   42
#define EM_SPARCV9  43
#define EM_TRICORE  44
#define EM_ARC    45
#define EM_H8_300 46
#define EM_H8_300H  47
#define EM_H8S    48
#define EM_H8_500 49
#define EM_IA_64  50
#define EM_MIPS_X 51
#define EM_COLDFIRE 52
#define EM_68HC12 53
#define EM_X86_64 62
#define EM_PDSP   63
#define EM_FX66   66
#define EM_ST9PLUS  67
#define EM_ST7    68
#define EM_68HC16 69
#define EM_68HC11 70
#define EM_68HC08 71
#define EM_68HC05 72
#define EM_SVX    73
#define EM_ST19   74
#define EM_VAX    75
#define EM_CRIS   76
#define EM_JAVELIN  77
#define EM_FIREPATH 78
#define EM_ZSP    79
#define EM_MMIX   80
#define EM_HUANY  81
#define EM_PRISM  82
#define EM_AVR    83
#define EM_FR30   84
#define EM_D10V   85
#define EM_D30V   86
#define EM_V850   87
#define EM_M32R   88
#define EM_MN10300  89
#define EM_MN10200  90
#define EM_PJ   91
#define EM_OPENRISC 92
#define EM_ARC_A5 93
#define EM_XTENSA 94
#define EM_ALTERA_NIOS2 113
#define EM_AARCH64  183
#define EM_TILEPRO  188
```

```
#define EM_MICROBLAZE 189
#define EM_TILEGX 191
#define 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

/* object file version - page 1-4 (e_version) */

#define EV_NONE   0
#define EV_CURRENT  1
/* e_ident[] Identification Indexes - figure 1-4, page 1-5 */

#define EI_MAG0   0
#define EI_MAG1   1
#define EI_MAG2   2
#define EI_MAG3   3
#define EI_CLASS  4
#define EI_DATA   5
#define EI_VERSION  6
#define EI_OSABI  7
#define EI_ABIVERSION 8
#define EI_PAD    9
/* magic number - page 1-5 */

#define ELFMAG0   0x7f
#define ELFMAG1   'E'
#define ELFMAG2   'L'
#define ELFMAG3   'F'
/* file class or capacity - page 1-6 */

#define ELFCLASSNONE  0
#define ELFCLASSS32 1
#define ELFCLASSS64 2
/* data encoding - page 1-6 */

#define ELFDATANONE 0
#define ELFDATA2LSB 1
#define ELFDATA2MSB 2
/* Identify operating system and ABI to which the object is targeted */

#define ELFOSABI_SYSV 0
#define ELFOSABI_HPUX 1
#define ELFOSABI_STANDALONE 255
/************************/
/* Sections - page 1-8 */
/************************/

/* special section indexes */

#define SHN_UNDEF 0
#define SHN_LORESERVE 0xff00
#define SHN_LOPROC  0xff00
#define SHN_HIPROC  0xff1f
#define SHN_ABS   0xfff1
#define SHN_COMMON  0xfff2
#define 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;

/* section type - figure 1-10, page 1-10 */
```

```
#define SHT_NULL  0
#define SHT_PROGBITS  1
#define SHT_SYMTAB  2
#define SHT_STRTAB  3
#define SHT_RELA  4
#define SHT_HASH  5
#define SHT_DYNAMIC 6
#define SHT_NOTE  7
#define SHT_NOBITS  8
#define SHT_REL   9
#define SHT_SHLIB 10
#define SHT_DYNSYM  11
#define SHT_INIT_ARRAY    14
#define SHT_FINI_ARRAY    15
#define SHT_PREINIT_ARRAY 16
#define SHT_GROUP   17
#define SHT_SYMTAB_SHNDX  18
#define SHT_NUM     19
#define SHT_LOOS  0x60000000
#define SHT_HIOS  0x6fffffff
#define SHT_LOPROC  0x70000000
#define SHT_HIPROC  0x7fffffff
#define SHT_LOUSER  0x80000000
#define SHT_HIUSER  0xffffffff

/* section attribute flags - page 1-12, figure 1-12 */

#define SHF_WRITE 0x1
#define SHF_ALLOC 0x2
#define SHF_EXECINSTR 0x4
#define SHF_MERGE 0x10
#define SHF_STRINGS 0x20
#define SHF_INFO_LINK 0x40
#define SHF_LINK_ORDER  0x80
#define SHF_OS_NONCONFORMING 0x100
#define SHF_GROUP 0x200
#define SHF_TLS   0x400
#define SHF_MASKOS  0x0ff00000
#define SHF_MASKPROC  0xf0000000
/****************************************/
/* Program Header - figure 2-1, page 2-2 */
/****************************************/

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;

/* segment types - figure 2-2, page 2-3 */

#define PT_NULL   0
#define PT_LOAD   1
#define PT_DYNAMIC  2
#define PT_INTERP 3
#define PT_NOTE   4
#define PT_SHLIB  5
#define PT_PHDR   6
#define PT_TLS    7
#define PT_NUM    8
#define PT_LOOS   0x60000000
#define PT_HIOS   0x6fffffff
#define PT_LOPROC 0x70000000
#define PT_HIPROC 0x7fffffff
#define PT_GNU_EH_FRAME (PT_LOOS + 0x474e550)
#define PT_GNU_STACK    (PT_LOOS + 0x474e551)
#define PT_GNU_RELRO    (PT_LOOS + 0x474e552)
#define PT_L4_STACK     (PT_LOOS + 0x12)
#define PT_L4_KIP       (PT_LOOS + 0x13)
#define PT_L4_AUX       (PT_LOOS + 0x14)
```

```
/* segment permissions - page 2-3 */

#define PF_X    0x1
#define PF_W    0x2
#define PF_R    0x4
#define PF_MASKOS 0x0ff00000
#define PF_MASKPROC 0x7fffffff

/* Legal values for note segment descriptor types for core files. */

#define NT_PRSTATUS 1
#define NT_FPREGSET 2
#define NT_PRPSINFO 3
#define NT_PRXREG 4
#define NT_TASKSTRUCT 4
#define NT_PLATFORM 5
#define NT_AUXV   6
#define NT_GWINDOWS 7
#define NT_ASRS   8
#define NT_PSTATUS  10
#define NT_PSINFO 13
#define NT_PRCRED 14
#define NT_UTSNAME  15
#define NT_LWPSTATUS  16
#define NT_LWPSINFO 17
#define NT_PRFPXREG 20
/* Legal values for the note segment descriptor types for object files.  */

#define NT_VERSION  1
/* Dynamic structure - figure 2-9, page 2-12 */

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;

#define DT_NULL   0
#define DT_NEEDED 1
#define DT_PLTRELSZ 2
#define DT_PLTGOT 3
#define DT_HASH   4
#define DT_STRTAB 5
#define DT_SYMTAB 6
#define DT_RELA   7
#define DT_RELASZ 8
#define DT_RELAENT  9
#define DT_STRSZ  10
#define DT_SYMENT 11
#define DT_INIT   12
#define DT_FINI   13
#define DT_SONAME 14
#define DT_RPATH  15
#define DT_SYMBOLIC 16
#define DT_REL    17
#define DT_RELSZ  18
#define DT_RELENT 19
#define DT_PTRREL 20
#define DT_DEBUG  21
#define DT_TEXTREL  22
#define DT_JMPREL 23
#define DT_BIND_NOW 24
#define DT_INIT_ARRAY 25
#define DT_FINI_ARRAY 26
#define DT_INIT_ARRAYSZ 27
#define DT_FINI_ARRAYSZ 28
#define DT_RUNPATH  29
#define DT_FLAGS  30
#define DT_ENCODING 32
#define DT_PREINIT_ARRAY 32
#define DT_PREINIT_ARRAYSZ 33
#define DT_NUM    34
#define DT_LOOS   0x6000000d
#define DT_HIOS   0x6ffff000
#define DT_LOPROC 0x70000000
#define DT_HIPROC 0x7fffffff
/* Values of `d_un.d_val' in the DT_FLAGS entry.  */
```

```
#define DF_ORIGIN 0x00000001
#define DF_SYMBOLIC 0x00000002
#define DF_TEXTREL  0x00000004
#define DF_BIND_NOW 0x00000008
#define DF_STATIC_TLS 0x00000010
/* State flags selectable in the 'd_un.d_val' element of the DT_FLAGS_1
   entry in the dynamic section.  */
#define DF_1_NOW  0x00000001
#define DF_1_GLOBAL 0x00000002
#define DF_1_GROUP  0x00000004
#define DF_1_NODELETE 0x00000008
#define DF_1_LOADFLTR 0x00000010
#define DF_1_INITFIRST  0x00000020
#define DF_1_NOOPEN 0x00000040
#define DF_1_ORIGIN 0x00000080
#define DF_1_DIRECT 0x00000100
#define DF_1_TRANS  0x00000200
#define DF_1_INTERPOSE  0x00000400
#define DF_1_NODEFLIB 0x00000800
#define DF_1_NODUMP 0x00001000
#define DF_1_CONFALT  0x00002000
#define DF_1_ENDFILTEE  0x00004000
#define DF_1_DISPRELDNE 0x00008000
#define DF_1_DISPRELPND 0x00010000
/* Flags for the feature selection in DT_FEATURE_1.  */
#define DTF_1_PARINIT 0x00000001
#define DTF_1_CONFEXP 0x00000002

/* Flags in the DT_POSFLAG_1 entry effecting only the next DT_* entry.  */
#define DF_P1_LAZYLOAD  0x00000001
#define DF_P1_GROUPPERM 0x00000002
/* Relocation - page 1-21, figure 1-20 */

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)

/* Relocation types (processor specific) - page 1-23, figure 1-22 */

#define R_386_NONE  0
#define R_386_32  1
#define R_386_PC32  2
#define R_386_GOT32 3
#define R_386_PLT32 4
#define R_386_COPY  5
#define R_386_GLOB_DAT  6
#define R_386_JMP_SLOT  7
#define R_386_RELATIVE  8
#define R_386_GOTOFF  9
#define R_386_GOTPC 10
#define R_386_32PLT    11
#define R_386_TLS_TPOFF    14   /* Offset in static TLS block */
#define R_386_TLS_IE     15   /* Address of GOT entry for static TLS
             block offset */
#define R_386_TLS_GOTIE    16   /* GOT entry for static TLS block
             offset */
#define R_386_TLS_LE     17   /* Offset relative to static TLS
             block */
#define R_386_TLS_GD     18   /* Direct 32 bit for GNU version of
             general dynamic thread local data */
#define R_386_TLS_LDM    19   /* Direct 32 bit for GNU version of
```

```
             local dynamic thread local data
             in LE code */
#define R_386_16     20
#define R_386_PC16     21
#define R_386_8      22
#define R_386_PC8    23
#define R_386_TLS_GD_32    24   /* Direct 32 bit for general dynamic
             thread local data */
#define R_386_TLS_GD_PUSH  25   /* Tag for pushl in GD TLS code */
#define R_386_TLS_GD_CALL  26   /* Relocation for call to
             __tls_get_addr() */
#define R_386_TLS_GD_POP   27   /* Tag for popl in GD TLS code */
#define R_386_TLS_LDM_32   28   /* Direct 32 bit for local dynamic
             thread local data in LE code */
#define R_386_TLS_LDM_PUSH 29   /* Tag for pushl in LDM TLS code */
#define R_386_TLS_LDM_CALL 30   /* Relocation for call to
             __tls_get_addr() in LDM code */
#define R_386_TLS_LDM_POP  31   /* Tag for popl in LDM TLS code */
#define R_386_TLS_LDO_32   32   /* Offset relative to TLS block */
#define R_386_TLS_IE_32    33   /* GOT entry for negated static TLS
             block offset */
#define R_386_TLS_LE_32    34   /* Negated offset relative to static
             TLS block */
#define R_386_TLS_DTPMOD32 35   /* ID of module containing symbol */
#define R_386_TLS_DTPOFF32 36   /* Offset in TLS block */
#define R_386_TLS_TPOFF32  37   /* Negated offset in static TLS block */
/* Keep this the last entry.  */
#define R_386_NUM    38

/* ARM specific declarations */

/* Processor specific flags for the ELF header e_flags field.  */
#define EF_ARM_RELEXEC     0x01
#define EF_ARM_HASENTRY    0x02
#define EF_ARM_INTERWORK   0x04
#define EF_ARM_APCS_26     0x08
#define EF_ARM_APCS_FLOAT  0x10
#define EF_ARM_PIC         0x20
#define EF_ARM_ALIGN8      0x40   /* 8-bit structure alignment is in use */
#define EF_ARM_NEW_ABI     0x80
#define EF_ARM_OLD_ABI     0x100

/* Other constants defined in the ARM ELF spec. version B-01.  */
/* NB. These conflict with values defined above.  */
#define EF_ARM_SYMSARESORTED  0x04
#define EF_ARM_DYNSYMSUSESEGIDX 0x08
#define EF_ARM_MAPSYMSFIRST 0x10
#define EF_ARM_EABIMASK   0XFF000000

#define EF_ARM_EABI_VERSION(flags) ((flags) & EF_ARM_EABIMASK)
#define EF_ARM_EABI_UNKNOWN  0x00000000
#define EF_ARM_EABI_VER1     0x01000000
#define EF_ARM_EABI_VER2     0x02000000

/* Additional symbol types for Thumb */
#define STT_ARM_TFUNC      0xd

/* ARM-specific values for sh_flags */
#define SHF_ARM_ENTRYSECT  0x10000000   /* Section contains an entry point */
#define SHF_ARM_COMDEF     0x80000000   /* Section may be multiply defined
             in the input to a link step */

/* ARM-specific program header flags */
#define PF_ARM_SB          0x10000000   /* Segment contains the location
             addressed by the static base */

/* ARM relocs.  */
#define R_ARM_NONE    0 /* No reloc */
#define R_ARM_PC24    1 /* PC relative 26 bit branch */
#define R_ARM_ABS32   2 /* Direct 32 bit  */
#define R_ARM_REL32   3 /* PC relative 32 bit */
#define R_ARM_PC13    4
#define R_ARM_ABS16   5 /* Direct 16 bit */
#define R_ARM_ABS12   6 /* Direct 12 bit */
#define R_ARM_THM_ABS5    7
#define R_ARM_ABS8    8 /* Direct 8 bit */
#define R_ARM_SBREL32   9
#define R_ARM_THM_PC22    10
#define R_ARM_THM_PC8   11
#define R_ARM_AMP_VCALL9  12
#define R_ARM_SWI24   13
#define R_ARM_THM_SWI8    14
#define R_ARM_XPC25   15
#define R_ARM_THM_XPC22   16
#define R_ARM_COPY    20  /* Copy symbol at runtime */
#define R_ARM_GLOB_DAT    21  /* Create GOT entry */
#define R_ARM_JUMP_SLOT   22  /* Create PLT entry */
```

```
#define R_ARM_RELATIVE    23  /* Adjust by program base */
#define R_ARM_GOTOFF    24  /* 32 bit offset to GOT */
#define R_ARM_GOTPC   25  /* 32 bit PC relative offset to GOT */
#define R_ARM_GOT32   26  /* 32 bit GOT entry */
#define R_ARM_PLT32   27  /* 32 bit PLT address */
#define R_ARM_ALU_PCREL_7_0 32
#define R_ARM_ALU_PCREL_15_8  33
#define R_ARM_ALU_PCREL_23_15 34
#define R_ARM_LDR_SBREL_11_0  35
#define R_ARM_ALU_SBREL_19_12 36
#define R_ARM_ALU_SBREL_27_20 37
#define R_ARM_GNU_VTENTRY 100
#define R_ARM_GNU_VTINHERIT 101
#define R_ARM_THM_PC11    102 /* thumb unconditional branch */
#define R_ARM_THM_PC9   103 /* thumb conditional branch */
#define R_ARM_RXPC25    249
#define R_ARM_RSBREL32    250
#define R_ARM_THM_RPC22   251
#define R_ARM_RREL32    252
#define R_ARM_RABS22    253
#define R_ARM_RPC24   254
#define R_ARM_RBASE   255
/* Keep this the last entry.  */
#define R_ARM_NUM   256

/* AMD x86-64 relocations.  */
#define R_X86_64_NONE   0 /* No reloc */
#define R_X86_64_64   1 /* Direct 64 bit  */
#define R_X86_64_PC32   2 /* PC relative 32 bit signed */
#define R_X86_64_GOT32    3 /* 32 bit GOT entry */
#define R_X86_64_PLT32    4 /* 32 bit PLT address */
#define R_X86_64_COPY   5 /* Copy symbol at runtime */
#define R_X86_64_GLOB_DAT 6 /* Create GOT entry */
#define R_X86_64_JUMP_SLOT  7 /* Create PLT entry */
#define R_X86_64_RELATIVE 8 /* Adjust by program base */
#define R_X86_64_GOTPCREL 9 /* 32 bit signed PC relative
             offset to GOT */
#define R_X86_64_32   10  /* Direct 32 bit zero extended */
#define R_X86_64_32S    11  /* Direct 32 bit sign extended */
#define R_X86_64_16   12  /* Direct 16 bit zero extended */
#define R_X86_64_PC16   13  /* 16 bit sign extended pc relative */
#define R_X86_64_8    14  /* Direct 8 bit sign extended  */
#define R_X86_64_PC8    15  /* 8 bit sign extended pc relative */
#define R_X86_64_DTPMOD64 16  /* ID of module containing symbol */
#define R_X86_64_DTPOFF64 17  /* Offset in module's TLS block */
#define R_X86_64_TPOFF64  18  /* Offset in initial TLS block */
#define R_X86_64_TLSGD    19  /* 32 bit signed PC relative offset
             to two GOT entries for GD symbol */
#define R_X86_64_TLSLD    20  /* 32 bit signed PC relative offset
             to two GOT entries for LD symbol */
#define R_X86_64_DTPOFF32 21  /* Offset in TLS block */
#define R_X86_64_GOTTPOFF 22  /* 32 bit signed PC relative offset
             to GOT entry for IE symbol */
#define R_X86_64_TPOFF32  23  /* Offset in initial TLS block */

#define R_X86_64_NUM    24

/* Symbol Table Entry - page 1-17, figure 1-16 */

#define 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))
```

```
/* Symbol Binding - page 1-18, figure 1-17 */

#define STB_LOCAL 0
#define STB_GLOBAL  1
#define STB_WEAK  2
#define STB_LOOS  10
#define STB_HIOS  12
#define STB_LOPROC  13
#define STB_HIPROC  15
/* Symbol Types - page 1-19, figure 1-18 */

#define STT_NOTYPE  0
#define STT_OBJECT  1
#define STT_FUNC  2
#define STT_SECTION 3
#define STT_FILE  4
#define STT_LOOS  10
#define STT_HIOS  12
#define STT_LOPROC  13
#define 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;

/* Some helpers */
static inline int l4util_elf_check_magic(ElfW(Ehdr) *hdr);
static inline int l4util_elf_check_arch(ElfW(Ehdr) *hdr);
static inline ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) *hdr);


/* Implemeantions */
static inline
int l4util_elf_check_magic(ElfW(Ehdr) *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) *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) *hdr)
{
  return (ElfW(Phdr) *)((char *)hdr + hdr->e_phoff);
}
#endif /* _L4_EXEC_ELF_H */
```

15.413 l4/util/getopt.h File Reference

getopt

`#include <l4/sys/compiler.h>`
Include dependency graph for getopt.h:

15.413.1 Detailed Description

getopt

Definition in file getopt.h.

15.414 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;
```

```

/* Index in ARGV of the next element to be scanned.
   This is used for communication to and from the caller
   and for communication between successive calls to 'getopt'.

   On entry to 'getopt', zero means this is the first call; initialize.

   When 'getopt' returns -1, this is the index of the first of the
   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 */
```

15.415 l4/util/keymap.h File Reference

Event to ASCII key mapping.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for keymap.h:

15.415.1 Detailed Description

Event to ASCII key mapping.

Definition in file keymap.h.

15.416 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__ */
```

15.417 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 native 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.
- l4_addr_t l4util_memdesc_vm_high (l4_kernel_info_t ∗kinfo)

  Return end of virtual memory.

15.418 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  */
```

```
#pragma once

#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>



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 *);

L4_CV l4_addr_t l4util_memdesc_vm_high(
    l4_kernel_info_t *kinfo);

EXTERN_C_END

#define l4util_kip_for_each_feature(s)        \
    for (s += strlen(s) + 1; *s; s += strlen(s) + 1)

```

15.419 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").

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.

15.419.1 Detailed Description

Kernel Info Page access functions.

Definition in file kip.h.

15.420 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
```

```
 * invalidate any other reasons why the executable file might be covered by
 * the GNU General Public License.
 */
#pragma once

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

#include <l4/sys/__kip-arch.h>

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,
};

#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;

/*****************************************************************************
 * 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
{
  unsigned long h1, l;
  unsigned long *c;

  if (sizeof(unsigned long) == 8)
    return kip->_clock_val;

  c = (unsigned long *)&kip->_clock_val;
  do
    {
      h1 = c[1];
      l4_mb();
      l  = c[0];
      l4_mb();
    }
  while (h1 != c[1]);

  return ((unsigned long long)h1 << 32) | l;
```

```
}

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];
}
```

15.421 l4/util/kprintf.h File Reference

printf using the kernel debugger

`#include <l4/sys/compiler.h>`

Include dependency graph for kprintf.h:

15.421.1 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.

15.422 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__ */
```

15.423 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.

15.423.1 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.

15.423.2 Function Documentation

15.423.2.1 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

15.423.2.2 l4la_avail()

```
l4_size_t l4la_avail (
            l4la_free_t ** first )
```

Show available memory in pool.

Parameters

first	list identifier

15.423.2.3 l4la_dump()

```
void l4la_dump (
            l4la_free_t ** first )
```

Show all list members.

Parameters

first	list identifier

15.423.2.4 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

15.423.2.5 l4la_init()

```
void l4la_init (
            l4la_free_t ** first )
```

Init memory pool.

Parameters

first	list identifier

15.424 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
```

15.425 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:

15.425.1 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.

15.426 lock.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_LOCK_H__
00019 #define __L4UTIL_LOCK_H__
00020
00021 #include <l4/sys/thread.h>
00022 #include <l4/sys/compiler.h>
00023 #include <l4/util/atomic.h>
00024
00025 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
```

15.427 l4/util/mb_info.h File Reference

Multiboot info structure as defined by GRUB.

`#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
- 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

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_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.

15.427.1 Detailed Description

Multiboot info structure as defined by GRUB.

Definition in file mb_info.h.

15.427.2 Macro Definition Documentation

15.427.2.1 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 263 of file mb_info.h.

15.427.2.2 MB_ARD_MEMORY

```
#define MB_ARD_MEMORY 1
```

usable memory "Type", all others are reserved.

Definition at line 62 of file mb_info.h.

15.427.2.3 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 68 of file mb_info.h.

15.428 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/l4int.h>
00024
00031 typedef struct
00032 {
00033   l4_uint32_t mod_start;
00034   l4_uint32_t mod_end;
00035   l4_uint32_t cmdline;
00036   l4_uint32_t pad;
00037 } l4util_mb_mod_t;
00038
00039
00047 typedef struct __attribute__((packed))
00048 {
00049   l4_uint32_t struct_size;
00050   l4_uint64_t addr;
00051   l4_uint64_t size;
00052   l4_uint32_t type;
00053   /* unspecified optional padding... */
00054 } l4util_mb_addr_range_t;
00055
00056 #define l4util_mb_for_each_mmap_entry(i, mbi) \
00057   for (i = (l4util_mb_addr_range_t *)(unsigned long)mbi->mmap_addr; \
00058        (unsigned long)i < (unsigned long)mbi->mmap_addr + mbi->mmap_length; \
00059        i = (l4util_mb_addr_range_t *)((unsigned long)i + mmap->struct_size + sizeof (mmap->struct_size)))
00060
00062 #define MB_ARD_MEMORY   1
00063
00068 #define MB_ART_MEMORY   1
00069 #define MB_ART_RESERVED 2
```

```
#define MB_ART_ACPI     3
#define MB_ART_NVS      4
#define MB_ART_UNUSABLE 5
typedef struct
{
  l4_uint32_t size;
  l4_uint8_t drive_number;
  l4_uint8_t drive_mode;
  l4_uint16_t drive_cylinders;
  l4_uint8_t drive_heads;
  l4_uint8_t drive_sectors;
  l4_uint16_t drive_ports[0];
} l4util_mb_drive_t;

/* Drive Mode.  */
#define MB_DI_CHS_MODE    0
#define MB_DI_LBA_MODE    1


typedef struct
{
  l4_uint16_t version;
  l4_uint16_t cseg;
  l4_uint32_t offset;
  l4_uint16_t cseg_16;
  l4_uint16_t dseg_16;
  l4_uint16_t cseg_len;
  l4_uint16_t cseg_16_len;
  l4_uint16_t dseg_16_len;
} l4util_mb_apm_t;


typedef struct
{
  l4_uint8_t signature[4];
  l4_uint16_t version;
  l4_uint32_t oem_string;
  l4_uint32_t capabilities;
  l4_uint32_t video_mode;
  l4_uint16_t total_memory;
  l4_uint16_t oem_software_rev;
  l4_uint32_t oem_vendor_name;
  l4_uint32_t oem_product_name;
  l4_uint32_t oem_product_rev;
  l4_uint8_t reserved[222];
  l4_uint8_t oem_data[256];
} __attribute__((packed)) l4util_mb_vbe_ctrl_t;


typedef struct
{
  l4_uint16_t mode_attributes;
  l4_uint8_t win_a_attributes;
  l4_uint8_t win_b_attributes;
  l4_uint16_t win_granularity;
  l4_uint16_t win_size;
  l4_uint16_t win_a_segment;
  l4_uint16_t win_b_segment;
  l4_uint32_t win_func;
  l4_uint16_t bytes_per_scanline;
  l4_uint16_t x_resolution;
  l4_uint16_t y_resolution;
  l4_uint8_t x_char_size;
  l4_uint8_t y_char_size;
  l4_uint8_t number_of_planes;
  l4_uint8_t bits_per_pixel;
  l4_uint8_t number_of_banks;
  l4_uint8_t memory_model;
  l4_uint8_t bank_size;
  l4_uint8_t number_of_image_pages;
  l4_uint8_t reserved0;
  l4_uint8_t red_mask_size;
  l4_uint8_t red_field_position;
  l4_uint8_t green_mask_size;
  l4_uint8_t green_field_position;
  l4_uint8_t blue_mask_size;
  l4_uint8_t blue_field_position;
  l4_uint8_t reserved_mask_size;
  l4_uint8_t reserved_field_position;
  l4_uint8_t direct_color_mode_info;
  l4_uint32_t phys_base;
  l4_uint32_t reserved1;
  l4_uint16_t reversed2;
  l4_uint16_t linear_bytes_per_scanline;
  l4_uint8_t banked_number_of_image_pages;
  l4_uint8_t linear_number_of_image_pages;
  l4_uint8_t linear_red_mask_size;
```

```
  l4_uint8_t linear_red_field_position;
  l4_uint8_t linear_green_mask_size;
  l4_uint8_t linear_green_field_position;
  l4_uint8_t linear_blue_mask_size;
  l4_uint8_t linear_blue_field_position;
  l4_uint8_t linear_reserved_mask_size;
  l4_uint8_t linear_reserved_field_position;
  l4_uint32_t max_pixel_clock;

  /* The VBE spec says this structure should have a size of 256 bytes but
   * the described structure layout is only 255 bytes... */
  l4_uint8_t reserved3[189 + 1];
} __attribute__ ((packed)) l4util_mb_vbe_mode_t;


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;

#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_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

#endif
```

15.429 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.

15.429.1 Detailed Description

comfortable command-line parsing

Date

2002

Author

Jork Loeser jork.loeser@inf.tu-dresden.de

Definition in file parse_cmd.h.

15.430 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
00028
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
```

15.431 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.

15.431.1 Detailed Description

Simple Pseudo-Random Number Generator.

Date

1998

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file rand.h.

15.432 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 */
```

15.433 l4/util/reboot.h File Reference

Machine restarting functions.

`#include <l4/sys/compiler.h>`
Include dependency graph for reboot.h:

Functions

- void l4util_reboot (void))

  Machine reboot.

15.433.1 Detailed Description

Machine restarting functions.

Definition in file reboot.h.

15.434 reboot.h

```
00001
00005 /*
00006  * (c) 2000-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *               Norman Feske <nf2@os.inf.tu-dresden.de>
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 _L4UTIL_REBOOT_H
00016 #define _L4UTIL_REBOOT_H
00017
00018 #include <l4/sys/compiler.h>
00019
00029 EXTERN_C_BEGIN
00030 L4_CV void l4util_reboot(void) __attribute__ ((__noreturn__));
00031 EXTERN_C_END
00032
00033 #endif /* ! _L4UTIL_REBOOT_H */
```

15.435 l4/util/sll.h File Reference

List implemenation.

```
#include <stdlib.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for sll.h:

15.435.1 Detailed Description

List implemenation.

Definition in file sll.h.

15.436 sll.h

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Ronald Aigner <ra3@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__SLL_H__
00014 #define __L4UTIL__SLL_H__
00015
00016 #include <stdlib.h>
00017
00018 #include <l4/sys/compiler.h>
00019
00020 EXTERN_C_BEGIN
00021
00022 /*
00023  * the linked list structure
00024  */
00025
00026 typedef struct slist_t
00027 {
00028   struct slist_t *next;   /* pointer to next node */
00029   void *data;          /* void pointer for user data */
```

```
} slist_t;

/*
 * function prototypes
 */
static inline slist_t*
list_new_entry(void *data);

static inline slist_t*
list_append(slist_t *list, slist_t *new_node);

static inline slist_t*
list_remove(slist_t *list, slist_t *node);

static inline void
list_free_entry(slist_t **list);

static inline unsigned char
list_is_empty(slist_t *list);

static inline slist_t*
list_get_at(slist_t *list, int n);

static inline slist_t*
list_add(slist_t *list, slist_t *new_node);

static inline void
list_insert_after(slist_t *after, slist_t *new_node);

static inline int
list_elements(slist_t *head);

/*
 *  allocateNode()
 *  allocate a new node.
 *
 *  Parameters:
 *  void    *data       a generic pointer to object data
 *
 *  Return Values:
 *  pointer to slist_t if succeeds
 *  NULL otherwise
 *
 */
static inline slist_t*
list_new_entry(void *data)
{
  slist_t *sll;

  sll = (slist_t *)malloc(sizeof(slist_t));
  if (!sll)
    return ((slist_t *) NULL);

  sll->data=data;
  sll->next=NULL;

  return (sll);
}

/*
 *  appendNode()
 *  appends a node to the end of a list
 *
 *  Parameters:
 *  slist_t *head      - modify the list
 *  slist_t *new       - appends this node
 *
 *  Return Values:
 *  the new list
 *
 */
static inline slist_t*
list_append(slist_t *head, slist_t *new_node)
{
  slist_t *ret = head;
  if (!head)
    return new_node;

  while (head->next)
    head = head->next;
  head->next = new_node;
  return ret;
}

/*
 *  insertNode()
 *  insert a node at the beginning of a list
```

```
 *
 *  Parameters:
 *  slist_t *head      - modify this list
 *  slist_t *new       - appends this node
 *
 *  Return Values:
 *  the new list
 *
 */
static inline slist_t*
list_add(slist_t *head, slist_t *new_node)
{
  if (!new_node)
    return head;
  new_node->next = head;
  return new_node;
}

/*
 *  insertNode()
 *  insert a node at the beginning of a list
 *
 *  Parameters:
 *  slist_t *head      - modify this list
 *  slist_t *new       - appends this node
 *
 *  Return Values:
 *  the new list
 *
 */
static inline void
list_insert_after(slist_t *after, slist_t *new_node)
{
  if (!new_node)
    return;
  if (!after)
    return;
  new_node->next = after->next;
  after->next = new_node;
}


/*
 *  emptyList()
 *  check if a list variable is NULL
 *
 *  Parameters:
 *  slist_t *list      list
 *
 *  Return Values:
 *  TRUE    if empty
 *  FALSE   if not empty
 *
 */
static inline unsigned char
list_is_empty(slist_t *list)
{
  return ((list) ? 0 : 1);
}

/*
 *  delNode()
 *  remove a node from a list
 *
 *  Parameters:
 *  slist_t  *head      - list to modify
 *  slist_t *node       - node to remove
 *
 *  Return Values:
 *  none
 *
 */
static inline slist_t*
list_remove(slist_t *head, slist_t *node)
{
  slist_t *ret = head;
  if (list_is_empty(head))
    return ret;
  if (!node)
    return ret;

  if (head == node)
    {
      ret = head->next;
    }
  else
    {
```

```
      while (head && (head->next != node))
  head = head->next;
      if (!head)
  return ret;
      else
  head->next = node->next;
    }
  list_free_entry(&node);
  return ret;
}

/*
 *  freeNode()
 *  frees a node
 *
 *  Parameters:
 *  slist_t  *list  node to free
 *
 *  Return Values:
 *  none
 *
 */
static inline void
list_free_entry(slist_t **list)
{
  if (*list)
    {
      free ((void *) (*list));
      (*list)=NULL;
    }
}


/*
 *  getNthNode()
 *  get nth node in a list
 *
 *  Parameters:
 *  slist_t *list       - the head list
 *  int n           - return the node
 *  Return Values:
 *  a pointer to the list at position n
 *  NULL if there's no such node at posion n
 *
 */
static inline slist_t*
list_get_at(slist_t *list, int n)
{
  int j=0;

  while (list)
    {
      j++;
      if (j == n)
  return (list);
      list = list->next;
    }

  return ((slist_t *) NULL);
}

/*
 *  numNodes()
 *  returns number of nodes in the list
 *
 *  Parameters:
 *  slist_t  *head      - the head node of the list
 *
 *  Return Values:
 *  number of node/s
 *
 */
static inline int
list_elements(slist_t *head)
{
  register int n;
  for (n=0; head; head=head->next) n++;
  return (n);
}

EXTERN_C_END

#endif  /* __L4UTIL__SLL_H__ */
```

15.437 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.

15.437.1 Detailed Description

Split a range in log2 aligned and size-aligned chunks.

Definition in file splitlog2.h.

15.438 splitlog2.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 Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */
#ifndef __L4UTIL__INCLUDE__SPLITLOG2_H__
#define __L4UTIL__INCLUDE__SPLITLOG2_H__

#include <l4/sys/linkage.h>
#include <l4/sys/err.h>
#include <l4/util/bitops.h>

EXTERN_C_BEGIN

L4_INLINE long
l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
                     long (*handler)(l4_addr_t s, l4_addr_t e, int log2size));

L4_INLINE l4_addr_t
l4util_splitlog2_size(l4_addr_t start, l4_addr_t end);

EXTERN_C_END

/* Implementation */

L4_INLINE long
l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
                     long (*handler)(l4_addr_t s, l4_addr_t e, int log2size))
{
  if (end < start)
    return -L4_EINVAL;

  while (start <= end)
    {
      long retval;
      int len2 = l4util_splitlog2_size(start, end);
      l4_addr_t len = 1UL << len2;
      if ((retval = handler(start, start + len - 1, len2)))
  return retval;
      start += len;
    }
  return 0;
}

L4_INLINE l4_addr_t
l4util_splitlog2_size(l4_addr_t start, l4_addr_t end)
{
  int start_bits = l4util_bsf(start);
  int len_bits = l4util_bsr(end - start + 1);
  if (start_bits != -1 && len_bits > start_bits)
    len_bits = start_bits;

  return len_bits;
}

#endif /* ! __L4UTIL__INCLUDE__SPLITLOG2_H__ */
```

15.439 l4/util/stack.h File Reference

Some helper functions for stack manipulation.

```
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/util/stack_impl.h>
```

Include dependency graph for stack.h:

Functions

- l4_addr_t l4util_stack_get_sp (void)

  Get current stack pointer.

15.439.1 Detailed Description

Some helper functions for stack manipulation.

Newer versions of gcc forbid to cast the lvalue of an expression resulting that the following expression is invalid:

∗–((l4_threadid_t)esp) = tid

Date

03/2004

Author

Frank Mehnert `fm3@os.inf.tu-dresden.de`

Definition in file stack.h.

15.439.2 Function Documentation

15.439.2.1 l4util_stack_get_sp()

```
l4_addr_t l4util_stack_get_sp (
            void  )  [inline]
```

Get current stack pointer.

Returns

stack pointer.

Definition at line 23 of file stack_impl.h.

References EXTERN_C_END.

15.440 stack.h

```
00001
00012 /*
00013  * (c) 2003-2009 Author(s)
00014  *     economic rights: Technische Universität Dresden (Germany)
00015  * This file is part of TUD:OS and distributed under the terms of the
00016  * GNU Lesser General Public License 2.1.
00017  * Please see the COPYING-LGPL-2.1 file for details.
00018  */
00019
00020 #ifndef _L4UTIL_STACK_H
00021 #define _L4UTIL_STACK_H
00022
00023 #include <l4/sys/types.h>
00024 #include <l4/sys/compiler.h>
00025
00026 EXTERN_C_BEGIN
00027
00028 L4_INLINE void l4util_stack_push_mword(l4_addr_t *stack, l4_mword_t val);
00029
00030 /****************************************************************************/
00036 L4_INLINE l4_addr_t l4util_stack_get_sp(void);
00037
00038 /*
00039  * Implementations.
00040  */
00041
00042 #include <l4/util/stack_impl.h>
00043
00044 L4_INLINE void
00045 l4util_stack_push_mword(l4_addr_t *stack, l4_mword_t val)
00046 {
00047   l4_mword_t *esp = (l4_mword_t*)(*stack);
00048   *--esp = val;
00049   *stack = (l4_addr_t)esp;
00050 }
00051
00052 EXTERN_C_END
00053
00054 #endif
```

15.441 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.

15.441.1 Detailed Description

Low-level Thread Functions.

Date

1997

Author

Sebastian Schönberg

Definition in file thread.h.

15.441.2 Macro Definition Documentation

15.441.2.1 __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.

15.442 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
00056
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 */
```

15.443 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 = 2↩UL, L4_THREAD_STATS_OP = 3UL,
  L4_THREAD_VCPU_RESUME_OP = 4UL, L4_THREAD_REGISTER_DELETE_IRQ_OP = 5UL, L4_THR↩EAD_MODIFY_SENDER_OP = 6UL, L4_THREAD_VCPU_CONTROL_OP = 7UL ,
  L4_THREAD_X86_GDT_OP = 0x10UL, L4_THREAD_ARM_TPIDRURO_OP = 0x10UL, L4_THREAD_A↩MD64_SET_SEGMENT_BASE_OP = 0x12UL, L4_THREAD_AMD64_GET_SEGMENT_INFO_OP = 0x13↩UL,
  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_TH↩READ_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_R↩EGS_TRIGGER_EXCEPTION = 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_N↩OTHROW

  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 or disable 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_N↩OTHROW

  Enable or disable 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 or disable 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 or disable 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 modifiction 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.

15.443.1 Detailed Description

Common thread related definitions.

Definition in file thread.h.

15.444 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
```

```
 * 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/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_FPAGE_RWX).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());
}
```

15.445 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.

15.445.1 Detailed Description

ARM-specific thread related definitions.

Definition in file thread.h.

15.446 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());
}
```

15.447 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 = 0, L4VBUS_ROOT_BUS = 0 }

  Constants for device nodes.
- enum L4vbus_dma_domain_assign_flags { L4VBUS_DMAD_UNBIND = 0, L4VBUS_DMAD_BIND = 1, L4↩VBUS_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 human 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_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_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.

15.447.1 Detailed Description

Description of the vbus C API.

Definition in file vbus.h.

15.447.2 Enumeration Type Documentation

15.447.2.1 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.

15.448 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 = 0,
  L4VBUS_ROOT_BUS = 0,
};

__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_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_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

```

15.449 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.

15.449.1 Detailed Description

This header file contains descriptions of vbus related data types and constants.

Definition in file vbus_types.h.

15.449.2 Enumeration Type Documentation

15.449.2.1 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.

15.450 vbus_types.h

```
/*
 * (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_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 16

Example Documentation

16.1 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 substract 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;
}
```

16.2 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");
```

16.3 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 substractions and negations.\n");

  // Wait for client requests
  server.loop();

  return 0;
}
```

16.4 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::Search_addr,
                                          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;
}
```

16.5 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::Search_addr,
                                           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;
}
```

16.6 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::Search_addr,
                                        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;
}
```

16.7 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"
);
```

16.8 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_SEARCH_ADDR, 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;
}
```

16.9 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::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;
}
```

16.10 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;
}
```

16.11 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");
```

16.12 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;
}
```

16.13 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;
}
```

16.14 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;
}
```

16.15 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;
}
```

16.16 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");
```

16.17 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;
}
```

16.18 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;
}
```

16.19 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;
}
```

16.20 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");
```

16.21 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>
#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;
}
```

16.22 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;
}
```

16.23 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/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/util/thread.h>

#include <stdio.h>
#include <string.h>

static unsigned char stack2[8 << 10];
static l4_cap_idx_t thread1_cap, thread2_cap;

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");
    }
}

L4UTIL_THREAD_STATIC_FUNC(thread2)
{
  l4_msgtag_t tag;
  l4_msg_regs_t mr;
  unsigned i;

  printf("Thread2 up (%p)\n", l4_utcb());

  while (1)
    {
      if (l4_msgtag_has_error(tag = l4_ipc_receive(thread1_cap,
      l4_utcb(), L4_IPC_NEVER)))
        printf("IPC receive error\n");
      memcpy(&mr, l4_utcb_mr(), sizeof(mr));
      printf("Thread2 receive (%d): ", l4_msgtag_words(tag));
      for (i = 0; i < l4_msgtag_words(tag); i++)
        printf("%c", (char)mr.mr[i]);
      printf("\n");
    }

  __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))
    return 1;

  tag = l4_factory_create_thread(l4re_env()->factory, thread2_cap);
  if (l4_error(tag))
```

```
    return 1;

  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 *)l4re_env()->first_free_utcb,
                         L4RE_THIS_TASK_CAP);
  tag = l4_thread_control_commit(thread2_cap);
  if (l4_error(tag))
    return 2;

  tag = l4_thread_ex_regs(thread2_cap,
                          (l4_umword_t)thread2,
                          (l4_umword_t)(stack2 + sizeof(stack2)), 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, thread2_cap, &sp);
  if (l4_error(tag))
    return 4;

  thread1();

  return 0;
}
```

16.24 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;
}
```

16.25 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);
    }
}
```

16.26 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;

}
```

Index

__Iface
L4::Kobject_2t, 1099
L4::Kobject_3t, 1103
__Iface_list
L4::Kobject_2t, 1099
L4::Kobject_3t, 1103
__L4UTIL_THREAD_FUNC
l4/util/thread.h, 2328
__check_protocols__
L4::Kobject_2t, 1100
L4::Kobject_3t, 1104
_bus
L4vbus::Device, 1542
_c
L4::Cap_base, 903
~Auto_ptr
cxx::Auto_ptr, 703
~Be_file_system
L4Re::Vfs::Be_file_system, 1449
~H_list_item_t
cxx::H_list_item_t, 807

a
L4Re::Video::Pixel_info, 1495
ARM Virtual Registers (UTCB), 89
Access_width
L4Re::Mmio_space, 1350
acked
L4virtio::Svr::Dev_status, 1637, 1638
acked_bfm_t
L4virtio::Svr::Dev_status, 1636
acquire
L4Re::Inhibitor, 1337
add
L4::Ipc_svr::Timeout_queue, 1062
L4::Thread::Modify_senders, 1192
L4vcpu::State, 1582
L4virtio::Svr::Driver_mem_list_t, 1648
add_ku_mem
L4::Task, 1172
add_timeout
L4::Ipc_svr::Server_iface, 1052
L4::Ipc_svr::Timeout_queue_hooks, 1067
align_to
L4::Ipc::Msg, 668
all
L4::Kip::Mem_desc, 1087, 1088
alloc
cxx::Base_slab_static, 727
cxx::List_alloc, 815
cxx::Slab, 848
cxx::Slab_static, 852
L4Re::Cap_alloc, 1290, 1291
L4Re::Mem_alloc, 1346
L4Re::Util::Counting_cap_alloc, 1391, 1392
alloc_buffer_demand
L4::Ipc_svr::Br_manager_no_buffers, 1039
L4::Ipc_svr::Server_iface, 1052
L4Re::Util::Br_manager, 1382
alloc_descriptor
L4virtio::Driver::Virtqueue, 1598
alloc_dynamic_entry
L4Re::Util::Names::Name_space, 1422
alloc_fd
L4Re::Vfs::Fs, 1462
alloc_max
cxx::List_alloc, 816
allocate
L4Re::Dataspace, 1299
L4Re::Util::Dataspace_svr, 1398
amd64 Virtual Registers (UTCB), 629
amd64/l4/sys/cache.h, 2066
amd64/l4/sys/consts.h, 2080, 2081
amd64/l4/sys/l4int.h, 2173, 2174
amd64/l4/sys/linkage.h, 1780
amd64/l4/sys/segment.h, 1755, 1759
fiasco_amd64_segment_info, 1757
fiasco_amd64_set_fs, 1758
fiasco_amd64_set_segment_base, 1758
L4_task_ldt_x86_consts, 1757
amd64/l4/sys/utcb.h, 2228, 2230
amd64/l4/util/apic.h, 1709, 1710
amd64/l4/util/bitops_arch.h, 1786
amd64/l4/util/cpu.h, 1794, 1795
amd64/l4/util/idt.h, 1720, 1721
amd64/l4/util/irq.h, 1890, 1892
l4util_irq_acknowledge, 1891
amd64/l4/util/l4_macros.h, 1799
amd64/l4/util/mbi_argv.h, 1802, 1803
amd64/l4/util/perform.h, 1723, 1724
amd64/l4/util/port_io.h, 1765, 1766
amd64/l4/util/rdtsc.h, 1735, 1737
amd64/l4/util/spin.h, 1745
amd64/l4/util/stack_impl.h, 1806, 1807
l4util_stack_get_sp, 1806
amd64/l4/util/util.h, 1747, 1749
l4_sleep, 1748
l4util_micros2l4to, 1748
amd64/l4f/l4/sys/segment.h, 1752, 1754

fiasco_amd64_set_fs, 1753
fiasco_amd64_set_segment_base, 1754
amd64/l4f/l4/util/port_io.h, 1764, 1765
amd64/l4f/l4/util/setjmp.h, 1773, 1775
l4_thread_longjmp, 1774
l4_thread_setjmp, 1775
arm/l4/sys/atomic.h, 2261, 2262
arm/l4/sys/cache.h, 2063, 2064
arm/l4/sys/consts.h, 2079, 2080
arm/l4/sys/l4int.h, 2173
arm/l4/sys/linkage.h, 1779
arm/l4/sys/mem_op.h, 1781, 1783
arm/l4/sys/thread.h, 2339
arm/l4/sys/utcb.h, 2226, 2227
arm/l4/sys/vm.h, 1784
arm/l4/util/bitops_arch.h, 1785, 1786
arm/l4/util/cpu.h, 1793, 1794
arm/l4/util/irq.h, 1889, 1890
arm/l4/util/l4_macros.h, 1798
arm/l4/util/mbi_argv.h, 1801, 1802
arm/l4/util/stack_impl.h, 1805, 1806
l4util_stack_get_sp, 1805
arm/l4f/l4/sys/ipc.h, 2151, 2152
arm/l4f/l4/sys/syscall_defs.h, 1808, 1809
assign_dma_domain
L4vbus::Vbus, 1575, 1576
associate
L4Re::Dma_space, 1311
Atomic Instructions, 90
l4util_add8, 92
l4util_add8_res, 92
l4util_atomic_add, 92
l4util_atomic_inc, 93
l4util_cmpxchg, 93
l4util_cmpxchg16, 94
l4util_cmpxchg32, 94
l4util_cmpxchg64, 96
l4util_cmpxchg8, 96
l4util_inc8, 97
l4util_inc8_res, 97
l4util_xchg, 97
l4util_xchg16, 98
l4util_xchg32, 98
l4util_xchg8, 99
attach
L4::Irq, 1071
L4Re::Rm, 1367, 1368
L4Re::Util::Event_buffer_t, 1412
Attach_flags
L4Re::Rm, 1366
Attr
L4::Thread::Attr, 1189
Attribute
L4Re::Dma_space, 1310
Attributes
L4Re::Dma_space, 1309
Auto_ptr
cxx::Auto_ptr, 703
auto_refresh
L4Re::Video::Goos::Info, 1492
Auxiliary data, 100
avail
cxx::List_alloc, 817
avail_align
L4virtio::Virtqueue, 1678
avail_size
L4virtio::Virtqueue, 1678
Avl_map
cxx::Avl_map, 708
ax
l4_vcpu_regs_t, 1274

b
L4Re::Video::Pixel_info, 1496
backtrace.h
l4util_backtrace, 2265
Bad_descriptor
L4virtio::Svr::Bad_descriptor, 1606
Base API, 101
Base_avl_set
cxx::Bits::Base_avl_set, 766
Basic Macros, 104
L4_ALWAYS_INLINE, 105
L4_HIDDEN, 105
L4_NOTHROW, 105
Be_file_system
L4Re::Vfs::Be_file_system, 1449
begin
cxx::Bits::Base_avl_set, 767
cxx::Bits::Bst, 781, 782
bind
L4::Icu, 940
L4::Iommu, 954
L4::Thread::Attr, 1189
bind_thread
L4::Rcv_endpoint, 1135
bit
cxx::Bitmap_base, 751
Bit Manipulation, 107
l4util_bsf, 107
l4util_bsr, 108
l4util_btc, 109
l4util_btr, 109
l4util_bts, 110
l4util_clear_bit, 111
l4util_complement_bit, 111
l4util_find_first_set_bit, 112
l4util_find_first_zero_bit, 112
l4util_next_power2, 112
l4util_set_bit, 113
l4util_test_bit, 113
bit_index
cxx::Bitmap_base, 752
Bitmap
cxx::Bitmap, 746
Bitmap graphics and fonts, 115
bitmap.h

pSLIM_BMAP_START_LSB, 1903
bits_per_pixel
L4Re::Video::Pixel_info, 1496
Bits_type
cxx::Bitfield, 730
blk_size
l4virtio_block_config_t, 1702
Block_dev
L4virtio::Svr::Block_dev, 1610
bp
l4_vcpu_regs_t, 1275
buf
L4Re::Util::Event_buffer_t, 1413
buf_cp_in
L4::Ipc, 659
buf_cp_out
L4::Ipc, 660
buf_in
L4::Ipc, 660
buffer
L4Re::Util::Event_t, 1415
Buffer Registers (BRs), 116
l4_buffer_desc_consts_t, 116
bus_cap
L4vbus::Device, 1536
bx
l4_vcpu_regs_t, 1275
bytes_per_pixel
L4Re::Video::Pixel_info, 1497

c
L4::Kobject_2t, 1100
L4::Kobject_3t, 1104
C++ Exceptions, 118
C++ IPC Interface Definition., 119
CPU related functions, 120
l4util_cpu_capabilities, 120
l4util_cpu_capabilities_nocheck, 121
l4util_cpu_has_cpuid, 122
Cache Consistency, 123
l4_cache_clean_data, 123
l4_cache_coherent, 124
l4_cache_dma_coherent, 124
l4_cache_flush_data, 125
l4_cache_inv_data, 125
call
L4::Arm_smccc, 880
L4::Ipc::Iostream, 978
Cap
L4::Cap, 890, 891
L4::Ipc::Cap, 969
cap
L4::Cap_base, 896
L4::Invalid_capability, 949
L4::Kobject, 1094
cap_alloc
L4Re Capability API, 383
Cap_base
L4::Cap_base, 895, 896
cap_cast
L4, 650, 651
cap_dynamic_cast
L4, 652
cap_equal
L4::Task, 1173
cap_has_child
L4::Task, 1173
cap_received
L4::Ipc::Gen_fpage, 972
cap_reinterpret_cast
L4, 653, 654
Cap_type
L4::Cap_base, 895
cap_valid
L4::Task, 1174
Capabilities, 127
l4_cap_consts_t, 128
l4_capability_equal, 129
l4_default_caps_t, 128
l4_is_invalid_cap, 129
l4_is_valid_cap, 130
capability
L4_DISABLE_COPY, 2070
Capability allocator, 131
l4re_util_cap_last, 131
cast
L4vcpu::Vcpu, 1587
cfg_read
L4vbus::Pci_dev, 1563
L4vbus::Pci_host_bridge, 1568
cfg_write
L4vbus::Pci_dev, 1563
L4vbus::Pci_host_bridge, 1569
chain
L4::Irq_mux, 1083
change_queue_config
L4virtio::Svr::Dev_config, 1623
chars
cxx::Bitmap_base, 753
check_size
L4::Ipc::Msg, 669, 670
chkcap
L4Re, 680
chkipc
L4Re, 680
chksys
L4Re, 682–684
Chunks, 132
l4shmc_add_chunk, 132
l4shmc_chunk_capacity, 133
l4shmc_chunk_ptr, 133
l4shmc_chunk_signal, 134
l4shmc_get_chunk, 134
l4shmc_get_chunk_to, 135
l4shmc_iterate_chunk, 135
Class
L4::Kobject_2t, 1099

L4::Kobject_3t, 1103
clear
cxx::Bits::Basic_list, 776
L4::Types::Flags, 1229
L4Re::Dataspace, 1299
L4Re::Util::Dataspace_svr, 1399
L4vcpu::State, 1582
clear_bit
cxx::Bitmap_base, 753
cnt_iobmap_tlb_flush
l4_tracebuffer_status_t, 1269
Color_component
L4Re::Video::Color_component, 1482
Com_error
L4::Com_error, 906
Comfortable Command Line Parsing, 136
parse_cmdline, 136
config_get
L4vbus::Gpio_pin, 1553
config_pad
L4vbus::Gpio_module, 1546
L4vbus::Gpio_pin, 1554
config_pull
L4vbus::Gpio_pin, 1554
Console API, 139
consumed
L4virtio::Svr::Virtqueue, 1669
Consumer, 140, 144
l4shmc_chunk_consumed, 140
l4shmc_chunk_size, 141
l4shmc_enable_chunk, 141
l4shmc_enable_signal, 144
l4shmc_is_chunk_ready, 142
l4shmc_wait_any, 145
l4shmc_wait_any_to, 145
l4shmc_wait_any_try, 145
l4shmc_wait_chunk, 142
l4shmc_wait_chunk_to, 142
l4shmc_wait_chunk_try, 143
l4shmc_wait_signal, 146
l4shmc_wait_signal_to, 146
l4shmc_wait_signal_try, 147
contains
L4virtio::Svr::Driver_mem_region_t, 1656
contrib/libio-io/l4/io/io.h, 1809, 1812
control
L4::Thread, 1181
copy
L4::Cap, 892
L4::Cap_base, 897
L4Re::Util::Dataspace_svr, 1399
copy_in
L4Re::Dataspace, 1300
copy_receive_cap
L4Re::Util::Names::Name_space, 1422
copy_to
L4virtio::Svr::Data_buffer, 1618
count
L4::Kip::Mem_desc, 1088, 1089
l4_vhw_descriptor, 1281
Counting_cap_alloc
L4Re::Util::Counting_cap_alloc, 1391
cpu_disable
L4::Platform_control, 1125
cpu_enable
L4::Platform_control, 1126
create
L4::Factory, 923, 924
create_buffer
L4Re::Video::Goos, 1488
create_factory
L4::Factory, 925
create_gate
L4::Factory, 926
create_irq
L4::Factory, 927
create_task
L4::Factory, 928
create_thread
L4::Factory, 929
create_view
L4Re::Video::Goos, 1488
create_vm
L4::Factory, 930
current_flags
L4virtio::Svr::Request_processor, 1661
cx
l4_vcpu_regs_t, 1275
cxx, 643
H_list_item, 645
cxx::Auto_ptr
~Auto_ptr, 703
Auto_ptr, 703
get, 703
operator Priv_type *, 704
operator*, 704
operator->, 704
operator=, 704
Ref_type, 702
release, 705
cxx::Auto_ptr< T >, 701
cxx::Avl_map
Avl_map, 708
insert, 709
operator[], 709, 710
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, 706
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >, 710
cxx::Avl_tree
insert, 717
remove, 717
cxx::Avl_tree< Node, Get_key, Compare >, 713
cxx::Avl_tree_node, 718
cxx::Base_slab
free_objects, 723
total_objects, 723

cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 721
cxx::Base_slab_static
alloc, 727
free, 727
free_objects, 728
total_objects, 728
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 724
cxx::Bitfield
Bits_type, 730
get, 732
get_unshifted, 733
Masks, 732
Ref, 730
Ref_unshifted, 731
set, 733
set_dirty, 734
set_unshifted, 735
set_unshifted_dirty, 736
Shift_type, 731
Val, 731
val, 737
val_dirty, 737
Val_unshifted, 731
val_unshifted, 738
cxx::Bitfield< T, LSB, MSB >, 729
cxx::Bitfield< T, LSB, MSB >::Value< TT >, 739
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >, 740
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >, 742
cxx::Bitmap
Bitmap, 746
scan_zero, 746
cxx::Bitmap< BITS >, 743
cxx::Bitmap_base, 747
bit, 751
bit_index, 752
chars, 753
clear_bit, 753
operator[], 754
scan_zero, 755
set_bit, 756
word_index, 757
words, 757
cxx::Bitmap_base::Bit, 758
cxx::Bitmap_base::Char< BITS >, 759
cxx::Bitmap_base::Word< BITS >, 759
cxx::Bits, 645
cxx::Bits::Avl_map_get_key< KEY_TYPE >, 761
cxx::Bits::Avl_set_get_key< KEY_TYPE >, 761
cxx::Bits::Base_avl_set
Base_avl_set, 766
begin, 767
end, 768
erase, 769
find_node, 769
insert, 770
lower_bound_node, 770
rbegin, 770, 771
remove, 771
rend, 772
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 762
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, 773
cxx::Bits::Base_avl_set::Node
operator*, 774
operator->, 774
valid, 774
cxx::Bits::Basic_list
clear, 776
iter, 776
cxx::Bits::Basic_list< POLICY >, 775
cxx::Bits::Bst
begin, 781, 782
dir, 782, 783
end, 784
find, 785
find_node, 786
lower_bound_node, 786
rbegin, 787, 788
remove_all, 788
remove_tree, 789
rend, 789, 790
cxx::Bits::Bst< Node, Get_key, Compare >, 777
cxx::Bits::Bst_node, 791
cxx::Bits::Direction, 793
Direction_e, 794
operator!, 794
cxx::Bits::Smart_ptr_list
pop_front, 796
cxx::Bits::Smart_ptr_list< ITEM >, 795
cxx::Bits::Smart_ptr_list_item< T, STORE_T >, 797
cxx::H_list
erase, 801
insert, 802
insert_after, 802
insert_before, 803
iter, 803
pop_front, 803
remove, 804
replace, 804
cxx::H_list< T, POLICY >, 798
cxx::H_list_item_t
~H_list_item_t, 807
H_list_item_t, 806
cxx::H_list_item_t< ELEM_TYPE >, 805
cxx::H_list_t< T >, 807
cxx::List
items, 811
operator[], 811
push_back, 811
push_front, 812
remove, 812
size, 812

cxx::List< D, Alloc >, 810
cxx::List< D, Alloc >::Iter, 813
cxx::List_alloc, 814
alloc, 815
alloc_max, 816
avail, 817
free, 817
List_alloc, 814
cxx::List_item, 818
get_next_item, 820
get_prev_item, 820
insert_next_item, 820
insert_prev_item, 821
push_back, 821
push_front, 822
remove, 822
remove_me, 823
cxx::List_item::Iter, 824
remove_me, 826
cxx::List_item::T_iter< T, Poly >, 827
cxx::Lt_functor< Obj >, 830
cxx::New_allocator< _Type >, 830
cxx::Nothrow, 831
cxx::Pair
Pair, 833
cxx::Pair< First, Second >, 832
cxx::Pair_first_compare
operator(), 834
Pair_first_compare, 834
cxx::Pair_first_compare< Cmp, Typ >, 833
cxx::Ref_obj_list_item< T >, 835
cxx::Ref_ptr
get, 840
ptr, 841
Ref_ptr, 839, 840
release, 841
cxx::Ref_ptr< T, CNT >, 836
cxx::S_list
pop_front, 845
cxx::S_list< T, POLICY >, 842
cxx::Slab
alloc, 848
free, 848
cxx::Slab< Type, Slab_size, Max_free, Alloc >, 845
cxx::Slab_static
alloc, 852
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >, 849
cxx::String, 854
find, 857
from_dec, 858
from_hex, 859
starts_with, 860
String, 857
cxx::Weak_ref< T >, 861
cxx::Weak_ref_base, 863
cxx::static_vector< T, IDX >, 852

d_val
Elf32_Dyn, 866
Elf64_Dyn, 873
DF_1_CONFALT
ELF binary format, 177
DF_1_DIRECT
ELF binary format, 177
DF_1_DISPRELDNE
ELF binary format, 178
DF_1_DISPRELPND
ELF binary format, 178
DF_1_ENDFILTEE
ELF binary format, 178
DF_1_GLOBAL
ELF binary format, 178
DF_1_GROUP
ELF binary format, 178
DF_1_INTERPOSE
ELF binary format, 179
DF_1_LOADFLTR
ELF binary format, 179
DF_1_NODEFLIB
ELF binary format, 179
DF_1_NODELETE
ELF binary format, 179
DF_1_NODUMP
ELF binary format, 179
DF_1_NOOPEN
ELF binary format, 180
DF_1_NOW
ELF binary format, 180
DF_1_ORIGIN
ELF binary format, 180
DF_P1_GROUPPERM
ELF binary format, 180
DF_P1_LAZYLOAD
ELF binary format, 180
DMA API for L4Re, 148
DMA Space Interface, 149
l4re_dma_space_associate, 150
l4re_dma_space_disassociate, 150
l4re_dma_space_map, 151
l4re_dma_space_t, 149
l4re_dma_space_unmap, 152
DT_HIPROC
ELF binary format, 181
DT_LOPROC
ELF binary format, 181
DT_NULL
ELF binary format, 181
data
L4::Ipc::Varg, 1026
Data_buffer
L4virtio::Svr::Data_buffer, 1618
data_size
L4virtio::Svr::Block_request, 1615
data_space
L4Re::Vfs::Be_file, 1446
L4Re::Vfs::Regular_file, 1475

Dataspace interface, 153
l4re_ds_allocate, 154
l4re_ds_clear, 154
l4re_ds_copy_in, 155
l4re_ds_flags, 155
l4re_ds_info, 155
l4re_ds_map_flags, 154
l4re_ds_phys, 156
l4re_ds_size, 156
debug
L4Re::Debug_obj, 1308
Debug interface, 158
l4re_debug_obj_debug, 158
debugger.h
l4_debugger_get_object_name, 2119
l4_debugger_query_log_name, 2119
l4_debugger_query_log_typeid, 2120
l4_debugger_switch_log, 2120
Debugging API, 159
dec_refcnt
L4::Kobject, 1096
delete_buffer
L4Re::Video::Goos, 1488
delete_obj
L4::Task, 1174
delete_view
L4Re::Video::Goos, 1489
Demand
L4::Kobject_typeid, 1108
L4::Type_info::Demand, 1199
demand
L4::Kobject_typeid, 1108
desc
L4virtio::Driver::Virtqueue, 1598
L4virtio::Svr::Virtqueue, 1670
L4virtio::Svr::Virtqueue::Head_desc, 1673
desc_align
L4virtio::Virtqueue, 1679
desc_avail
L4virtio::Svr::Virtqueue, 1670
desc_size
L4virtio::Virtqueue, 1679
descs
l4_vhw_descriptor, 1281
detach
L4::Irq, 1072
L4Re::Rm, 1370, 1371
L4Re::Util::Event_buffer_t, 1413
Detach_flags
L4Re::Rm, 1366
Detach_result
L4Re::Rm, 1366
Dev_config
L4virtio::Svr::Dev_config, 1622
dev_handle
L4vbus::Device, 1537
device
L4vbus::Device, 1537
device_by_hid
L4vbus::Device, 1539
device_error
L4virtio::Svr::Device_t, 1644
di
l4_vcpu_regs_t, 1275
dir
cxx::Bits::Bst, 782, 783
Direction
L4Re::Dma_space, 1310
Direction_e
cxx::Bits::Direction, 794
disable
L4virtio::Virtqueue, 1680
disable_notify
L4virtio::Svr::Virtqueue, 1671
disassociate
L4Re::Dma_space, 1311
dispatch
L4::Basic_registry, 884
L4::Server_object, 1158
dispatch_meta_request
L4::Server_object_t, 1161
dma_space.h
l4re_dma_space_direction, 1913
l4re_dma_space_dma_addr_t, 1913
l4re_dma_space_space_attribs, 1914
done
L4virtio::Svr::Data_buffer, 1619
down
L4::Semaphore, 1152
drive_cylinders
l4util_mb_drive_t, 1526
drive_mode
l4util_mb_drive_t, 1526
drive_number
l4util_mb_drive_t, 1527
driver
L4virtio::Svr::Dev_status, 1638
driver_bfm_t
L4virtio::Svr::Dev_status, 1636
Driver_mem_region_t
L4virtio::Svr::Driver_mem_region_t, 1656
driver_ok
L4virtio::Svr::Dev_status, 1639
driver_ok_bfm_t
L4virtio::Svr::Dev_status, 1637
drv_base
L4virtio::Svr::Driver_mem_region_t, 1657
ds
L4virtio::Svr::Dev_config, 1624
L4virtio::Svr::Driver_mem_region_t, 1657
ds_offset
L4virtio::Svr::Driver_mem_region_t, 1657
dump
L4Re::Video::Color_component, 1482
L4Re::Video::Pixel_info, 1497
L4virtio::Virtqueue, 1680

dx
l4_vcpu_regs_t, 1276

e_phnum
Elf32_Ehdr, 868
Elf64_Ehdr, 874
e_shnum
Elf32_Ehdr, 868
Elf64_Ehdr, 875
EDID parsing functionality, 160
libedid_check_header, 161
libedid_checksum, 161
Libedid_consts, 160
libedid_dump, 161
libedid_dump_standard_timings, 162
libedid_num_ext_blocks, 162
libedid_pnp_id, 162
libedid_prefered_resolution, 163
libedid_revision, 163
libedid_version, 163
EI_CLASS
ELF binary format, 181
EI_DATA
ELF binary format, 182
EI_OSABI
ELF binary format, 182
EI_PAD
ELF binary format, 183
EI_VERSION
ELF binary format, 183
ELF binary format, 165
DF_1_CONFALT, 177
DF_1_DIRECT, 177
DF_1_DISPRELDNE, 178
DF_1_DISPRELPND, 178
DF_1_ENDFILTEE, 178
DF_1_GLOBAL, 178
DF_1_GROUP, 178
DF_1_INTERPOSE, 179
DF_1_LOADFLTR, 179
DF_1_NODEFLIB, 179
DF_1_NODELETE, 179
DF_1_NODUMP, 179
DF_1_NOOPEN, 180
DF_1_NOW, 180
DF_1_ORIGIN, 180
DF_P1_GROUPPERM, 180
DF_P1_LAZYLOAD, 180
DT_HIPROC, 181
DT_LOPROC, 181
DT_NULL, 181
EI_CLASS, 181
EI_DATA, 182
EI_OSABI, 182
EI_PAD, 183
EI_VERSION, 183
ELFCLASSNONE, 184
ELFDATA2LSB, 184
ELFDATA2MSB, 185
ELFDATANONE, 185
ELFOSABI_AIX, 186
ELFOSABI_FREEBSD, 186
ELFOSABI_HPUX, 186
ELFOSABI_IRIX, 186
ELFOSABI_LINUX, 187
ELFOSABI_MODESTO, 187
ELFOSABI_NETBSD, 187
ELFOSABI_OPENBSD, 187
ELFOSABI_SOLARIS, 187
ELFOSABI_SYSV, 188
ELFOSABI_TRU64, 188
EM_ARC, 188
NT_VERSION, 188
PT_GNU_EH_FRAME, 189
PT_GNU_RELRO, 189
PT_GNU_STACK, 189
PT_HIOS, 189
PT_HIPROC, 189
PT_L4_AUX, 190
PT_L4_KIP, 190
PT_L4_STACK, 190
PT_LOOS, 190
PT_LOPROC, 190
SHF_GROUP, 191
SHF_MASKOS, 191
SHF_TLS, 191
SHT_NUM, 191
ELFCLASSNONE
ELF binary format, 184
ELFDATA2LSB
ELF binary format, 184
ELFDATA2MSB
ELF binary format, 185
ELFDATANONE
ELF binary format, 185
ELFOSABI_AIX
ELF binary format, 186
ELFOSABI_FREEBSD
ELF binary format, 186
ELFOSABI_HPUX
ELF binary format, 186
ELFOSABI_IRIX
ELF binary format, 186
ELFOSABI_LINUX
ELF binary format, 187
ELFOSABI_MODESTO
ELF binary format, 187
ELFOSABI_NETBSD
ELF binary format, 187
ELFOSABI_OPENBSD
ELF binary format, 187
ELFOSABI_SOLARIS
ELF binary format, 187
ELFOSABI_SYSV
ELF binary format, 188
ELFOSABI_TRU64
ELF binary format, 188

EM_ARC
ELF binary format, 188
Elf32_Dyn, 866
d_val, 866
Elf32_Ehdr, 867
e_phnum, 868
e_shnum, 868
Elf32_Phdr, 869
Elf32_Shdr, 870
Elf32_Sym, 871
Elf64_Dyn, 872
d_val, 873
Elf64_Ehdr, 873
e_phnum, 874
e_shnum, 875
Elf64_Phdr, 875
Elf64_Shdr, 876
Elf64_Sym, 877
empty
L4virtio::Svr::Driver_mem_region_t, 1657
enable_notify
L4virtio::Svr::Virtqueue, 1671
end
cxx::Bits::Base_avl_set, 768
cxx::Bits::Bst, 784
L4::Kip::Mem_desc, 1089
enqueue_descriptor
L4virtio::Driver::Virtqueue, 1599
entry_ip
L4vcpu::Vcpu, 1587
entry_sp
L4vcpu::Vcpu, 1588
env
L4Re::Env, 1316
env.h
l4re_env_t, 1967
erase
cxx::Bits::Base_avl_set, 769
cxx::H_list, 801
err_no
L4::Runtime_error, 1143
Error
L4virtio::Svr::Bad_descriptor, 1606
Error codes, 199
l4_error_code_t, 199
Error Handling, 192
l4_error, 193
l4_ipc_error, 194
l4_ipc_error_code, 196
l4_ipc_is_rcv_error, 197
l4_ipc_is_snd_error, 197
l4_ipc_tcr_error_t, 193
Event API, 201
Event interface, 202
l4re_event_get_axis_info, 202
l4re_event_get_buffer, 203
l4re_event_get_num_streams, 203
l4re_event_get_stream_info, 204
l4re_event_get_stream_info_for_id, 204
Event_buffer_t
L4Re::Event_buffer_t, 1331
ex_regs
L4::Thread, 1181, 1182
exc_handler
L4::Thread::Attr, 1190
Exception registers, 206
l4_utcb_exc, 206
l4_utcb_exc_is_ex_regs_exception, 206
l4_utcb_exc_is_pf, 207
l4_utcb_exc_pc, 207
l4_utcb_exc_pc_set, 208
execute
L4Re::Ned::Cmd_control, 1359
expired
L4::Ipc_svr::Timeout, 1060
ext_alloc
L4vcpu::Vcpu, 1588
Extended vCPU support, 209
l4vcpu_ext_alloc, 209
extra_str
L4::Runtime_error, 1143

faccessat
L4Re::Vfs::Directory, 1452
Factory, 211
l4_factory_create_factory, 212
l4_factory_create_factory_u, 213
l4_factory_create_gate, 214
l4_factory_create_gate_u, 215
l4_factory_create_irq, 216
l4_factory_create_irq_u, 217
l4_factory_create_task, 218
l4_factory_create_task_u, 219
l4_factory_create_thread, 220
l4_factory_create_thread_u, 221
l4_factory_create_vm, 222
l4_factory_create_vm_u, 223
factory
L4Re::Env, 1316, 1317
failed
L4virtio::Svr::Dev_status, 1639, 1640
failed_bfm_t
L4virtio::Svr::Dev_status, 1637
fchmod
L4Re::Vfs::Generic_file, 1466
fd
l4_vhw_entry, 1283
fdatasync
L4Re::Vfs::Regular_file, 1475
features
l4_icu_info_t, 1253
features_ok
L4virtio::Svr::Dev_status, 1640
features_ok_bfm_t
L4virtio::Svr::Dev_status, 1637
Fiasco extensions, 224
fiasco_gdt_get_entry_offset, 226

fiasco_gdt_set, 226
fiasco_ldt_set, 227
fiasco_tbuf_get_status, 227
fiasco_tbuf_get_status_phys, 227
fiasco_tbuf_log, 227
fiasco_tbuf_log_3val, 228
fiasco_tbuf_log_binary, 229
Fiasco real time scheduling extensions, 230
Fiasco-UX Virtual devices, 231
l4_vhw_entry_type, 231
fiasco_amd64_segment_info
amd64/l4/sys/segment.h, 1757
fiasco_amd64_set_fs
amd64/l4/sys/segment.h, 1758
amd64/l4f/l4/sys/segment.h, 1753
fiasco_amd64_set_segment_base
amd64/l4/sys/segment.h, 1758
amd64/l4f/l4/sys/segment.h, 1754
fiasco_gdt_get_entry_offset
Fiasco extensions, 226
fiasco_gdt_set
Fiasco extensions, 226
fiasco_ldt_set
Fiasco extensions, 227
fiasco_tbuf_get_status
Fiasco extensions, 227
fiasco_tbuf_get_status_phys
Fiasco extensions, 227
fiasco_tbuf_log
Fiasco extensions, 227
fiasco_tbuf_log_3val
Fiasco extensions, 228
fiasco_tbuf_log_binary
Fiasco extensions, 229
finalize_request
L4virtio::Svr::Block_dev, 1611
find
cxx::Bits::Bst, 785
cxx::String, 857
L4::Basic_registry, 884
L4Re::Rm, 1372
L4virtio::Svr::Driver_mem_list_t, 1649
find_next_used
L4virtio::Driver::Virtqueue, 1599
find_node
cxx::Bits::Base_avl_set, 769
cxx::Bits::Bst, 786
first
L4::Kip::Mem_desc, 1090
first_free_cap
L4Re::Env, 1317
first_free_utcb
L4Re::Env, 1318
Flags
L4::Types::Flags, 1228
L4Re::Video::Goos, 1487
L4Re::Video::View, 1502
flags
l4_exc_regs_t, 1250
l4_msgtag_t, 1260
L4Re::Dataspace, 1300
l4re_env_cap_entry_t, 1513
L4virtio::Svr::Driver_mem_region_t, 1658
Flex pages, 233
L4_cap_fpage_rights, 235
l4_fpage, 239
l4_fpage_all, 239
l4_fpage_cacheability_opt_t, 237
l4_fpage_consts, 237
l4_fpage_contains, 240
l4_fpage_invalid, 240
l4_fpage_ioport, 241
l4_fpage_max_order, 241
l4_fpage_memaddr, 242
l4_fpage_obj, 243
l4_fpage_page, 244
L4_fpage_rights, 238
l4_fpage_rights, 244
l4_fpage_set_rights, 245
l4_fpage_size, 245
l4_fpage_type, 246
l4_iofpage, 246
l4_is_fpage_writable, 247
l4_obj_fpage, 247
L4_obj_fpage_ctl, 238
foreach_available_event
L4Re::Util::Event_buffer_consumer_t, 1409
fpage
L4::Cap_base, 898
free
cxx::Base_slab_static, 727
cxx::List_alloc, 817
cxx::Slab, 848
L4Re::Cap_alloc, 1292
L4Re::Mem_alloc, 1346
L4Re::Util::Counting_cap_alloc, 1393
free_area
L4Re::Rm, 1373
free_capability
L4Re::Util::Names::Name_space, 1422
free_descriptor
L4virtio::Driver::Virtqueue, 1599
free_dynamic_entry
L4Re::Util::Names::Name_space, 1423
free_epiface
L4Re::Util::Names::Name_space, 1423
free_fd
L4Re::Vfs::Fs, 1462
free_objects
cxx::Base_slab, 723
cxx::Base_slab_static, 728
from_ci
L4::Ipc::Cap, 970
from_dec
cxx::String, 858
from_hex

cxx::String, 859
from_raw
L4::Types::Flags, 1229
fstat64
L4Re::Vfs::Be_file, 1446
L4Re::Vfs::Generic_file, 1467
fsync
L4Re::Vfs::Regular_file, 1475
ftruncate64
L4Re::Vfs::Regular_file, 1475
full
L4virtio::Svr::Driver_mem_list_t, 1649
Functions for rendering bitmap data in frame buffers, 249
gfxbitmap_bmap, 250
gfxbitmap_color_pix_t, 249
gfxbitmap_color_t, 250
gfxbitmap_convert_color, 251
gfxbitmap_copy, 251
gfxbitmap_fill, 251
gfxbitmap_set, 252
Functions for rendering bitmap fonts to frame buffers, 253
gfxbitmap_font_data, 254
gfxbitmap_font_get, 254
gfxbitmap_font_height, 255
gfxbitmap_font_init, 255
gfxbitmap_font_text, 255
gfxbitmap_font_text_scale, 256
gfxbitmap_font_width, 256
Functions to manipulate the local IDT, 258

g
L4Re::Video::Pixel_info, 1498
get
cxx::Auto_ptr, 703
cxx::Bitfield, 732
cxx::Ref_ptr, 840
L4::Ipc::Istream, 985, 986
L4Re::Env, 1318
L4Re::Video::Color_component, 1482
L4vbus::Gpio_module, 1547
L4vbus::Gpio_pin, 1555
L4virtio::Ptr, 1603
get_attr
L4::Vcon, 1239
get_avail_idx
L4virtio::Virtqueue, 1681
get_buffer
L4Re::Event, 1329
get_buffer_demand
L4::Server_object_t, 1162
get_cap
L4Re::Env, 1319, 1320
get_cap_alloc
L4Re::Cap_alloc, 1293
get_cmd
L4virtio::Svr::Dev_config, 1625
get_epiface
L4Re::Util::Names::Name_space, 1423
get_fb
L4Re::Util::Video::Goos_svr, 1440
get_file
L4Re::Vfs::Fs, 1462
get_infos
L4::Ipc_gate, 1036
get_lock
L4Re::Vfs::Regular_file, 1476
get_next_item
cxx::List_item, 820
get_object_name
L4::Debugger, 909
get_prev_item
cxx::List_item, 820
get_rcv_cap
L4::Ipc_svr::Server_iface, 1053
L4Re::Util::Br_manager, 1382
get_resource
L4vbus::Device, 1540
get_static_buffer
L4Re::Video::Goos, 1489
get_status_flags
L4Re::Vfs::Generic_file, 1467
get_tail_avail_idx
L4virtio::Virtqueue, 1681
get_unshifted
cxx::Bitfield, 733
get_value
L4::Ipc::Varg, 1026
gfxbitmap_bmap
Functions for rendering bitmap data in frame buffers, 250
gfxbitmap_color_pix_t
Functions for rendering bitmap data in frame buffers, 249
gfxbitmap_color_t
Functions for rendering bitmap data in frame buffers, 250
gfxbitmap_convert_color
Functions for rendering bitmap data in frame buffers, 251
gfxbitmap_copy
Functions for rendering bitmap data in frame buffers, 251
gfxbitmap_fill
Functions for rendering bitmap data in frame buffers, 251
gfxbitmap_font_data
Functions for rendering bitmap fonts to frame buffers, 254
gfxbitmap_font_get
Functions for rendering bitmap fonts to frame buffers, 254
gfxbitmap_font_height
Functions for rendering bitmap fonts to frame buffers, 255
gfxbitmap_font_init

Functions for rendering bitmap fonts to frame buffers, 255
gfxbitmap_font_text
Functions for rendering bitmap fonts to frame buffers, 255
gfxbitmap_font_text_scale
Functions for rendering bitmap fonts to frame buffers, 256
gfxbitmap_font_width
Functions for rendering bitmap fonts to frame buffers, 256
gfxbitmap_offset, 878
gfxbitmap_set
Functions for rendering bitmap data in frame buffers, 252
global_id
L4::Debugger, 909
gran_offset
l4_sched_cpu_set_t, 1262
granularity
l4_sched_cpu_set_t, 1261
guest_features
L4virtio::Svr::Dev_config, 1626

H_list_item
cxx, 645
H_list_item_t
cxx::H_list_item_t, 806
handle_expired_timeouts
L4::Ipc_svr::Timeout_queue, 1062
handle_mem_cmd_write
L4virtio::Svr::Device_t, 1644
has_alpha
L4Re::Video::Pixel_info, 1499
has_more
L4virtio::Svr::Request_processor, 1661
hdr
L4virtio::Svr::Dev_config, 1626

i
L4vcpu::Vcpu, 1589
IA32 Port I/O API, 259
l4util_in16, 259
l4util_in32, 261
l4util_in8, 261
l4util_ins16, 262
l4util_ins32, 262
l4util_ins8, 263
l4util_out16, 263
l4util_out32, 263
l4util_out8, 264
l4util_outs16, 264
l4util_outs32, 265
l4util_outs8, 265
IO interface, 266
l4io_device_types_t, 267
l4io_has_resource, 268
l4io_iomem_flags_t, 267
l4io_lookup_device, 269
l4io_lookup_resource, 269
l4io_release_iomem, 270
l4io_release_ioport, 270
l4io_request_iomem, 270
l4io_request_iomem_region, 271
l4io_request_ioport, 272
l4io_request_resource_iomem, 272
l4io_resource_t, 267
l4io_resource_types_t, 268
l4io_search_iomem_region, 273
IPC-Gate API, 274
l4_ipc_gate_bind_thread, 275
l4_ipc_gate_get_infos, 275
l4_rcv_ep_bind_thread, 276
IRQ handling library, 277
IRQs, 278
l4_irq_attach, 280
l4_irq_attach_u, 280
l4_irq_detach, 281
l4_irq_detach_u, 282
L4_irq_mode, 279
l4_irq_mux_chain, 283
l4_irq_mux_chain_u, 284
l4_irq_receive, 285
l4_irq_receive_u, 286
l4_irq_trigger, 287
l4_irq_trigger_u, 288
l4_irq_unmask, 289
l4_irq_unmask_u, 289
l4_irq_wait, 290
l4_irq_wait_u, 291
id
L4::Kobject_typeid, 1108
id_received
L4::Ipc::Gen_fpage, 972
idle_time
L4::Scheduler, 1146
indirect
L4virtio::Virtqueue::Desc::Flags, 1695
indirect_bfm_t
L4virtio::Virtqueue::Desc::Flags, 1694
info
L4::Icu, 941
L4::Scheduler, 1147
L4Re::Dataspace, 1302
L4Re::Video::Goos, 1490
L4Re::Video::View, 1503
Info_sub_type
L4::Kip::Mem_desc, 1086
init
L4Re::Util::Event_t, 1415
L4virtio::Svr::Driver_mem_list_t, 1650
init_infos
L4Re::Util::Video::Goos_svr, 1440
init_mem_info
L4virtio::Svr::Device_t, 1644
init_poll
L4Re::Util::Event_t, 1416

init_queue
L4virtio::Driver::Virtqueue, 1600
Initial Environment, 293
l4re_env, 294
l4re_env_get_cap, 294
l4re_env_get_cap_e, 295
l4re_env_get_cap_l, 296
l4re_kip, 297
initial_caps
L4Re::Env, 1320
initialize_rings
L4virtio::Driver::Virtqueue, 1601
insert
cxx::Avl_map, 709
cxx::Avl_tree, 717
cxx::Bits::Base_avl_set, 770
cxx::H_list, 802
insert_after
cxx::H_list, 802
insert_before
cxx::H_list, 803
insert_next_item
cxx::List_item, 820
insert_prev_item
cxx::List_item, 821
Integer Types, 298
l4_int16_t, 299
l4_int32_t, 299
l4_int64_t, 300
l4_int8_t, 300
l4_uint16_t, 300
l4_uint32_t, 300
l4_uint64_t, 300
l4_uint8_t, 300
interface
L4::Meta, 1115
Interface for asynchronous ISR handlers with a given I↩
RQ capability., 301
l4irq_request_cap, 301
Interface for asynchronous ISR handlers., 303
l4irq_release, 303
l4irq_request, 304
Interface using direct functionality., 305, 308
l4irq_attach, 308
l4irq_attach_cap, 305
l4irq_attach_cap_ft, 305
l4irq_attach_ft, 309
l4irq_attach_thread, 309
l4irq_attach_thread_cap, 306
l4irq_attach_thread_cap_ft, 306
l4irq_attach_thread_ft, 310
l4irq_detach, 310
l4irq_unmask, 311
l4irq_unmask_and_wait_any, 311
l4irq_wait, 311
l4irq_wait_any, 313
Internal constants, 315
Internal functions, 317
Internal Helpers, 314
internal_loop
L4::Server, 1156
Interrupt controller, 318
l4_icu_bind, 320
l4_icu_bind_u, 321
L4_icu_flags, 320
l4_icu_info, 322
l4_icu_info_t, 319
l4_icu_info_u, 322
l4_icu_mask, 323
l4_icu_mask_u, 324
l4_icu_msi_info, 325
l4_icu_msi_info_u, 326
l4_icu_set_mode, 326
l4_icu_set_mode_u, 327
l4_icu_unbind, 328
l4_icu_unbind_u, 329
l4_icu_unmask, 330
l4_icu_unmask_u, 330
Invalid_capability
L4::Invalid_capability, 949
Invalid_type
L4virtio::Ptr, 1603
io.h
l4io_get_root_device, 1811
l4io_iterate_devices, 1811
l4io_request_all_ioports, 1812
l4io_request_icu, 1812
io_page_fault
L4::Io_pager, 951
ioctl
L4Re::Vfs::Special_file, 1480
Iostream
L4::Ipc::Iostream, 977
ipc_client
L4_RPC_DEF, 2085
ipc_iface
L4_INLINE_RPC_NF_OP, 2089
L4_INLINE_RPC_NF, 2089
L4_INLINE_RPC_OP, 2090
L4_INLINE_RPC, 2088
L4_RPC_NF_OP, 2091
L4_RPC_NF, 2091
L4_RPC_OP, 2092
L4_RPC, 2090
ipc_stream
operator<<, 1858–1860
operator>>, 1861–1865
irq
L4Re::Util::Event_t, 1417
irq_disable_save
L4vcpu::Vcpu, 1589
irq_enable
L4vbus::Pci_dev, 1564
L4vbus::Pci_host_bridge, 1570
L4vcpu::Vcpu, 1590
irq_no

l4_vhw_entry, 1283
irq_restore
L4vcpu::Vcpu, 1591
is_compatible
L4vbus::Device, 1540
is_compound
L4::Ipc::Gen_fpage, 973
is_irq_entry
L4vcpu::Vcpu, 1591
is_nil
L4::Ipc::Varg, 1027
is_of
L4::Ipc::Varg, 1027
is_of_int
L4::Ipc::Varg, 1027
is_online
L4::Scheduler, 1147
is_page_fault_entry
L4vcpu::Vcpu, 1592
is_static
L4Re::Util::Dataspace_svr, 1400
is_valid
L4::Cap_base, 899
L4virtio::Ptr, 1604
is_virtual
L4::Kip::Mem_desc, 1091
is_writable
L4virtio::Svr::Driver_mem_region_t, 1658
Istream
L4::Ipc::Istream, 984
items
cxx::List, 811
iter
cxx::Bits::Basic_list, 776
cxx::H_list, 803

Kernel Debugger, 332
l4_debugger_global_id, 332
l4_debugger_kobj_to_id, 333
l4_debugger_set_object_name, 333
Kernel Interface Page, 335
l4_kernel_info_version_offset, 336
l4_kip_clock, 337
l4_kip_clock_lw, 338
l4_kip_version, 338
l4_kip_version_string, 339
Kernel Interface Page API, 340
l4util_kip_for_each_feature, 340
l4util_kip_kernel_abi_version, 341
l4util_kip_kernel_has_feature, 341
l4util_kip_kernel_is_ux, 341
l4util_memdesc_vm_high, 341
Kernel Objects, 343
kobj_to_id
L4::Debugger, 910
kobject_typeid
L4 kernel object type information, 374
Kumem allocator utility, 345
Kumem utiltities, 346
kumem_alloc, 346
kumem_alloc
Kumem utiltities, 346
kumem_alloc.h
l4re_util_kumem_alloc, 1932

L4, 646
cap_cast, 650, 651
cap_dynamic_cast, 652
cap_reinterpret_cast, 653, 654
throw_ipc_exception, 655, 656
L4 IPC Opcodes, 348
L4_icu_opcode, 348
L4_ipc_gate_ops, 349
L4_platform_ctl_ops, 350
L4_task_ops, 350
L4_thread_ops, 350
L4_vcon_ops, 351
L4 kernel object type information, 374
kobject_typeid, 374
L4 VIRTIO Block Device, 352
L4virtio_block_operations, 352
L4virtio_block_status, 353
L4 VIRTIO Interface, 354
L4 VIRTIO Transport Layer, 355
L4_virtio_cmd, 357
L4_virtio_irq_status, 357
L4_virtio_opcodes, 358
l4virtio_config_queue, 359
l4virtio_config_queue_t, 356
l4virtio_config_queues, 360
l4virtio_device_config, 360
L4virtio_device_ids, 358
L4virtio_device_status, 359
L4virtio_feature_bits, 359
l4virtio_register_ds, 361
l4virtio_register_iface, 362
l4virtio_set_status, 362
L4 Vbus functions, 364
l4vbus_assign_dma_domain, 365
L4vbus_dma_domain_assign_flags, 365
l4vbus_get_device, 366
l4vbus_get_device_by_hid, 367
l4vbus_get_hid, 368
l4vbus_get_next_device, 368
l4vbus_get_resource, 369
l4vbus_is_compatible, 370
l4vbus_release_resource, 370
l4vbus_request_resource, 371
l4vbus_vicu_get_cap, 372
l4/cxx/alloc.h, 2248
l4/cxx/atomic.h, 2262, 2263
l4/cxx/avl_map, 1819, 1820
l4/cxx/avl_set, 1822, 1823
l4/cxx/avl_tree, 1826, 1828
l4/cxx/basic_ostream, 1832, 1833
l4/cxx/basic_vector.h, 1836, 1837
l4/cxx/bits/bst.h, 1837, 1839
l4/cxx/bits/bst_base.h, 1841, 1843

l4/cxx/bits/bst_iter.h, 1844, 1846
l4/cxx/exceptions, 1847, 1849
l4/cxx/iostream, 1852, 1853
l4/cxx/ipc_helper, 1853, 1855
l4/cxx/ipc_server, 2098, 2099
l4/cxx/ipc_stream, 1855, 1866
l4/cxx/l4iostream, 1876, 1877
l4/cxx/l4types.h, 1878, 1879
l4/cxx/main_thread, 1879, 1880
l4/cxx/pair, 1881, 1882
l4/cxx/std_exc_io, 1883, 1884
l4/cxx/string.h, 1884, 1886
l4/cxx/thread, 2217, 2218
l4/irq/irq.h, 1886, 1888
l4/libedid/edid.h, 1900, 1901
l4/libgfxbitmap/bitmap.h, 1902, 1904
l4/libgfxbitmap/font.h, 1905, 1906
l4/libgfxbitmap/support, 1907, 1908
l4/re/c/dataspace.h, 1908, 1910
l4/re/c/debug.h, 1911
l4/re/c/dma_space.h, 1912, 1914
l4/re/c/event.h, 1915, 1917
l4/re/c/log.h, 1919, 1920
l4/re/c/mem_alloc.h, 1921, 1923
l4/re/c/namespace.h, 1924, 1925
l4/re/c/rm.h, 1926, 1927
l4/re/c/util/cap_alloc.h, 1930, 1931
l4/re/c/util/kumem_alloc.h, 1931, 1933
l4/re/c/util/video/goos_fb.h, 1933, 1934
l4/re/c/video/colors.h, 1935, 1937
l4/re/c/video/goos.h, 1937, 1939
l4/re/c/video/view.h, 1940, 1943
l4/re/cap_alloc, 1944, 1945
l4/re/consts, 1951, 1952
l4/re/consts.h, 2082, 2083
l4/re/dataspace, 1953, 1954
l4/re/dataspace-sys.h, 1955, 1956
l4/re/debug, 1956, 1958
l4/re/dma_space, 1958, 1960
l4/re/elf_aux.h, 1961, 1962
l4/re/env, 1963, 1964
l4/re/env.h, 1965, 1967
l4/re/error_helper, 1969, 1971
l4/re/event.h, 1917, 1918
l4/re/impl/dataspace_impl.h, 1975, 1976
l4/re/impl/mem_alloc_impl.h, 1977, 1978
l4/re/impl/namespace_impl.h, 1979, 1980
l4/re/impl/rm_impl.h, 1981, 1982
l4/re/l4aux.h, 1984, 1985
l4/re/log, 1985, 1987
l4/re/log-sys.h, 1987, 1988
l4/re/mem_alloc, 1988, 1990
l4/re/mem_alloc-sys.h, 1990, 1991
l4/re/namespace, 1992, 1993
l4/re/namespace-sys.h, 1994, 1995
l4/re/parent, 1995, 1997
l4/re/parent-sys.h, 1997, 1998
l4/re/protocols.h, 1998, 1999
l4/re/rm, 2000, 2001
l4/re/rm-sys.h, 2005, 2006
l4/re/shared_cap, 2006, 2008
l4/re/unique_cap, 2011, 2013
l4/re/util/bitmap_cap_alloc, 2016, 2017
l4/re/util/cap, 2018, 2020
l4/re/util/cap_alloc, 1947, 1949
l4/re/util/cap_alloc_impl.h, 2020, 2021
l4/re/util/counting_cap_alloc, 2022, 2023
l4/re/util/event, 1972, 1973
l4/re/util/item_alloc, 2025, 2026
l4/re/util/kumem_alloc, 2027, 2028
l4/re/util/region_mapping, 2029, 2030
l4/re/util/shared_cap, 2009, 2010
l4/re/util/unique_cap, 2013, 2015
l4/re/video/goos-sys.h, 2035, 2036
l4/shmc/shmc.h, 2036, 2039
l4/sigma0/sigma0.h, 2041, 2043
l4/sys/__kernel_object_impl.h, 2044, 2045
l4/sys/__ktrace-impl.h, 2046, 2047
l4/sys/__typeinfo.h, 2048, 2051
l4/sys/assert.h, 2251, 2253
l4/sys/cache.h, 2061, 2062
l4/sys/capability, 2068, 2070
l4/sys/compiler.h, 2072, 2073
l4/sys/consts.h, 2075, 2077
l4/sys/cxx/ipc_client, 2083, 2085
l4/sys/cxx/ipc_iface, 2086, 2093
l4/sys/cxx/ipc_types, 2100, 2102
l4/sys/cxx/smart_capability_1x, 2109, 2110
l4/sys/cxx/types, 2113, 2114
l4/sys/debugger, 2115, 2116
l4/sys/debugger.h, 2117, 2121
l4/sys/err.h, 2124, 2126
l4/sys/exception, 2126, 2128
l4/sys/factory, 2128, 2130
l4/sys/factory.h, 2132, 2133
l4/sys/icu, 2138, 2139
l4/sys/icu.h, 2139, 2142
l4/sys/ipc.h, 2145, 2148
    l4_ipc_to_errno, 2147
l4/sys/ipc_gate, 2154, 2156
l4/sys/ipc_gate.h, 2156, 2158
l4/sys/irq, 2159, 2161
l4/sys/irq.h, 1895, 1897
l4/sys/kernel_object.h, 2163, 2164
l4/sys/kip, 2165, 2167
l4/sys/kip.h, 2299, 2300
l4/sys/ktrace.h, 2168, 2170
l4/sys/l4int.h, 2171, 2172
l4/sys/memdesc.h, 2175, 2177
l4/sys/meta, 2179, 2181
l4/sys/pager, 2181, 2183
l4/sys/platform_control, 2183, 2185
l4/sys/platform_control.h, 2185, 2187
l4/sys/rcv_endpoint, 2189, 2190
l4/sys/rcv_endpoint.h, 2191, 2193
l4/sys/scheduler, 2193, 2195

l4/sys/scheduler.h, 2195, 2197
l4/sys/semaphore, 2200, 2202
l4/sys/smart_capability, 2202, 2204
l4/sys/task, 2206, 2208
l4/sys/task.h, 2209, 2210
l4/sys/thread, 2214, 2215
l4/sys/thread.h, 2329, 2331
l4/sys/typeinfo_svr, 2219, 2221
l4/sys/types.h, 1813, 1816
    l4_capability_next, 1816
l4/sys/utcb.h, 2221, 2223
l4/sys/vcon, 2234, 2236
l4/sys/vcon.h, 2236, 2239
l4/sys/vhw.h, 2242, 2244
l4/sys/vm, 2245, 2246
l4/util/alloc.h, 2246, 2247
l4/util/assert.h, 2249, 2250
l4/util/atomic.h, 2254, 2257
l4/util/backtrace.h, 2264, 2265
l4/util/base64.h, 2265, 2267
l4/util/bitops.h, 2267, 2269
l4/util/elf.h, 2272, 2285
l4/util/getopt.h, 2295
l4/util/keymap.h, 2296, 2297
l4/util/kip.h, 2297, 2298
l4/util/kprintf.h, 2302, 2303
l4/util/l4_macros.h, 1799, 1800
l4/util/list_alloc.h, 2303, 2306
l4/util/lock.h, 2306, 2307
l4/util/mb_info.h, 2308, 2311
l4/util/parse_cmd.h, 2315, 2316
l4/util/rand.h, 2316, 2317
l4/util/reboot.h, 2318
l4/util/sll.h, 2319
l4/util/splitlog2.h, 2323, 2324
l4/util/stack.h, 2324, 2326
l4/util/thread.h, 2326, 2328
    __L4UTIL_THREAD_FUNC, 2328
l4/vbus/vbus.h, 2340, 2342
l4/vbus/vbus_types.h, 2343, 2344
L4::Alloc_list, 879
L4::Arm_smccc, 880
    call, 880
L4::Base_exception, 881
L4::Basic_registry, 883
    dispatch, 884
    find, 884
L4::Bounds_error, 885
L4::Cap
    Cap, 890, 891
    copy, 892
    move, 892
L4::Cap< T >, 888
L4::Cap_base, 893
    _c, 903
    cap, 896
    Cap_base, 895, 896
    Cap_type, 895
    copy, 897
    fpage, 898
    is_valid, 899
    move, 900
    No_init_type, 895
    snd_base, 901
    validate, 902
L4::Com_error, 904
    Com_error, 906
L4::Debugger, 906
    get_object_name, 909
    global_id, 909
    kobj_to_id, 910
    query_log_name, 910
    query_log_typeid, 911
    set_object_name, 911
    switch_log, 912
L4::Element_already_exists, 913
L4::Element_not_found, 915
L4::Exception, 917
    Rpcs, 917
L4::Exception_tracer, 918
L4::Factory, 919
    create, 923, 924
    create_factory, 925
    create_gate, 926
    create_irq, 927
    create_task, 928
    create_thread, 929
    create_vm, 930
L4::Factory::Lstr, 931
    Lstr, 932
L4::Factory::Nil, 932
L4::Factory::S, 933
    operator l4_msgtag_t, 935
    operator<<, 935–937
    S, 934
L4::IOModifier, 954
L4::Icu, 938
    bind, 940
    info, 941
    mask, 942
    msi_info, 943
    set_mode, 943
    unbind, 944
L4::Icu::Info, 945
L4::Invalid_capability, 946
    cap, 949
    Invalid_capability, 949
L4::Io_pager, 950
    io_page_fault, 951
L4::Iommu, 952
    bind, 954
    unbind, 954
L4::Ipc, 657
    buf_cp_in, 659
    buf_cp_out, 660
    buf_in, 660

make_cap, 661
make_cap_full, 662
make_cap_rw, 663
make_cap_rws, 664
msg_ptr, 664
read, 665
str_cp_in, 665
L4::Ipc::Array< ELEM_TYPE, LEN_TYPE >, 955
L4::Ipc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >, 958
L4::Ipc::Array_ref< ELEM_TYPE, LEN_TYPE >, 959
L4::Ipc::As_value< T >, 961
L4::Ipc::Buf_item, 962
L4::Ipc::Call, 963
L4::Ipc::Call_t< RIGHTS >, 964
L4::Ipc::Call_zero_send_timeout, 966
L4::Ipc::Cap
Cap, 969
from_ci, 970
L4::Ipc::Cap< T >, 967
L4::Ipc::Gen_fpage
cap_received, 972
id_received, 972
is_compound, 973
local_id_received, 973
L4::Ipc::Gen_fpage< T >, 970
L4::Ipc::In_out< T >, 974
L4::Ipc::Iostream, 974
call, 978
Iostream, 977
reply_and_wait, 979
reset, 980
L4::Ipc::Istream, 981
get, 985, 986
Istream, 984
receive, 987
reset, 988
skip, 988
tag, 989
wait, 990
L4::Ipc::Msg, 666
align_to, 668
check_size, 669, 670
msg_add, 670
msg_get, 671
L4::Ipc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >, 991
L4::Ipc::Msg::Cls_buffer, 992
L4::Ipc::Msg::Cls_data, 994
L4::Ipc::Msg::Cls_item, 995
L4::Ipc::Msg::Dir_in, 996
L4::Ipc::Msg::Dir_out, 997
L4::Ipc::Msg::Do_in_data, 998
L4::Ipc::Msg::Do_in_items, 999
L4::Ipc::Msg::Do_out_data, 1000
L4::Ipc::Msg::Do_out_items, 1001
L4::Ipc::Msg::Do_rcv_buffers, 1002
L4::Ipc::Msg::Elem< Array< A, LEN > >, 1005
L4::Ipc::Msg::Elem< Array< A, LEN > &>, 1004
L4::Ipc::Msg::Elem< Array_ref< A, LEN > &>, 1006
L4::Ipc::Msg::Is_valid_rpc_type< T >, 1007
L4::Ipc::Msg::Svr_arg_pack< IPC_TYPE >, 1009
L4::Ipc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >, 1009
L4::Ipc::Msg_ptr
Msg_ptr, 1011
L4::Ipc::Msg_ptr< T >, 1010
L4::Ipc::Opt< T >, 1011
L4::Ipc::Ostream, 1013
put, 1016, 1017
send, 1017
tag, 1018
L4::Ipc::Out< T >, 1019
L4::Ipc::Ret_array< T >, 1020
L4::Ipc::Send_only, 1021
L4::Ipc::Small_buf, 1021
Small_buf, 1022
L4::Ipc::Snd_item, 1023
L4::Ipc::Str_cp_in
Str_cp_in, 1024
L4::Ipc::Str_cp_in< T >, 1023
L4::Ipc::Varg, 1025
data, 1026
get_value, 1026
is_nil, 1027
is_of, 1027
is_of_int, 1027
length, 1027
tag, 1028
type, 1029
value, 1029
L4::Ipc::Varg_list< MAX >, 1030
L4::Ipc::Varg_list_ref, 1031
Varg_list_ref, 1032
L4::Ipc_gate, 1033
get_infos, 1036
L4::Ipc_svr, 672
L4::Ipc_svr::Br_manager_no_buffers, 1036
alloc_buffer_demand, 1039
L4::Ipc_svr::Compound_reply, 1040
L4::Ipc_svr::Default_loop_hooks, 1041
L4::Ipc_svr::Default_setup_wait, 1042
L4::Ipc_svr::Default_timeout, 1043
L4::Ipc_svr::Direct_dispatch< R >, 1044
L4::Ipc_svr::Direct_dispatch< R * >, 1046
L4::Ipc_svr::Exc_dispatch< R, Exc >, 1047
L4::Ipc_svr::Ignore_errors, 1048
L4::Ipc_svr::Server_iface, 1049
add_timeout, 1052
alloc_buffer_demand, 1052
get_rcv_cap, 1053
rcv_cap, 1054, 1055
realloc_rcv_cap, 1056
remove_timeout, 1057
L4::Ipc_svr::Timed_work< HOOKS >, 1058
L4::Ipc_svr::Timeout, 1058

expired, 1060
timeout, 1060
L4::Ipc_svr::Timeout_queue, 1061
add, 1062
handle_expired_timeouts, 1062
next_timeout, 1063
remove, 1063
timeout_expired, 1064
L4::Ipc_svr::Timeout_queue_hooks
add_timeout, 1067
remove_timeout, 1067
L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >, 1065
L4::Irq, 1068
attach, 1071
detach, 1072
receive, 1073
unmask, 1074
wait, 1075
L4::Irq_eoi, 1075
unmask, 1077
L4::Irq_handler_object, 1078
L4::Irq_mux, 1081
chain, 1083
L4::Kip::Mem_desc, 1084
all, 1087, 1088
count, 1088, 1089
end, 1089
first, 1090
Info_sub_type, 1086
is_virtual, 1091
Mem_desc, 1086
Mem_type, 1086
set, 1091
size, 1091
start, 1092
sub_type, 1092
type, 1093
L4::Kobject, 1093
cap, 1094
dec_refcnt, 1096
L4::Kobject_2t
__Iface, 1099
__Iface_list, 1099
__check_protocols__, 1100
c, 1100
Class, 1099
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, 1097
L4::Kobject_3t
__Iface, 1103
__Iface_list, 1103
__check_protocols__, 1104
c, 1104
Class, 1103
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, 1101
L4::Kobject_demand< T >, 1104
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >, 1105
L4::Kobject_typeid
Demand, 1108
demand, 1108
id, 1108
proto_dispatch, 1109
L4::Kobject_typeid< T >, 1107
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, 1121
L4::Platform_control, 1122
cpu_disable, 1125
cpu_enable, 1126
Opcode, 1125
system_shutdown, 1126
system_suspend, 1126
L4::Poll_timeout_kipclock, 1127
Poll_timeout_kipclock, 1128
set, 1128
test, 1129
timed_out, 1130
L4::Proto_t< P >, 1130
L4::Rcv_endpoint, 1132
bind_thread, 1135
L4::Registry_iface, 1136
register_irq_obj, 1137
register_obj, 1138, 1139
unregister_obj, 1140
L4::Runtime_error, 1141
err_no, 1143
extra_str, 1143
Runtime_error, 1143
L4::Scheduler, 1144
idle_time, 1146
info, 1147
is_online, 1147
run_thread, 1148
L4::Semaphore, 1149
down, 1152
up, 1153
L4::Server
internal_loop, 1156
loop, 1156
Server, 1155
L4::Server< LOOP_HOOKS >, 1154
L4::Server_object, 1157
dispatch, 1158
L4::Server_object_t
dispatch_meta_request, 1161
get_buffer_demand, 1162
proto_dispatch, 1162

L4::Server_object_t< IFACE, BASE >, 1159
L4::Server_object_x< Derived, IFACE, BASE >, 1163
L4::Smart_cap
    Smart_cap, 1168
L4::Smart_cap< T, SMART >, 1165
L4::String, 1169
L4::Task, 1169
    add_ku_mem, 1172
    cap_equal, 1173
    cap_has_child, 1173
    cap_valid, 1174
    delete_obj, 1174
    map, 1175
    release_cap, 1175
    unmap, 1176
    unmap_batch, 1176
L4::Thread, 1177
    control, 1181
    ex_regs, 1181, 1182
    modify_senders, 1183
    register_del_irq, 1184
    stats_time, 1184
    switch_to, 1185
    vcpu_control, 1185
    vcpu_control_ext, 1186
    vcpu_resume_commit, 1187
    vcpu_resume_start, 1187
L4::Thread::Attr, 1188
    Attr, 1189
    bind, 1189
    exc_handler, 1190
    pager, 1190, 1191
    ux_host_syscall, 1191
L4::Thread::Modify_senders, 1191
    add, 1192
L4::Triggerable, 1193
    trigger, 1195
L4::Type_info, 1197
L4::Type_info::Demand, 1198
    Demand, 1199
    no_demand, 1200
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, 1201
L4::Type_info::Demand_union_t< D1, D2 >, 1203
L4::Typeid, 673
L4::Typeid::Detail::_Rpcs< OPCODE, O, Default_op< R > >::Rpc< Y >, 1207
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >, 1208
L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y >, 1209
L4::Typeid::Detail::_Rpcs< OPCODE, O, X >, 1206
L4::Typeid::Detail::Rpcs_end, 1210
L4::Typeid::P_dispatch< LIST >, 1211
L4::Typeid::Raw_ipc< CLASS >, 1212
L4::Typeid::Rpc_nocode< OPERATION >, 1212
L4::Typeid::Rpcs< RPCS >, 1215
L4::Typeid::Rpcs_code< OPCODE_TYPE >, 1217
L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS >, 1218
L4::Typeid::Rpcs_sys< ARG >, 1220
L4::Types, 673
L4::Types::Bool< V >, 1222
L4::Types::False, 1223
L4::Types::Flags
    clear, 1229
    Flags, 1228
    from_raw, 1229
    None_type, 1228
L4::Types::Flags< BITS_ENUM, UNDERLYING >, 1225
L4::Types::Same< A, B >, 1230
L4::Types::True, 1232
L4::Unknown_error, 1234
L4::Vcon, 1237
    get_attr, 1239
    read, 1240
    read_with_flags, 1241
    send, 1241
    set_attr, 1242
    write, 1243
L4::Vm, 1244
L4_ALWAYS_INLINE
    Basic Macros, 105
L4_DISABLE_COPY
    capability, 2070
L4_HIDDEN
    Basic Macros, 105
L4_INLINE_RPC_NF_OP
    ipc_iface, 2089
L4_INLINE_RPC_NF
    ipc_iface, 2089
L4_INLINE_RPC_OP
    ipc_iface, 2090
L4_INLINE_RPC
    ipc_iface, 2088
L4_IPC_TIMEOUT_0
    Timeouts, 558
L4_LOG2_PAGESIZE
    Memory related, 429
L4_LOG2_SUPERPAGESIZE
    Memory related, 429
L4_NOTHROW
    Basic Macros, 105
L4_PAGEMASK
    Memory related, 429
L4_RPC_DEF
    ipc_client, 2085
L4_RPC_NF_OP
    ipc_iface, 2091
L4_RPC_NF
    ipc_iface, 2091
L4_RPC_OP
    ipc_iface, 2092
L4_RPC
    ipc_iface, 2090

L4_SUPERPAGEMASK
Memory related, 430
L4_SUPERPAGESIZE
Memory related, 430
l4_addr_consts_t
Memory related, 430
l4_assert
sys/assert.h, 2253
l4_buf_regs_t, 1247
l4_buffer_desc_consts_t
Buffer Registers (BRs), 116
l4_busy_wait_ns
Timestamp Counter, 567
l4_busy_wait_us
Timestamp Counter, 568
l4_cache_clean_data
Cache Consistency, 123
l4_cache_coherent
Cache Consistency, 124
l4_cache_dma_coherent
Cache Consistency, 124
l4_cache_flush_data
Cache Consistency, 125
l4_cache_inv_data
Cache Consistency, 125
l4_calibrate_tsc
Timestamp Counter, 568
l4_cap_consts_t
Capabilities, 128
L4_cap_fpage_rights
Flex pages, 235
l4_capability_equal
Capabilities, 129
l4_capability_next
l4/sys/types.h, 1816
l4_debugger_get_object_name
debugger.h, 2119
l4_debugger_global_id
Kernel Debugger, 332
l4_debugger_kobj_to_id
Kernel Debugger, 333
l4_debugger_query_log_name
debugger.h, 2119
l4_debugger_query_log_typeid
debugger.h, 2120
l4_debugger_set_object_name
Kernel Debugger, 333
l4_debugger_switch_log
debugger.h, 2120
l4_default_caps_t
Capabilities, 128
l4_error
Error Handling, 193
l4_error_code_t
Error codes, 199
l4_exc_regs_t, 1248
flags, 1250
ss, 1250
l4_factory_create_factory
Factory, 212
l4_factory_create_factory_u
Factory, 213
l4_factory_create_gate
Factory, 214
l4_factory_create_gate_u
Factory, 215
l4_factory_create_irq
Factory, 216
l4_factory_create_irq_u
Factory, 217
l4_factory_create_task
Factory, 218
l4_factory_create_task_u
Factory, 219
l4_factory_create_thread
Factory, 220
l4_factory_create_thread_u
Factory, 221
l4_factory_create_vm
Factory, 222
l4_factory_create_vm_u
Factory, 223
l4_fpage
Flex pages, 239
l4_fpage_all
Flex pages, 239
l4_fpage_cacheability_opt_t
Flex pages, 237
l4_fpage_consts
Flex pages, 237
l4_fpage_contains
Flex pages, 240
l4_fpage_invalid
Flex pages, 240
l4_fpage_ioport
Flex pages, 241
l4_fpage_max_order
Flex pages, 241
l4_fpage_memaddr
Flex pages, 242
l4_fpage_obj
Flex pages, 243
l4_fpage_page
Flex pages, 244
L4_fpage_rights
Flex pages, 238
l4_fpage_rights
Flex pages, 244
l4_fpage_set_rights
Flex pages, 245
l4_fpage_size
Flex pages, 245
l4_fpage_t, 1251
l4_fpage_type
Flex pages, 246
l4_get_hz

Timestamp Counter, 569
l4_icu_bind
Interrupt controller, 320
l4_icu_bind_u
Interrupt controller, 321
L4_icu_flags
Interrupt controller, 320
l4_icu_info
Interrupt controller, 322
l4_icu_info_t, 1251
features, 1253
Interrupt controller, 319
l4_icu_info_u
Interrupt controller, 322
l4_icu_mask
Interrupt controller, 323
l4_icu_mask_u
Interrupt controller, 324
l4_icu_msi_info
Interrupt controller, 325
l4_icu_msi_info_t, 1253
msi_data, 1254
l4_icu_msi_info_u
Interrupt controller, 326
L4_icu_opcode
L4 IPC Opcodes, 348
l4_icu_set_mode
Interrupt controller, 326
l4_icu_set_mode_u
Interrupt controller, 327
l4_icu_unbind
Interrupt controller, 328
l4_icu_unbind_u
Interrupt controller, 329
l4_icu_unmask
Interrupt controller, 330
l4_icu_unmask_u
Interrupt controller, 330
l4_int16_t
Integer Types, 299
l4_int32_t
Integer Types, 299
l4_int64_t
Integer Types, 300
l4_int8_t
Integer Types, 300
l4_iofpage
Flex pages, 246
l4_ipc
Object Invocation, 463
l4_ipc_call
Object Invocation, 464
l4_ipc_error
Error Handling, 194
l4_ipc_error_code
Error Handling, 196
l4_ipc_gate_bind_thread
IPC-Gate API, 275
l4_ipc_gate_get_infos
IPC-Gate API, 275
L4_ipc_gate_ops
L4 IPC Opcodes, 349
l4_ipc_is_rcv_error
Error Handling, 197
l4_ipc_is_snd_error
Error Handling, 197
l4_ipc_receive
Object Invocation, 466
l4_ipc_reply_and_wait
Object Invocation, 467
l4_ipc_send
Object Invocation, 468
l4_ipc_send_and_wait
Object Invocation, 469
l4_ipc_sleep
Object Invocation, 471
l4_ipc_tcr_error_t
Error Handling, 193
l4_ipc_timeout
Timeouts, 559
l4_ipc_to_errno
l4/sys/ipc.h, 2147
l4_ipc_wait
Object Invocation, 471
l4_irq_attach
IRQs, 280
l4_irq_attach_u
IRQs, 280
l4_irq_detach
IRQs, 281
l4_irq_detach_u
IRQs, 282
L4_irq_mode
IRQs, 279
l4_irq_mux_chain
IRQs, 283
l4_irq_mux_chain_u
IRQs, 284
l4_irq_receive
IRQs, 285
l4_irq_receive_u
IRQs, 286
l4_irq_trigger
IRQs, 287
l4_irq_trigger_u
IRQs, 288
l4_irq_unmask
IRQs, 289
l4_irq_unmask_u
IRQs, 289
l4_irq_wait
IRQs, 290
l4_irq_wait_u
IRQs, 291
l4_is_fpage_writable
Flex pages, 247

l4_is_invalid_cap
Capabilities, 129
l4_is_valid_cap
Capabilities, 130
l4_kernel_info_get_mem_desc_end
Memory descriptors (C version), 422
l4_kernel_info_get_mem_desc_is_virtual
Memory descriptors (C version), 422
l4_kernel_info_get_mem_desc_start
Memory descriptors (C version), 422
l4_kernel_info_get_mem_desc_subtype
Memory descriptors (C version), 423
l4_kernel_info_get_mem_desc_type
Memory descriptors (C version), 423
l4_kernel_info_get_num_mem_descs
Memory descriptors (C version), 423
l4_kernel_info_mem_desc_t, 1254
Memory descriptors (C version), 421
l4_kernel_info_set_mem_desc
Memory descriptors (C version), 424
l4_kernel_info_t, 1255
l4_kernel_info_version_offset
Kernel Interface Page, 336
l4_kip_clock
Kernel Interface Page, 337
l4_kip_clock_lw
Kernel Interface Page, 338
l4_kip_version
Kernel Interface Page, 338
l4_kip_version_string
Kernel Interface Page, 339
l4_map_control
Message Items, 436
l4_map_obj_control
Message Items, 437
l4_mem_info_sub_type_t
Memory descriptors (C version), 421
L4_mem_op_widths
Memory operations., 425
l4_mem_read
Memory operations., 426
l4_mem_type_t
Memory descriptors (C version), 422
l4_mem_write
Memory operations., 426
l4_msg_item_consts_t
Message Items, 435
l4_msg_regs_t, 1257
l4_msgtag
Message Tag, 442
l4_msgtag_flags
Message Tag, 441, 444
l4_msgtag_has_error
Message Tag, 445
l4_msgtag_is_exception
Message Tag, 446
l4_msgtag_is_io_page_fault
Message Tag, 447
l4_msgtag_is_page_fault
Message Tag, 448
l4_msgtag_is_preemption
Message Tag, 449
l4_msgtag_is_sigma0
Message Tag, 450
l4_msgtag_is_sys_exception
Message Tag, 451
l4_msgtag_items
Message Tag, 452
l4_msgtag_label
Message Tag, 453
l4_msgtag_protocol
Message Tag, 441
l4_msgtag_t, 1258
flags, 1260
Message Tag, 440
l4_msgtag_words
Message Tag, 454
l4_ns_to_tsc
Timestamp Counter, 569
l4_obj_fpage
Flex pages, 247
L4_obj_fpage_ctl
Flex pages, 238
l4_platform_ctl_cpu_disable
Platform Control C API, 475
l4_platform_ctl_cpu_enable
Platform Control C API, 476
L4_platform_ctl_ops
L4 IPC Opcodes, 350
L4_platform_ctl_proto
Message Tag, 442
l4_platform_ctl_system_shutdown
Platform Control C API, 476
l4_platform_ctl_system_suspend
Platform Control C API, 477
l4_rcv_ep_bind_thread
IPC-Gate API, 276
L4_rcv_ep_ops
rcv_endpoint.h, 2192
l4_rcv_timeout
Timeouts, 560
l4_rdpmc
Timestamp Counter, 570
l4_rdpmc_32
Timestamp Counter, 570
l4_rdtsc
Timestamp Counter, 571
l4_rdtsc_32
Timestamp Counter, 571
l4_round_page
Memory related, 431
l4_round_size
Memory related, 432
l4_sched_cpu_set
Scheduler, 499
l4_sched_cpu_set_t, 1260

gran_offset, 1262
granularity, 1261
offset, 1261
l4_sched_param_t, 1263
l4_scheduler_idle_time
Scheduler, 500
l4_scheduler_info
Scheduler, 501
l4_scheduler_is_online
Scheduler, 502
L4_scheduler_ops
Scheduler, 499
l4_scheduler_run_thread
Scheduler, 502
l4_sleep
amd64/l4/util/util.h, 1748
Utility Functions, 576
x86/l4/util/util.h, 1751
l4_snd_fpage_t, 1264
l4_snd_timeout
Timeouts, 560
l4_sndfpage_add
Object Invocation, 473
l4_syscall_flags_t
Object Invocation, 463
l4_task_add_ku_mem
Task, 528
l4_task_cap_equal
Task, 528
l4_task_cap_has_child
Task, 528
l4_task_cap_valid
Task, 529
l4_task_delete_obj
Task, 529
L4_task_ldt_x86_consts
amd64/l4/sys/segment.h, 1757
x86/l4/sys/segment.h, 1763
l4_task_map
Task, 530
L4_task_ops
L4 IPC Opcodes, 350
l4_task_release_cap
Task, 531
l4_task_unmap
Task, 531
l4_task_unmap_batch
Task, 532
l4_thread_arm_set_tpidruro
Thread, 538
l4_thread_control_alien
Thread control, 553
l4_thread_control_bind
Thread control, 553
l4_thread_control_commit
Thread control, 554
l4_thread_control_exc_handler
Thread control, 554
L4_thread_control_flags
Thread, 536
L4_thread_control_mr_indices
Thread, 536
l4_thread_control_pager
Thread control, 555
l4_thread_control_start
Thread control, 555
l4_thread_control_ux_host_syscall
Thread control, 555
l4_thread_ex_regs
Thread, 539
L4_thread_ex_regs_flags
Thread, 538
l4_thread_ex_regs_ret
Thread, 540
l4_thread_ex_regs_ret_u
Thread, 541
l4_thread_ex_regs_u
Thread, 542
l4_thread_longjmp
amd64/l4f/l4/util/setjmp.h, 1774
x86/l4f/l4/util/setjmp.h, 1777
l4_thread_modify_sender_add
Thread, 543
l4_thread_modify_sender_commit
Thread, 543
l4_thread_modify_sender_start
Thread, 544
L4_thread_ops
L4 IPC Opcodes, 350
l4_thread_register_del_irq
Thread, 544
l4_thread_regs_t, 1265
l4_thread_setjmp
amd64/l4f/l4/util/setjmp.h, 1775
x86/l4f/l4/util/setjmp.h, 1778
l4_thread_stats_time
Thread, 545
l4_thread_switch
Thread, 545
l4_thread_vcpu_control
Thread, 545
l4_thread_vcpu_control_ext
Thread, 546
l4_thread_vcpu_control_ext_u
Thread, 547
l4_thread_vcpu_control_u
Thread, 548
l4_thread_vcpu_resume_commit
Thread, 549
l4_thread_vcpu_resume_start
Thread, 550
l4_thread_yield
Thread, 550
l4_timeout
Timeouts, 561
l4_timeout_abs

Timeouts, 561
l4_timeout_abs_validity
Timeouts, 559
l4_timeout_get
Timeouts, 562
l4_timeout_is_absolute
Timeouts, 563
l4_timeout_rel
Timeouts, 563
l4_timeout_rel_get
Timeouts, 564
l4_timeout_s, 1266
Timeouts, 559
l4_timeout_t, 1267
Timeouts, 559
l4_touch_ro
Utility Functions, 577
l4_touch_rw
Utility Functions, 577
l4_tracebuffer_status_t, 1268
cnt_iobmap_tlb_flush, 1269
l4_tracebuffer_status_window_t, 1270
l4_trunc_page
Memory related, 433
l4_trunc_size
Memory related, 433
l4_tsc_init
Timestamp Counter, 572
l4_tsc_to_ns
Timestamp Counter, 573
l4_tsc_to_s_and_ns
Timestamp Counter, 573
l4_tsc_to_us
Timestamp Counter, 574
l4_uint16_t
Integer Types, 300
l4_uint32_t
Integer Types, 300
l4_uint64_t
Integer Types, 300
l4_uint8_t
Integer Types, 300
l4_unmap_flags_t
Task, 527
l4_usleep
Utility Functions, 578
l4_utcb_br
Virtual Registers (UTCBs), 627
L4_utcb_consts_x86
x86 Virtual Registers (UTCB), 641
l4_utcb_exc
Exception registers, 206
l4_utcb_exc_is_ex_regs_exception
Exception registers, 206
l4_utcb_exc_is_pf
Exception registers, 207
l4_utcb_exc_pc
Exception registers, 207
l4_utcb_exc_pc_set
Exception registers, 208
l4_utcb_mr
Virtual Registers (UTCBs), 627
l4_utcb_mr64_idx
Timeouts, 564
l4_utcb_t
Virtual Registers (UTCBs), 626
l4_utcb_tcr
Virtual Registers (UTCBs), 628
l4_vcon_attr_t, 1271
l4_vcon_get_attr
Virtual Console, 614
l4_vcon_get_attr_u
Virtual Console, 614
L4_vcon_i_flags
Virtual Console, 612
L4_vcon_l_flags
Virtual Console, 613
L4_vcon_o_flags
Virtual Console, 613
L4_vcon_ops
L4 IPC Opcodes, 351
l4_vcon_read
Virtual Console, 615
L4_vcon_read_flags
vcon.h, 2239
l4_vcon_read_u
Virtual Console, 616
l4_vcon_read_with_flags
Virtual Console, 617
l4_vcon_send
Virtual Console, 618
l4_vcon_send_u
Virtual Console, 619
l4_vcon_set_attr
Virtual Console, 620
l4_vcon_set_attr_u
Virtual Console, 620
L4_vcon_size_consts
Virtual Console, 613
l4_vcon_write
Virtual Console, 621
l4_vcon_write_u
Virtual Console, 622
l4_vcpu_ipc_regs_t, 1272
l4_vcpu_regs_t, 1273
ax, 1274
bp, 1275
bx, 1275
cx, 1275
di, 1275
dx, 1276
si, 1276
L4_vcpu_state_flags
vCPU API, 631
L4_vcpu_state_offset
vCPU API, 631

l4_vcpu_state_t, 1277
L4_vcpu_sticky_flags
vCPU API, 631
l4_vhw_descriptor, 1279
count, 1281
descs, 1281
magic, 1281
version, 1281
l4_vhw_entry, 1282
fd, 1283
irq_no, 1283
mem_size, 1283
mem_start, 1283
provider_pid, 1284
type, 1284
l4_vhw_entry_type
Fiasco-UX Virtual devices, 231
L4_virtio_cmd
L4 VIRTIO Transport Layer, 357
L4_virtio_irq_status
L4 VIRTIO Transport Layer, 357
L4_virtio_opcodes
L4 VIRTIO Transport Layer, 358
l4_vm_svm_vmcb_control_area, 1285
l4_vm_svm_vmcb_state_save_area, 1285
l4_vm_svm_vmcb_state_save_area_seg, 1287
l4_vm_svm_vmcb_t, 1287
l4_vm_tz_state, 1289
L4_vm_vmx_caps_regs
VM API for VMX, 586
l4_vm_vmx_clear
VM API for VMX, 587
L4_vm_vmx_dfl1_regs
VM API for VMX, 586
l4_vm_vmx_field_len
VM API for VMX, 587
l4_vm_vmx_field_order
VM API for VMX, 588
l4_vm_vmx_get_caps
VM API for VMX, 589
l4_vm_vmx_get_caps_default1
VM API for VMX, 589
l4_vm_vmx_get_cr2_index
VM API for VMX, 590
l4_vm_vmx_ptr_load
VM API for VMX, 590
l4_vm_vmx_read
VM API for VMX, 591
l4_vm_vmx_read_16
VM API for VMX, 592
l4_vm_vmx_read_32
VM API for VMX, 593
l4_vm_vmx_read_64
VM API for VMX, 593
l4_vm_vmx_read_nat
VM API for VMX, 594
l4_vm_vmx_write
VM API for VMX, 594
l4_vm_vmx_write_16
VM API for VMX, 595
l4_vm_vmx_write_32
VM API for VMX, 596
l4_vm_vmx_write_64
VM API for VMX, 597
l4_vm_vmx_write_nat
VM API for VMX, 597
L4RE_ELF_AUX_ELEM_T
L4Re ELF Auxiliary Information, 385
L4RE_ELF_AUX_ELEM
L4Re ELF Auxiliary Information, 385
L4Re, 674
chkcap, 680
chkipc, 680
chksys, 682–684
make_shared_cap, 685
make_shared_del_cap, 686
make_unique_cap, 686
make_unique_del_cap, 687
Shared_cap, 676
shared_cap, 676
Shared_del_cap, 677
shared_del_cap, 677
Unique_cap, 678
unique_cap, 678
Unique_del_cap, 679
unique_del_cap, 679
L4Re C Interface, 376
l4re_inhibitor_acquire, 377
L4Re C++ Interface, 379
L4Re Capability API, 381
cap_alloc, 383
make_auto_cap, 382
make_auto_del_cap, 382
make_ref_cap, 382
make_ref_del_cap, 383
L4Re ELF Auxiliary Information, 384
L4RE_ELF_AUX_ELEM_T, 385
L4RE_ELF_AUX_ELEM, 385
L4Re Protocol identifiers, 387
L4Re Util C Interface, 388
L4Re Util C++ Interface, 389
L4Re::Cap_alloc, 1289
alloc, 1290, 1291
free, 1292
get_cap_alloc, 1293
L4Re::Console, 1294
L4Re::Dataspace, 1296
allocate, 1299
clear, 1299
copy_in, 1300
flags, 1300
info, 1302
map, 1302
Map_flags, 1298
map_region, 1303
phys, 1304

size, 1305
L4Re::Dataspace::Stats, 1305
L4Re::Debug_obj, 1306
debug, 1308
L4Re::Dma_space, 1308
associate, 1311
Attribute, 1310
Attributes, 1309
Direction, 1310
disassociate, 1311
map, 1312
Space_attrib, 1311
unmap, 1312
L4Re::Env, 1313
env, 1316
factory, 1316, 1317
first_free_cap, 1317
first_free_utcb, 1318
get, 1318
get_cap, 1319, 1320
initial_caps, 1320
log, 1321
main_thread, 1321, 1322
mem_alloc, 1322
parent, 1323
rm, 1323, 1324
scheduler, 1324
task, 1325
utcb_area, 1325
L4Re::Event, 1326
get_buffer, 1329
L4Re::Event_buffer_t
Event_buffer_t, 1331
next, 1332
put, 1332
L4Re::Event_buffer_t< PAYLOAD >, 1330
L4Re::Event_buffer_t< PAYLOAD >::Event, 1333
L4Re::Inhibitor, 1334
acquire, 1337
next_lock_info, 1338
release, 1339
L4Re::Log, 1339
print, 1342
printn, 1342
L4Re::Mem_alloc, 1342
alloc, 1346
free, 1346
Mem_alloc_flags, 1345
L4Re::Mmio_space, 1347
Access_width, 1350
mmio_read, 1350
mmio_write, 1351
L4Re::Namespace, 1351
query, 1355, 1356
Query_result_flags, 1354
Register_flags, 1354
register_obj, 1357
unlink, 1357
L4Re::Ned::Cmd_control, 1358
execute, 1359
L4Re::Parent, 1360
signal, 1362
L4Re::Rm, 1362
attach, 1367, 1368
Attach_flags, 1366
detach, 1370, 1371
Detach_flags, 1366
Detach_result, 1366
find, 1372
free_area, 1373
Region_flags, 1367
reserve_area, 1373, 1374
L4Re::Smart_cap_auto< Unmap_flags >, 1375
L4Re::Smart_count_cap< Unmap_flags >, 1376
L4Re::Util, 688
make_shared_cap, 696
make_shared_del_cap, 696
make_unique_cap, 697
make_unique_del_cap, 697
Shared_cap, 690
shared_cap, 691
Shared_del_cap, 692
shared_del_cap, 692
Unique_cap, 693
unique_cap, 694
Unique_del_cap, 694
unique_del_cap, 695
L4Re::Util::Auto_cap< T >, 1377
L4Re::Util::Auto_del_cap< T >, 1378
L4Re::Util::Br_manager, 1379
alloc_buffer_demand, 1382
get_rcv_cap, 1382
realloc_rcv_cap, 1383
set_rcv_cap_flags, 1383
L4Re::Util::Br_manager_hooks, 1384
L4Re::Util::Br_manager_timeout_hooks, 1385
L4Re::Util::Cap_alloc_base, 1388
L4Re::Util::Counter< COUNTER >, 1389
L4Re::Util::Counting_cap_alloc
alloc, 1391, 1392
Counting_cap_alloc, 1391
free, 1393
release, 1394
setup, 1395
take, 1396
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, 1389
L4Re::Util::Dataspace_svr, 1396
allocate, 1398
clear, 1399
copy, 1399
is_static, 1400
map, 1401
map_hook, 1402
page_shift, 1403
phys, 1404

release, 1405
take, 1405
L4Re::Util::Event_buffer_consumer_t
foreach_available_event, 1409
process, 1409
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, 1406
L4Re::Util::Event_buffer_t
attach, 1412
buf, 1413
detach, 1413
L4Re::Util::Event_buffer_t< PAYLOAD >, 1410
L4Re::Util::Event_t
buffer, 1415
init, 1415
init_poll, 1416
irq, 1417
Mode, 1415
L4Re::Util::Event_t< PAYLOAD >, 1414
L4Re::Util::Item_alloc_base, 1418
L4Re::Util::Names::Name, 1419
L4Re::Util::Names::Name_space, 1420
alloc_dynamic_entry, 1422
copy_receive_cap, 1422
free_capability, 1422
free_dynamic_entry, 1423
free_epiface, 1423
get_epiface, 1423
L4Re::Util::Object_registry, 1424
Object_registry, 1426
register_irq_obj, 1427
register_obj, 1428, 1429
unregister_obj, 1429
L4Re::Util::Ref_cap< T >, 1430
L4Re::Util::Ref_del_cap< T >, 1431
L4Re::Util::Registry_server
registry, 1435, 1436
Registry_server, 1435
L4Re::Util::Registry_server< LOOP_HOOKS >, 1432
L4Re::Util::Smart_cap_auto< Unmap_flags >, 1436
L4Re::Util::Smart_count_cap< Unmap_flags >, 1437
L4Re::Util::Vcon_svr< SVR >, 1438
L4Re::Util::Video::Goos_svr, 1439
get_fb, 1440
init_infos, 1440
refresh, 1441
screen_info, 1442
view_info, 1442
L4Re::Vfs, 698
L4Re::Vfs::Be_file, 1443
data_space, 1446
fstat64, 1446
unlock_all_locks, 1446
L4Re::Vfs::Be_file_system, 1447
~Be_file_system, 1449
Be_file_system, 1449
type, 1449
L4Re::Vfs::Directory, 1450
faccessat, 1452
link, 1452
mkdir, 1452
rename, 1453
rmdir, 1453
symlink, 1454
unlink, 1454
L4Re::Vfs::File, 1455
L4Re::Vfs::File_system, 1457
mount, 1458
type, 1459
L4Re::Vfs::Fs, 1459
alloc_fd, 1462
free_fd, 1462
get_file, 1462
mount, 1463
set_fd, 1463
L4Re::Vfs::Generic_file, 1464
fchmod, 1466
fstat64, 1467
get_status_flags, 1467
set_status_flags, 1467
unlock_all_locks, 1468
L4Re::Vfs::Mman, 1469
L4Re::Vfs::Ops, 1470
L4Re::Vfs::Regular_file, 1473
data_space, 1475
fdatasync, 1475
fsync, 1475
ftruncate64, 1475
get_lock, 1476
lseek64, 1476
readv, 1476
set_lock, 1477
writev, 1477
L4Re::Vfs::Special_file, 1478
ioctl, 1480
L4Re::Video::Color_component, 1481
Color_component, 1482
dump, 1482
get, 1482
operator==, 1483
set, 1483
shift, 1483
size, 1484
L4Re::Video::Goos, 1485
create_buffer, 1488
create_view, 1488
delete_buffer, 1488
delete_view, 1489
Flags, 1487
get_static_buffer, 1489
info, 1490
view, 1490
L4Re::Video::Goos::Info, 1491
auto_refresh, 1492
L4Re::Video::Pixel_info, 1493
a, 1495

b, 1496
bits_per_pixel, 1496
bytes_per_pixel, 1497
dump, 1497
g, 1498
has_alpha, 1499
operator==, 1499
Pixel_info, 1494, 1495
r, 1500
L4Re::Video::View, 1500
Flags, 1502
info, 1503
refresh, 1503
set_info, 1504
set_viewport, 1504
stack, 1505
V_flags, 1502
L4Re::Video::View::Info, 1505
L4UTIL_MB_MEMORY
mb_info.h, 2310
l4io_device_types_t
IO interface, 267
l4io_get_root_device
io.h, 1811
l4io_has_resource
IO interface, 268
l4io_iomem_flags_t
IO interface, 267
l4io_iterate_devices
io.h, 1811
l4io_lookup_device
IO interface, 269
l4io_lookup_resource
IO interface, 269
l4io_release_iomem
IO interface, 270
l4io_release_ioport
IO interface, 270
l4io_request_all_ioports
io.h, 1812
l4io_request_icu
io.h, 1812
l4io_request_iomem
IO interface, 270
l4io_request_iomem_region
IO interface, 271
l4io_request_ioport
IO interface, 272
l4io_request_resource_iomem
IO interface, 272
l4io_resource_t
IO interface, 267
l4io_resource_types_t
IO interface, 268
l4io_search_iomem_region
IO interface, 273
l4irq_attach
Interface using direct functionality., 308
l4irq_attach_cap
Interface using direct functionality., 305
l4irq_attach_cap_ft
Interface using direct functionality., 305
l4irq_attach_ft
Interface using direct functionality., 309
l4irq_attach_thread
Interface using direct functionality., 309
l4irq_attach_thread_cap
Interface using direct functionality., 306
l4irq_attach_thread_cap_ft
Interface using direct functionality., 306
l4irq_attach_thread_ft
Interface using direct functionality., 310
l4irq_detach
Interface using direct functionality., 310
l4irq_release
Interface for asynchronous ISR handlers., 303
l4irq_request
Interface for asynchronous ISR handlers., 304
l4irq_request_cap
Interface for asynchronous ISR handlers with a given IRQ capability., 301
l4irq_unmask
Interface using direct functionality., 311
l4irq_unmask_and_wait_any
Interface using direct functionality., 311
l4irq_wait
Interface using direct functionality., 311
l4irq_wait_any
Interface using direct functionality., 313
l4la_alloc
list_alloc.h, 2304
l4la_avail
list_alloc.h, 2304
l4la_dump
list_alloc.h, 2305
l4la_free
list_alloc.h, 2305
l4la_init
list_alloc.h, 2305
l4re_aux_t, 1508
l4re_debug_obj_debug
Debug interface, 158
l4re_dma_space_associate
DMA Space Interface, 150
l4re_dma_space_direction
dma_space.h, 1913
l4re_dma_space_disassociate
DMA Space Interface, 150
l4re_dma_space_dma_addr_t
dma_space.h, 1913
l4re_dma_space_map
DMA Space Interface, 151
l4re_dma_space_space_attribs
dma_space.h, 1914
l4re_dma_space_t
DMA Space Interface, 149

l4re_dma_space_unmap
DMA Space Interface, 152
l4re_ds_allocate
Dataspace interface, 154
l4re_ds_clear
Dataspace interface, 154
l4re_ds_copy_in
Dataspace interface, 155
l4re_ds_flags
Dataspace interface, 155
l4re_ds_info
Dataspace interface, 155
l4re_ds_map_flags
Dataspace interface, 154
l4re_ds_phys
Dataspace interface, 156
l4re_ds_size
Dataspace interface, 156
l4re_ds_stats_t, 1509
l4re_elf_aux_mword_t, 1509
l4re_elf_aux_t, 1510
l4re_elf_aux_vma_t, 1511
l4re_env
Initial Environment, 294
l4re_env_cap_entry_t, 1512
flags, 1513
l4re_env_cap_entry_t, 1512
l4re_env_get_cap
Initial Environment, 294
l4re_env_get_cap_e
Initial Environment, 295
l4re_env_get_cap_l
Initial Environment, 296
l4re_env_t, 1514
env.h, 1967
l4re_event_get_axis_info
Event interface, 202
l4re_event_get_buffer
Event interface, 203
l4re_event_get_num_streams
Event interface, 203
l4re_event_get_stream_info
Event interface, 204
l4re_event_get_stream_info_for_id
Event interface, 204
l4re_event_t, 1515
l4re_inhibitor_acquire
L4Re C Interface, 377
l4re_kip
Initial Environment, 297
l4re_log_print
Log interface, 407
l4re_log_print_srv
Log interface, 408
l4re_log_printn
Log interface, 409
l4re_log_printn_srv
Log interface, 409
l4re_ma_alloc
Memory allocator, 415
l4re_ma_alloc_align
Memory allocator, 416
l4re_ma_alloc_align_srv
Memory allocator, 417
l4re_ma_flags
Memory allocator, 414
l4re_ma_free
Memory allocator, 418
l4re_ma_free_srv
Memory allocator, 419
l4re_ns_query_srv
Namespace interface, 457
l4re_ns_query_to_srv
Namespace interface, 457
l4re_ns_register_flags
Namespace interface, 456
l4re_ns_register_obj_srv
Namespace interface, 459
L4re_protocols
protocols.h, 1999
l4re_rm_attach
Region map interface, 487
l4re_rm_attach_srv
Region map interface, 489
l4re_rm_detach
Region map interface, 489
l4re_rm_detach_ds
Region map interface, 490
l4re_rm_detach_ds_unmap
Region map interface, 491
l4re_rm_detach_srv
Region map interface, 492
l4re_rm_detach_unmap
Region map interface, 493
l4re_rm_find
Region map interface, 494
l4re_rm_find_srv
Region map interface, 494
l4re_rm_flags_t
Region map interface, 487
l4re_rm_free_area
Region map interface, 495
l4re_rm_free_area_srv
Region map interface, 495
l4re_rm_reserve_area
Region map interface, 496
l4re_rm_reserve_area_srv
Region map interface, 496
l4re_rm_show_lists
Region map interface, 497
l4re_util_cap_last
Capability allocator, 131
l4re_util_kumem_alloc
kumem_alloc.h, 1932
l4re_video_color_component_t, 1516
l4re_video_goos_create_buffer

Video API, 603
l4re_video_goos_create_view
Video API, 604
l4re_video_goos_delete_buffer
Video API, 604
l4re_video_goos_delete_view
Video API, 606
l4re_video_goos_get_static_buffer
Video API, 606
l4re_video_goos_get_view
Video API, 606
l4re_video_goos_info
Video API, 607
l4re_video_goos_info_flags_t
Video API, 603
l4re_video_goos_info_t, 1517
l4re_video_goos_refresh
Video API, 607
l4re_video_pixel_info_t, 1518
l4re_video_view_get_info
Video API, 608
l4re_video_view_info_flags_t
Video API, 603
l4re_video_view_info_t, 1519
l4re_video_view_refresh
Video API, 608
l4re_video_view_set_info
Video API, 608
l4re_video_view_set_viewport
Video API, 609
l4re_video_view_stack
Video API, 609
l4re_video_view_t, 1521
Video API, 602
l4shmc_add_chunk
Chunks, 132
l4shmc_add_signal
Signals, 518
l4shmc_area_overhead
Shared Memory Library, 507
l4shmc_area_size
Shared Memory Library, 507
l4shmc_area_size_free
Shared Memory Library, 508
l4shmc_attach
Shared Memory Library, 508
l4shmc_attach_signal
Signals, 519
l4shmc_attach_signal_to
Signals, 519
l4shmc_attach_to
Shared Memory Library, 509
l4shmc_check_magic
Signals, 520
l4shmc_chunk_capacity
Chunks, 133
l4shmc_chunk_consumed
Consumer, 140
l4shmc_chunk_overhead
Shared Memory Library, 509
l4shmc_chunk_ptr
Chunks, 133
l4shmc_chunk_ready
Producer, 478
l4shmc_chunk_ready_sig
Producer, 479
l4shmc_chunk_signal
Chunks, 134
l4shmc_chunk_size
Consumer, 141
l4shmc_chunk_try_to_take
Producer, 479
l4shmc_connect_chunk_signal
Shared Memory Library, 509
l4shmc_create
Shared Memory Library, 511
l4shmc_enable_chunk
Consumer, 141
l4shmc_enable_signal
Consumer, 144
l4shmc_get_chunk
Chunks, 134
l4shmc_get_chunk_to
Chunks, 135
l4shmc_get_signal_to
Signals, 520
l4shmc_is_chunk_clear
Producer, 479
l4shmc_is_chunk_ready
Consumer, 142
l4shmc_iterate_chunk
Chunks, 135
l4shmc_signal_cap
Signals, 521
l4shmc_trigger
Producer, 481
l4shmc_wait_any
Consumer, 145
l4shmc_wait_any_to
Consumer, 145
l4shmc_wait_any_try
Consumer, 145
l4shmc_wait_chunk
Consumer, 142
l4shmc_wait_chunk_to
Consumer, 142
l4shmc_wait_chunk_try
Consumer, 143
l4shmc_wait_signal
Consumer, 146
l4shmc_wait_signal_to
Consumer, 146
l4shmc_wait_signal_try
Consumer, 147
l4sigma0_debug_dump
Sigma0 API, 513

l4sigma0_map_anypage
Sigma0 API, 513
l4sigma0_map_errstr
Sigma0 API, 514
l4sigma0_map_iomem
Sigma0 API, 515
l4sigma0_map_kip
Sigma0 API, 515
l4sigma0_map_mem
Sigma0 API, 516
l4sigma0_map_tbuf
Sigma0 API, 516
l4sigma0_new_client
Sigma0 API, 517
l4sigma0_return_flags_t
Sigma0 API, 513
l4util_add8
Atomic Instructions, 92
l4util_add8_res
Atomic Instructions, 92
l4util_atomic_add
Atomic Instructions, 92
l4util_atomic_inc
Atomic Instructions, 93
l4util_backtrace
backtrace.h, 2265
l4util_bsf
Bit Manipulation, 107
l4util_bsr
Bit Manipulation, 108
l4util_btc
Bit Manipulation, 109
l4util_btr
Bit Manipulation, 109
l4util_bts
Bit Manipulation, 110
l4util_clear_bit
Bit Manipulation, 111
l4util_cmpxchg
Atomic Instructions, 93
l4util_cmpxchg16
Atomic Instructions, 94
l4util_cmpxchg32
Atomic Instructions, 94
l4util_cmpxchg64
Atomic Instructions, 96
l4util_cmpxchg8
Atomic Instructions, 96
l4util_complement_bit
Bit Manipulation, 111
l4util_cpu_capabilities
CPU related functions, 120
l4util_cpu_capabilities_nocheck
CPU related functions, 121
l4util_cpu_has_cpuid
CPU related functions, 122
l4util_find_first_set_bit
Bit Manipulation, 112
l4util_find_first_zero_bit
Bit Manipulation, 112
l4util_idt_desc_t, 1522
l4util_idt_header_t, 1523
l4util_in16
IA32 Port I/O API, 259
l4util_in32
IA32 Port I/O API, 261
l4util_in8
IA32 Port I/O API, 261
l4util_inc8
Atomic Instructions, 97
l4util_inc8_res
Atomic Instructions, 97
l4util_ins16
IA32 Port I/O API, 262
l4util_ins32
IA32 Port I/O API, 262
l4util_ins8
IA32 Port I/O API, 263
l4util_ioport_map
x86/l4f/l4/util/port_io.h, 1768
l4util_irq_acknowledge
amd64/l4/util/irq.h, 1891
x86/l4/util/irq.h, 1894
l4util_kip_for_each_feature
Kernel Interface Page API, 340
l4util_kip_kernel_abi_version
Kernel Interface Page API, 341
l4util_kip_kernel_has_feature
Kernel Interface Page API, 341
l4util_kip_kernel_is_ux
Kernel Interface Page API, 341
l4util_mb_addr_range_t, 1524
l4util_mb_apm_t, 1525
l4util_mb_drive_t, 1525
drive_cylinders, 1526
drive_mode, 1526
drive_number, 1527
l4util_mb_info_t, 1527
l4util_mb_mod_t, 1529
mod_end, 1530
mod_start, 1530
l4util_mb_vbe_ctrl_t, 1530
l4util_mb_vbe_mode_t, 1531
l4util_memdesc_vm_high
Kernel Interface Page API, 341
l4util_micros2l4to
amd64/l4/util/util.h, 1748
Utility Functions, 579
x86/l4/util/util.h, 1751
l4util_next_power2
Bit Manipulation, 112
l4util_out16
IA32 Port I/O API, 263
l4util_out32
IA32 Port I/O API, 263
l4util_out8

IA32 Port I/O API, 264
l4util_outs16
IA32 Port I/O API, 264
l4util_outs32
IA32 Port I/O API, 265
l4util_outs8
IA32 Port I/O API, 265
l4util_rand
Random number support, 482
l4util_set_bit
Bit Manipulation, 113
l4util_splitlog2_hdl
Utility Functions, 579
l4util_splitlog2_size
Utility Functions, 580
l4util_srand
Random number support, 482
l4util_stack_get_sp
amd64/l4/util/stack_impl.h, 1806
arm/l4/util/stack_impl.h, 1805
stack.h, 2325
x86/l4/util/stack_impl.h, 1808
l4util_test_bit
Bit Manipulation, 113
l4util_xchg
Atomic Instructions, 97
l4util_xchg16
Atomic Instructions, 98
l4util_xchg32
Atomic Instructions, 98
l4util_xchg8
Atomic Instructions, 99
L4vbus GPIO functions, 390
l4vbus_gpio_config_get, 392
l4vbus_gpio_config_pad, 392
l4vbus_gpio_config_pull, 393
L4vbus_gpio_generic_func, 391
l4vbus_gpio_get, 394
l4vbus_gpio_multi_config_pad, 394
l4vbus_gpio_multi_get, 395
l4vbus_gpio_multi_set, 396
l4vbus_gpio_multi_setup, 397
L4vbus_gpio_pull_modes, 391
l4vbus_gpio_set, 398
l4vbus_gpio_setup, 398
l4vbus_gpio_to_irq, 399
L4vbus PCI functions, 401
l4vbus_pci_cfg_read, 401
l4vbus_pci_cfg_write, 402
l4vbus_pci_irq_enable, 403
l4vbus_pcidev_cfg_read, 404
l4vbus_pcidev_cfg_write, 404
l4vbus_pcidev_irq_enable, 405
L4vbus::Device, 1534
_bus, 1542
bus_cap, 1536
dev_handle, 1537
device, 1537
device_by_hid, 1539
get_resource, 1540
is_compatible, 1540
next_device, 1541
operator!=, 1542
operator==, 1542
L4vbus::Gpio_module, 1543
config_pad, 1546
get, 1547
pin, 1547
set, 1548
setup, 1548
L4vbus::Gpio_module::Pin_slice, 1549
L4vbus::Gpio_pin, 1550
config_get, 1553
config_pad, 1554
config_pull, 1554
get, 1555
pin, 1555
set, 1556
setup, 1556
to_irq, 1557
L4vbus::Icu, 1558
L4vbus::Pci_dev, 1560
cfg_read, 1563
cfg_write, 1563
irq_enable, 1564
L4vbus::Pci_host_bridge, 1565
cfg_read, 1568
cfg_write, 1569
irq_enable, 1570
L4vbus::Pm< DEC >, 1571
L4vbus::Vbus, 1572
assign_dma_domain, 1575, 1576
release_resource, 1577
request_resource, 1577
root, 1578
l4vbus_assign_dma_domain
L4 Vbus functions, 365
l4vbus_device_t, 1579
L4vbus_dma_domain_assign_flags
L4 Vbus functions, 365
l4vbus_get_device
L4 Vbus functions, 366
l4vbus_get_device_by_hid
L4 Vbus functions, 367
l4vbus_get_hid
L4 Vbus functions, 368
l4vbus_get_next_device
L4 Vbus functions, 368
l4vbus_get_resource
L4 Vbus functions, 369
l4vbus_gpio_config_get
L4vbus GPIO functions, 392
l4vbus_gpio_config_pad
L4vbus GPIO functions, 392
l4vbus_gpio_config_pull
L4vbus GPIO functions, 393

L4vbus_gpio_generic_func
L4vbus GPIO functions, 391
l4vbus_gpio_get
L4vbus GPIO functions, 394
l4vbus_gpio_multi_config_pad
L4vbus GPIO functions, 394
l4vbus_gpio_multi_get
L4vbus GPIO functions, 395
l4vbus_gpio_multi_set
L4vbus GPIO functions, 396
l4vbus_gpio_multi_setup
L4vbus GPIO functions, 397
L4vbus_gpio_pull_modes
L4vbus GPIO functions, 391
l4vbus_gpio_set
L4vbus GPIO functions, 398
l4vbus_gpio_setup
L4vbus GPIO functions, 398
l4vbus_gpio_to_irq
L4vbus GPIO functions, 399
l4vbus_is_compatible
L4 Vbus functions, 370
l4vbus_pci_cfg_read
L4vbus PCI functions, 401
l4vbus_pci_cfg_write
L4vbus PCI functions, 402
l4vbus_pci_irq_enable
L4vbus PCI functions, 403
l4vbus_pcidev_cfg_read
L4vbus PCI functions, 404
l4vbus_pcidev_cfg_write
L4vbus PCI functions, 404
l4vbus_pcidev_irq_enable
L4vbus PCI functions, 405
l4vbus_release_resource
L4 Vbus functions, 370
l4vbus_request_resource
L4 Vbus functions, 371
l4vbus_resource_t, 1580
l4vbus_resource_type_t
vbus_types.h, 2344
l4vbus_vicu_get_cap
L4 Vbus functions, 372
L4vcpu::State, 1581
add, 1582
clear, 1582
set, 1582
State, 1581
L4vcpu::Vcpu, 1583
cast, 1587
entry_ip, 1587
entry_sp, 1588
ext_alloc, 1588
i, 1589
irq_disable_save, 1589
irq_enable, 1590
irq_restore, 1591
is_irq_entry, 1591
is_page_fault_entry, 1592
r, 1592, 1593
saved_state, 1593
state, 1594
task, 1594
wait_for_event, 1595
l4vcpu_ext_alloc
Extended vCPU support, 209
l4vcpu_irq_disable
vCPU Support Library, 634
l4vcpu_irq_disable_save
vCPU Support Library, 634
l4vcpu_irq_enable
vCPU Support Library, 635
l4vcpu_irq_restore
vCPU Support Library, 636
l4vcpu_is_irq_entry
vCPU Support Library, 637
l4vcpu_is_page_fault_entry
vCPU Support Library, 638
l4vcpu_print_state
vCPU Support Library, 638
l4vcpu_wait_for_event
vCPU Support Library, 639
L4virtio, 699
L4virtio::Driver::Virtqueue, 1596
alloc_descriptor, 1598
desc, 1598
enqueue_descriptor, 1599
find_next_used, 1599
free_descriptor, 1599
init_queue, 1600
initialize_rings, 1601
L4virtio::Ptr
get, 1603
Invalid_type, 1603
is_valid, 1604
L4virtio::Ptr< T >, 1601
L4virtio::Svr::Bad_descriptor, 1605
Bad_descriptor, 1606
Error, 1606
message, 1607
L4virtio::Svr::Block_dev
Block_dev, 1610
finalize_request, 1611
process_request, 1611
register_obj, 1612
reset_client, 1613
set_blk_size, 1613
set_size_max, 1613
set_topology, 1614
L4virtio::Svr::Block_dev< Ds_data >, 1607
L4virtio::Svr::Block_request
data_size, 1615
next_block, 1616
L4virtio::Svr::Block_request< Ds_data >, 1614
L4virtio::Svr::Data_buffer, 1617
copy_to, 1618

Data_buffer, 1618
done, 1619
set, 1619
skip, 1620
L4virtio::Svr::Dev_config, 1621
change_queue_config, 1623
Dev_config, 1622
ds, 1624
get_cmd, 1625
guest_features, 1626
hdr, 1626
qconfig, 1627
reset_cmd, 1628
reset_queue, 1628
set_failed, 1629
set_status, 1630
status, 1631
L4virtio::Svr::Dev_features, 1632
ring_event_idx, 1634
ring_event_idx_bfm_t, 1633
ring_indirect_desc, 1634
ring_indirect_desc_bfm_t, 1633
L4virtio::Svr::Dev_status, 1635
acked, 1637, 1638
acked_bfm_t, 1636
driver, 1638
driver_bfm_t, 1636
driver_ok, 1639
driver_ok_bfm_t, 1637
failed, 1639, 1640
failed_bfm_t, 1637
features_ok, 1640
features_ok_bfm_t, 1637
running, 1641
L4virtio::Svr::Device_t
device_error, 1644
handle_mem_cmd_write, 1644
init_mem_info, 1644
reset_queue_config, 1645
setup_queue, 1645
L4virtio::Svr::Device_t< DATA >, 1642
L4virtio::Svr::Driver_mem_list_t
add, 1648
find, 1649
full, 1649
init, 1650
load_desc, 1651, 1652
remove, 1652
L4virtio::Svr::Driver_mem_list_t< DATA >, 1646
L4virtio::Svr::Driver_mem_region_t
contains, 1656
Driver_mem_region_t, 1656
drv_base, 1657
ds, 1657
ds_offset, 1657
empty, 1657
flags, 1658
is_writable, 1658
local, 1658
local_base, 1659
size, 1659
L4virtio::Svr::Driver_mem_region_t< DATA >, 1653
L4virtio::Svr::Request_processor, 1660
current_flags, 1661
has_more, 1661
next, 1662
start, 1664, 1665
L4virtio::Svr::Virtqueue, 1666
consumed, 1669
desc, 1670
desc_avail, 1670
disable_notify, 1671
enable_notify, 1671
next_avail, 1671
L4virtio::Svr::Virtqueue::Head_desc, 1673
desc, 1673
operator Null_ptr_check const *, 1674
valid, 1674
L4virtio::Virtqueue, 1675
avail_align, 1678
avail_size, 1678
desc_align, 1679
desc_size, 1679
disable, 1680
dump, 1680
get_avail_idx, 1681
get_tail_avail_idx, 1681
no_notify_guest, 1682
no_notify_host, 1682, 1683
num, 1683
ready, 1684
setup, 1684
setup_simple, 1685
total_size, 1686
used_align, 1687
used_size, 1687
L4virtio::Virtqueue::Avail, 1688
L4virtio::Virtqueue::Avail::Flags, 1689
no_irq, 1690
no_irq_bfm_t, 1690
L4virtio::Virtqueue::Desc, 1691
L4virtio::Virtqueue::Desc::Flags, 1693
indirect, 1695
indirect_bfm_t, 1694
next, 1696
next_bfm_t, 1695
write, 1696
write_bfm_t, 1695
L4virtio::Virtqueue::Used, 1697
L4virtio::Virtqueue::Used::Flags, 1698
no_notify, 1699, 1700
no_notify_bfm_t, 1699
L4virtio::Virtqueue::Used_elem, 1700
Used_elem, 1701
l4virtio_block_config_t, 1701
blk_size, 1702

l4virtio_block_header_t, 1703
L4virtio_block_operations
L4 VIRTIO Block Device, 352
L4virtio_block_status
L4 VIRTIO Block Device, 353
l4virtio_config_hdr_t, 1704
magic, 1705
status, 1705
l4virtio_config_queue
L4 VIRTIO Transport Layer, 359
l4virtio_config_queue_t, 1706
L4 VIRTIO Transport Layer, 356
l4virtio_config_queues
L4 VIRTIO Transport Layer, 360
l4virtio_device_config
L4 VIRTIO Transport Layer, 360
L4virtio_device_ids
L4 VIRTIO Transport Layer, 358
L4virtio_device_status
L4 VIRTIO Transport Layer, 359
L4virtio_feature_bits
L4 VIRTIO Transport Layer, 359
l4virtio_register_ds
L4 VIRTIO Transport Layer, 361
l4virtio_register_iface
L4 VIRTIO Transport Layer, 362
l4virtio_set_status
L4 VIRTIO Transport Layer, 362
length
L4::Ipc::Varg, 1027
libedid_check_header
EDID parsing functionality, 161
libedid_checksum
EDID parsing functionality, 161
Libedid_consts
EDID parsing functionality, 160
libedid_dump
EDID parsing functionality, 161
libedid_dump_standard_timings
EDID parsing functionality, 162
libedid_num_ext_blocks
EDID parsing functionality, 162
libedid_pnp_id
EDID parsing functionality, 162
libedid_prefered_resolution
EDID parsing functionality, 163
libedid_revision
EDID parsing functionality, 163
libedid_version
EDID parsing functionality, 163
link
L4Re::Vfs::Directory, 1452
List_alloc
cxx::List_alloc, 814
list_alloc.h
l4la_alloc, 2304
l4la_avail, 2304
l4la_dump, 2305
l4la_free, 2305
l4la_init, 2305
load_desc
L4virtio::Svr::Driver_mem_list_t, 1651, 1652
local
L4virtio::Svr::Driver_mem_region_t, 1658
local_base
L4virtio::Svr::Driver_mem_region_t, 1659
local_id_received
L4::Ipc::Gen_fpage, 973
log
L4Re::Env, 1321
Log interface, 407
l4re_log_print, 407
l4re_log_print_srv, 408
l4re_log_printn, 409
l4re_log_printn_srv, 409
Logging interface, 411
loop
L4::Server, 1156
Low-Level Thread Functions, 412
lower_bound_node
cxx::Bits::Base_avl_set, 770
cxx::Bits::Bst, 786
lseek64
L4Re::Vfs::Regular_file, 1476
Lstr
L4::Factory::Lstr, 932

MB_ARD_MEMORY
mb_info.h, 2310
MB_ART_MEMORY
mb_info.h, 2311
Machine Restarting Function, 413
magic
l4_vhw_descriptor, 1281
l4virtio_config_hdr_t, 1705
main_thread
L4Re::Env, 1321, 1322
make_auto_cap
L4Re Capability API, 382
make_auto_del_cap
L4Re Capability API, 382
make_cap
L4::Ipc, 661
make_cap_full
L4::Ipc, 662
make_cap_rw
L4::Ipc, 663
make_cap_rws
L4::Ipc, 664
make_ref_cap
L4Re Capability API, 382
make_ref_del_cap
L4Re Capability API, 383
make_shared_cap
L4Re, 685
L4Re::Util, 696
make_shared_del_cap

L4Re, 686
L4Re::Util, 696
make_unique_cap
L4Re, 686
L4Re::Util, 697
make_unique_del_cap
L4Re, 687
L4Re::Util, 697
map
L4::Task, 1175
L4Re::Dataspace, 1302
L4Re::Dma_space, 1312
L4Re::Util::Dataspace_svr, 1401
Map_flags
L4Re::Dataspace, 1298
map_hook
L4Re::Util::Dataspace_svr, 1402
map_region
L4Re::Dataspace, 1303
mask
L4::Icu, 942
Masks
cxx::Bitfield, 732
max
Small C++ Template Library, 523
mb_info.h
L4UTIL_MB_MEMORY, 2310
MB_ARD_MEMORY, 2310
MB_ART_MEMORY, 2311
mem_alloc
L4Re::Env, 1322
Mem_alloc_flags
L4Re::Mem_alloc, 1345
Mem_desc
L4::Kip::Mem_desc, 1086
mem_size
l4_vhw_entry, 1283
mem_start
l4_vhw_entry, 1283
Mem_type
L4::Kip::Mem_desc, 1086
Memory allocator, 414
l4re_ma_alloc, 415
l4re_ma_alloc_align, 416
l4re_ma_alloc_align_srv, 417
l4re_ma_flags, 414
l4re_ma_free, 418
l4re_ma_free_srv, 419
Memory descriptors (C version), 420
l4_kernel_info_get_mem_desc_end, 422
l4_kernel_info_get_mem_desc_is_virtual, 422
l4_kernel_info_get_mem_desc_start, 422
l4_kernel_info_get_mem_desc_subtype, 423
l4_kernel_info_get_mem_desc_type, 423
l4_kernel_info_get_num_mem_descs, 423
l4_kernel_info_mem_desc_t, 421
l4_kernel_info_set_mem_desc, 424
l4_mem_info_sub_type_t, 421
l4_mem_type_t, 422
Memory operations., 425
L4_mem_op_widths, 425
l4_mem_read, 426
l4_mem_write, 426
Memory related, 428
L4_LOG2_PAGESIZE, 429
L4_LOG2_SUPERPAGESIZE, 429
L4_PAGEMASK, 429
L4_SUPERPAGEMASK, 430
L4_SUPERPAGESIZE, 430
l4_addr_consts_t, 430
l4_round_page, 431
l4_round_size, 432
l4_trunc_page, 433
l4_trunc_size, 433
message
L4virtio::Svr::Bad_descriptor, 1607
Message Items, 435
l4_map_control, 436
l4_map_obj_control, 437
l4_msg_item_consts_t, 435
Message Registers (MRs), 438
Message Tag, 439
l4_msgtag, 442
l4_msgtag_flags, 441, 444
l4_msgtag_has_error, 445
l4_msgtag_is_exception, 446
l4_msgtag_is_io_page_fault, 447
l4_msgtag_is_page_fault, 448
l4_msgtag_is_preemption, 449
l4_msgtag_is_sigma0, 450
l4_msgtag_is_sys_exception, 451
l4_msgtag_items, 452
l4_msgtag_label, 453
l4_msgtag_protocol, 441
l4_msgtag_t, 440
l4_msgtag_words, 454
L4_platform_ctl_proto, 442
min
Small C++ Template Library, 523
mkdir
L4Re::Vfs::Directory, 1452
mmio_read
L4Re::Mmio_space, 1350
mmio_write
L4Re::Mmio_space, 1351
mod_end
l4util_mb_mod_t, 1530
mod_start
l4util_mb_mod_t, 1530
Mode
L4Re::Util::Event_t, 1415
modify_senders
L4::Thread, 1183
mount
L4Re::Vfs::File_system, 1458
L4Re::Vfs::Fs, 1463

move
L4::Cap, 892
L4::Cap_base, 900
msg_add
L4::Ipc::Msg, 670
msg_get
L4::Ipc::Msg, 671
Msg_ptr
L4::Ipc::Msg_ptr, 1011
msg_ptr
L4::Ipc, 664
msi_data
l4_icu_msi_info_t, 1254
msi_info
L4::Icu, 943

NT_VERSION
ELF binary format, 188
Name-space API, 455
Namespace interface, 456
l4re_ns_query_srv, 457
l4re_ns_query_to_srv, 457
l4re_ns_register_flags, 456
l4re_ns_register_obj_srv, 459
next
L4Re::Event_buffer_t, 1332
L4virtio::Svr::Request_processor, 1662
L4virtio::Virtqueue::Desc::Flags, 1696
next_avail
L4virtio::Svr::Virtqueue, 1671
next_bfm_t
L4virtio::Virtqueue::Desc::Flags, 1695
next_block
L4virtio::Svr::Block_request, 1616
next_device
L4vbus::Device, 1541
next_lock_info
L4Re::Inhibitor, 1338
next_timeout
L4::Ipc_svr::Timeout_queue, 1063
no_demand
L4::Type_info::Demand, 1200
No_init_type
L4::Cap_base, 895
no_irq
L4virtio::Virtqueue::Avail::Flags, 1690
no_irq_bfm_t
L4virtio::Virtqueue::Avail::Flags, 1690
no_notify
L4virtio::Virtqueue::Used::Flags, 1699, 1700
no_notify_bfm_t
L4virtio::Virtqueue::Used::Flags, 1699
no_notify_guest
L4virtio::Virtqueue, 1682
no_notify_host
L4virtio::Virtqueue, 1682, 1683
None_type
L4::Types::Flags, 1228
num
L4virtio::Virtqueue, 1683
num_interfaces
L4::Meta, 1115

Object Invocation, 461
l4_ipc, 463
l4_ipc_call, 464
l4_ipc_receive, 466
l4_ipc_reply_and_wait, 467
l4_ipc_send, 468
l4_ipc_send_and_wait, 469
l4_ipc_sleep, 471
l4_ipc_wait, 471
l4_sndfpage_add, 473
l4_syscall_flags_t, 463
Object_registry
L4Re::Util::Object_registry, 1426
offset
l4_sched_cpu_set_t, 1261
Opcode
L4::Platform_control, 1125
operator l4_msgtag_t
L4::Factory::S, 935
operator new
Small C++ Template Library, 524
operator Null_ptr_check const ∗
L4virtio::Svr::Virtqueue::Head_desc, 1674
operator Priv_type ∗
cxx::Auto_ptr, 704
operator!
cxx::Bits::Direction, 794
operator!=
L4vbus::Device, 1542
operator<<
ipc_stream, 1858–1860
L4::Factory::S, 935–937
operator>>
ipc_stream, 1861–1865
operator∗
cxx::Auto_ptr, 704
cxx::Bits::Base_avl_set::Node, 774
operator()
cxx::Pair_first_compare, 834
operator->
cxx::Auto_ptr, 704
cxx::Bits::Base_avl_set::Node, 774
operator=
cxx::Auto_ptr, 704
operator==
L4Re::Video::Color_component, 1483
L4Re::Video::Pixel_info, 1499
L4vbus::Device, 1542
operator[]
cxx::Avl_map, 709, 710
cxx::Bitmap_base, 754
cxx::List, 811

pSLIM_BMAP_START_LSB
bitmap.h, 1903

PT_GNU_EH_FRAME
ELF binary format, 189
PT_GNU_RELRO
ELF binary format, 189
PT_GNU_STACK
ELF binary format, 189
PT_HIOS
ELF binary format, 189
PT_HIPROC
ELF binary format, 189
PT_L4_AUX
ELF binary format, 190
PT_L4_KIP
ELF binary format, 190
PT_L4_STACK
ELF binary format, 190
PT_LOOS
ELF binary format, 190
PT_LOPROC
ELF binary format, 190
page_fault
L4::Pager, 1121
page_shift
L4Re::Util::Dataspace_svr, 1403
pager
L4::Thread::Attr, 1190, 1191
Pair
cxx::Pair, 833
Pair_first_compare
cxx::Pair_first_compare, 834
parent
L4Re::Env, 1323
Parent API, 474
parse_cmdline
Comfortable Command Line Parsing, 136
phys
L4Re::Dataspace, 1304
L4Re::Util::Dataspace_svr, 1404
pin
L4vbus::Gpio_module, 1547
L4vbus::Gpio_pin, 1555
Pixel_info
L4Re::Video::Pixel_info, 1494, 1495
Platform Control C API, 475
l4_platform_ctl_cpu_disable, 475
l4_platform_ctl_cpu_enable, 476
l4_platform_ctl_system_shutdown, 476
l4_platform_ctl_system_suspend, 477
Poll_timeout_kipclock
L4::Poll_timeout_kipclock, 1128
pop_front
cxx::Bits::Smart_ptr_list, 796
cxx::H_list, 803
cxx::S_list, 845
print
L4Re::Log, 1342
printn
L4Re::Log, 1342
process
L4Re::Util::Event_buffer_consumer_t, 1409
process_request
L4virtio::Svr::Block_dev, 1611
Producer, 478, 481
l4shmc_chunk_ready, 478
l4shmc_chunk_ready_sig, 479
l4shmc_chunk_try_to_take, 479
l4shmc_is_chunk_clear, 479
l4shmc_trigger, 481
proto_dispatch
L4::Kobject_typeid, 1109
L4::Server_object_t, 1162
protocols.h
L4re_protocols, 1999
provider_pid
l4_vhw_entry, 1284
ptr
cxx::Ref_ptr, 841
push_back
cxx::List, 811
cxx::List_item, 821
push_front
cxx::List, 812
cxx::List_item, 822
put
L4::Ipc::Ostream, 1016, 1017
L4Re::Event_buffer_t, 1332

qconfig
L4virtio::Svr::Dev_config, 1627
query
L4Re::Namespace, 1355, 1356
query_log_name
L4::Debugger, 910
query_log_typeid
L4::Debugger, 911
Query_result_flags
L4Re::Namespace, 1354

r
L4Re::Video::Pixel_info, 1500
L4vcpu::Vcpu, 1592, 1593
Random number support, 482
l4util_rand, 482
l4util_srand, 482
rbegin
cxx::Bits::Base_avl_set, 770, 771
cxx::Bits::Bst, 787, 788
rcv_cap
L4::Ipc_svr::Server_iface, 1054, 1055
rcv_endpoint.h
L4_rcv_ep_ops, 2192
read
L4::Ipc, 665
L4::Vcon, 1240
read_with_flags
L4::Vcon, 1241
readv

L4Re::Vfs::Regular_file, 1476
ready
L4virtio::Virtqueue, 1684
realloc_rcv_cap
L4::Ipc_svr::Server_iface, 1056
L4Re::Util::Br_manager, 1383
Realtime API, 484
receive
L4::Ipc::Istream, 987
L4::Irq, 1073
Ref
cxx::Bitfield, 730
Ref_ptr
cxx::Ref_ptr, 839, 840
Ref_type
cxx::Auto_ptr, 702
Ref_unshifted
cxx::Bitfield, 731
refresh
L4Re::Util::Video::Goos_svr, 1441
L4Re::Video::View, 1503
Region map API, 485
Region map interface, 486
l4re_rm_attach, 487
l4re_rm_attach_srv, 489
l4re_rm_detach, 489
l4re_rm_detach_ds, 490
l4re_rm_detach_ds_unmap, 491
l4re_rm_detach_srv, 492
l4re_rm_detach_unmap, 493
l4re_rm_find, 494
l4re_rm_find_srv, 494
l4re_rm_flags_t, 487
l4re_rm_free_area, 495
l4re_rm_free_area_srv, 495
l4re_rm_reserve_area, 496
l4re_rm_reserve_area_srv, 496
l4re_rm_show_lists, 497
Region_flags
L4Re::Rm, 1367
register_del_irq
L4::Thread, 1184
Register_flags
L4Re::Namespace, 1354
register_irq_obj
L4::Registry_iface, 1137
L4Re::Util::Object_registry, 1427
register_obj
L4::Registry_iface, 1138, 1139
L4Re::Namespace, 1357
L4Re::Util::Object_registry, 1428, 1429
L4virtio::Svr::Block_dev, 1612
registry
L4Re::Util::Registry_server, 1435, 1436
Registry_server
L4Re::Util::Registry_server, 1435
release
cxx::Auto_ptr, 705
cxx::Ref_ptr, 841
L4Re::Inhibitor, 1339
L4Re::Util::Counting_cap_alloc, 1394
L4Re::Util::Dataspace_svr, 1405
release_cap
L4::Task, 1175
release_resource
L4vbus::Vbus, 1577
remove
cxx::Avl_tree, 717
cxx::Bits::Base_avl_set, 771
cxx::H_list, 804
cxx::List, 812
cxx::List_item, 822
L4::Ipc_svr::Timeout_queue, 1063
L4virtio::Svr::Driver_mem_list_t, 1652
remove_all
cxx::Bits::Bst, 788
remove_me
cxx::List_item, 823
cxx::List_item::Iter, 826
remove_timeout
L4::Ipc_svr::Server_iface, 1057
L4::Ipc_svr::Timeout_queue_hooks, 1067
remove_tree
cxx::Bits::Bst, 789
rename
L4Re::Vfs::Directory, 1453
rend
cxx::Bits::Base_avl_set, 772
cxx::Bits::Bst, 789, 790
replace
cxx::H_list, 804
reply_and_wait
L4::Ipc::Iostream, 979
Reply_mode
Server-Side IPC framework, 505
request_resource
L4vbus::Vbus, 1577
reserve_area
L4Re::Rm, 1373, 1374
reset
L4::Ipc::Iostream, 980
L4::Ipc::Istream, 988
reset_client
L4virtio::Svr::Block_dev, 1613
reset_cmd
L4virtio::Svr::Dev_config, 1628
reset_queue
L4virtio::Svr::Dev_config, 1628
reset_queue_config
L4virtio::Svr::Device_t, 1645
ring_event_idx
L4virtio::Svr::Dev_features, 1634
ring_event_idx_bfm_t
L4virtio::Svr::Dev_features, 1633
ring_indirect_desc
L4virtio::Svr::Dev_features, 1634

ring_indirect_desc_bfm_t
L4virtio::Svr::Dev_features, 1633
rm
L4Re::Env, 1323, 1324
rmdir
L4Re::Vfs::Directory, 1453
root
L4vbus::Vbus, 1578
Rpcs
L4::Exception, 917
run_thread
L4::Scheduler, 1148
running
L4virtio::Svr::Dev_status, 1641
Runtime_error
L4::Runtime_error, 1143

S
L4::Factory::S, 934
SHF_GROUP
ELF binary format, 191
SHF_MASKOS
ELF binary format, 191
SHF_TLS
ELF binary format, 191
SHT_NUM
ELF binary format, 191
saved_state
L4vcpu::Vcpu, 1593
scan_zero
cxx::Bitmap, 746
cxx::Bitmap_base, 755
Scheduler, 498
l4_sched_cpu_set, 499
l4_scheduler_idle_time, 500
l4_scheduler_info, 501
l4_scheduler_is_online, 502
L4_scheduler_ops, 499
l4_scheduler_run_thread, 502
scheduler
L4Re::Env, 1324
screen_info
L4Re::Util::Video::Goos_svr, 1442
send
L4::Ipc::Ostream, 1017
L4::Vcon, 1241
Server
L4::Server, 1155
Server-Side IPC framework, 504
Reply_mode, 505
set
cxx::Bitfield, 733
L4::Kip::Mem_desc, 1091
L4::Poll_timeout_kipclock, 1128
L4Re::Video::Color_component, 1483
L4vbus::Gpio_module, 1548
L4vbus::Gpio_pin, 1556
L4vcpu::State, 1582
L4virtio::Svr::Data_buffer, 1619
set_attr
L4::Vcon, 1242
set_bit
cxx::Bitmap_base, 756
set_blk_size
L4virtio::Svr::Block_dev, 1613
set_dirty
cxx::Bitfield, 734
set_failed
L4virtio::Svr::Dev_config, 1629
set_fd
L4Re::Vfs::Fs, 1463
set_info
L4Re::Video::View, 1504
set_lock
L4Re::Vfs::Regular_file, 1477
set_mode
L4::Icu, 943
set_object_name
L4::Debugger, 911
set_rcv_cap_flags
L4Re::Util::Br_manager, 1383
set_size_max
L4virtio::Svr::Block_dev, 1613
set_status
L4virtio::Svr::Dev_config, 1630
set_status_flags
L4Re::Vfs::Generic_file, 1467
set_topology
L4virtio::Svr::Block_dev, 1614
set_unshifted
cxx::Bitfield, 735
set_unshifted_dirty
cxx::Bitfield, 736
set_viewport
L4Re::Video::View, 1504
setup
L4Re::Util::Counting_cap_alloc, 1395
L4vbus::Gpio_module, 1548
L4vbus::Gpio_pin, 1556
L4virtio::Virtqueue, 1684
setup_queue
L4virtio::Svr::Device_t, 1645
setup_simple
L4virtio::Virtqueue, 1685
Shared Memory Library, 506
l4shmc_area_overhead, 507
l4shmc_area_size, 507
l4shmc_area_size_free, 508
l4shmc_attach, 508
l4shmc_attach_to, 509
l4shmc_chunk_overhead, 509
l4shmc_connect_chunk_signal, 509
l4shmc_create, 511
Shared_cap
L4Re, 676
L4Re::Util, 690
shared_cap

L4Re, 676
L4Re::Util, 691
Shared_del_cap
L4Re, 677
L4Re::Util, 692
shared_del_cap
L4Re, 677
L4Re::Util, 692
shift
L4Re::Video::Color_component, 1483
Shift_type
cxx::Bitfield, 731
si
l4_vcpu_regs_t, 1276
Sigma0 API, 512
l4sigma0_debug_dump, 513
l4sigma0_map_anypage, 513
l4sigma0_map_errstr, 514
l4sigma0_map_iomem, 515
l4sigma0_map_kip, 515
l4sigma0_map_mem, 516
l4sigma0_map_tbuf, 516
l4sigma0_new_client, 517
l4sigma0_return_flags_t, 513
signal
L4Re::Parent, 1362
Signals, 518
l4shmc_add_signal, 518
l4shmc_attach_signal, 519
l4shmc_attach_signal_to, 519
l4shmc_check_magic, 520
l4shmc_get_signal_to, 520
l4shmc_signal_cap, 521
size
cxx::List, 812
L4::Kip::Mem_desc, 1091
L4Re::Dataspace, 1305
L4Re::Video::Color_component, 1484
L4virtio::Svr::Driver_mem_region_t, 1659
skip
L4::Ipc::Istream, 988
L4virtio::Svr::Data_buffer, 1620
Small C++ Template Library, 522
max, 523
min, 523
operator new, 524
Small_buf
L4::Ipc::Small_buf, 1022
Smart_cap
L4::Smart_cap, 1168
snd_base
L4::Cap_base, 901
Space_attrib
L4Re::Dma_space, 1311
ss
l4_exc_regs_t, 1250
stack
L4Re::Video::View, 1505
stack.h
l4util_stack_get_sp, 2325
start
L4::Kip::Mem_desc, 1092
L4virtio::Svr::Request_processor, 1664, 1665
starts_with
cxx::String, 860
State
L4vcpu::State, 1581
state
L4vcpu::Vcpu, 1594
stats_time
L4::Thread, 1184
status
L4virtio::Svr::Dev_config, 1631
l4virtio_config_hdr_t, 1705
Str_cp_in
L4::Ipc::Str_cp_in, 1024
str_cp_in
L4::Ipc, 665
String
cxx::String, 857
sub_type
L4::Kip::Mem_desc, 1092
supports
L4::Meta, 1115
switch_log
L4::Debugger, 912
switch_to
L4::Thread, 1185
symlink
L4Re::Vfs::Directory, 1454
sys/assert.h
l4_assert, 2253
system_shutdown
L4::Platform_control, 1126
system_suspend
L4::Platform_control, 1126

tag
L4::Ipc::Istream, 989
L4::Ipc::Ostream, 1018
L4::Ipc::Varg, 1028
take
L4Re::Util::Counting_cap_alloc, 1396
L4Re::Util::Dataspace_svr, 1405
Task, 526
l4_task_add_ku_mem, 528
l4_task_cap_equal, 528
l4_task_cap_has_child, 528
l4_task_cap_valid, 529
l4_task_delete_obj, 529
l4_task_map, 530
l4_task_release_cap, 531
l4_task_unmap, 531
l4_task_unmap_batch, 532
l4_unmap_flags_t, 527
task
L4Re::Env, 1325

L4vcpu::Vcpu, 1594
test
L4::Poll_timeout_kipclock, 1129
Thread, 534
l4_thread_arm_set_tpidruro, 538
L4_thread_control_flags, 536
L4_thread_control_mr_indices, 536
l4_thread_ex_regs, 539
L4_thread_ex_regs_flags, 538
l4_thread_ex_regs_ret, 540
l4_thread_ex_regs_ret_u, 541
l4_thread_ex_regs_u, 542
l4_thread_modify_sender_add, 543
l4_thread_modify_sender_commit, 543
l4_thread_modify_sender_start, 544
l4_thread_register_del_irq, 544
l4_thread_stats_time, 545
l4_thread_switch, 545
l4_thread_vcpu_control, 545
l4_thread_vcpu_control_ext, 546
l4_thread_vcpu_control_ext_u, 547
l4_thread_vcpu_control_u, 548
l4_thread_vcpu_resume_commit, 549
l4_thread_vcpu_resume_start, 550
l4_thread_yield, 550
Thread control, 552
l4_thread_control_alien, 553
l4_thread_control_bind, 553
l4_thread_control_commit, 554
l4_thread_control_exc_handler, 554
l4_thread_control_pager, 555
l4_thread_control_start, 555
l4_thread_control_ux_host_syscall, 555
Thread Control Registers (TCRs), 551
throw_ipc_exception
L4, 655, 656
timed_out
L4::Poll_timeout_kipclock, 1130
timeout
L4::Ipc_svr::Timeout, 1060
timeout_expired
L4::Ipc_svr::Timeout_queue, 1064
Timeouts, 557
L4_IPC_TIMEOUT_0, 558
l4_ipc_timeout, 559
l4_rcv_timeout, 560
l4_snd_timeout, 560
l4_timeout, 561
l4_timeout_abs, 561
l4_timeout_abs_validity, 559
l4_timeout_get, 562
l4_timeout_is_absolute, 563
l4_timeout_rel, 563
l4_timeout_rel_get, 564
l4_timeout_s, 559
l4_timeout_t, 559
l4_utcb_mr64_idx, 564
Timestamp Counter, 566
l4_busy_wait_ns, 567
l4_busy_wait_us, 568
l4_calibrate_tsc, 568
l4_get_hz, 569
l4_ns_to_tsc, 569
l4_rdpmc, 570
l4_rdpmc_32, 570
l4_rdtsc, 571
l4_rdtsc_32, 571
l4_tsc_init, 572
l4_tsc_to_ns, 573
l4_tsc_to_s_and_ns, 573
l4_tsc_to_us, 574
to_irq
L4vbus::Gpio_pin, 1557
total_objects
cxx::Base_slab, 723
cxx::Base_slab_static, 728
total_size
L4virtio::Virtqueue, 1686
trigger
L4::Triggerable, 1195
type
L4::Ipc::Varg, 1029
L4::Kip::Mem_desc, 1093
l4_vhw_entry, 1284
L4Re::Vfs::Be_file_system, 1449
L4Re::Vfs::File_system, 1459

unbind
L4::Icu, 944
L4::Iommu, 954
Unique_cap
L4Re, 678
L4Re::Util, 693
unique_cap
L4Re, 678
L4Re::Util, 694
Unique_del_cap
L4Re, 679
L4Re::Util, 694
unique_del_cap
L4Re, 679
L4Re::Util, 695
unlink
L4Re::Namespace, 1357
L4Re::Vfs::Directory, 1454
unlock_all_locks
L4Re::Vfs::Be_file, 1446
L4Re::Vfs::Generic_file, 1468
unmap
L4::Task, 1176
L4Re::Dma_space, 1312
unmap_batch
L4::Task, 1176
unmask
L4::Irq, 1074
L4::Irq_eoi, 1077
unregister_obj

L4::Registry_iface, 1140
L4Re::Util::Object_registry, 1429
up
L4::Semaphore, 1153
used_align
L4virtio::Virtqueue, 1687
Used_elem
L4virtio::Virtqueue::Used_elem, 1701
used_size
L4virtio::Virtqueue, 1687
utcb_area
L4Re::Env, 1325
Utility Functions, 575
l4_sleep, 576
l4_touch_ro, 577
l4_touch_rw, 577
l4_usleep, 578
l4util_micros2l4to, 579
l4util_splitlog2_hdl, 579
l4util_splitlog2_size, 580
ux_host_syscall
L4::Thread::Attr, 1191

V_flags
L4Re::Video::View, 1502
vCPU API, 630
L4_vcpu_state_flags, 631
L4_vcpu_state_offset, 631
L4_vcpu_sticky_flags, 631
vCPU Support Library, 633
l4vcpu_irq_disable, 634
l4vcpu_irq_disable_save, 634
l4vcpu_irq_enable, 635
l4vcpu_irq_restore, 636
l4vcpu_is_irq_entry, 637
l4vcpu_is_page_fault_entry, 638
l4vcpu_print_state, 638
l4vcpu_wait_for_event, 639
VM API for SVM, 582
VM API for TZ, 583
VM API for VMX, 584
L4_vm_vmx_caps_regs, 586
l4_vm_vmx_clear, 587
L4_vm_vmx_dfl1_regs, 586
l4_vm_vmx_field_len, 587
l4_vm_vmx_field_order, 588
l4_vm_vmx_get_caps, 589
l4_vm_vmx_get_caps_default1, 589
l4_vm_vmx_get_cr2_index, 590
l4_vm_vmx_ptr_load, 590
l4_vm_vmx_read, 591
l4_vm_vmx_read_16, 592
l4_vm_vmx_read_32, 593
l4_vm_vmx_read_64, 593
l4_vm_vmx_read_nat, 594
l4_vm_vmx_write, 594
l4_vm_vmx_write_16, 595
l4_vm_vmx_write_32, 596
l4_vm_vmx_write_64, 597
l4_vm_vmx_write_nat, 597
Val
cxx::Bitfield, 731
val
cxx::Bitfield, 737
val_dirty
cxx::Bitfield, 737
Val_unshifted
cxx::Bitfield, 731
val_unshifted
cxx::Bitfield, 738
valid
cxx::Bits::Base_avl_set::Node, 774
L4virtio::Svr::Virtqueue::Head_desc, 1674
validate
L4::Cap_base, 902
value
L4::Ipc::Varg, 1029
Varg_list_ref
L4::Ipc::Varg_list_ref, 1032
Vbus API, 600
vbus_types.h
l4vbus_resource_type_t, 2344
vcon.h
L4_vcon_read_flags, 2239
vcpu_control
L4::Thread, 1185
vcpu_control_ext
L4::Thread, 1186
vcpu_resume_commit
L4::Thread, 1187
vcpu_resume_start
L4::Thread, 1187
version
l4_vhw_descriptor, 1281
Video API, 601
l4re_video_goos_create_buffer, 603
l4re_video_goos_create_view, 604
l4re_video_goos_delete_buffer, 604
l4re_video_goos_delete_view, 606
l4re_video_goos_get_static_buffer, 606
l4re_video_goos_get_view, 606
l4re_video_goos_info, 607
l4re_video_goos_info_flags_t, 603
l4re_video_goos_refresh, 607
l4re_video_view_get_info, 608
l4re_video_view_info_flags_t, 603
l4re_video_view_refresh, 608
l4re_video_view_set_info, 608
l4re_video_view_set_viewport, 609
l4re_video_view_stack, 609
l4re_video_view_t, 602
view
L4Re::Video::Goos, 1490
view_info
L4Re::Util::Video::Goos_svr, 1442
Virtual Console, 611
l4_vcon_get_attr, 614

l4_vcon_get_attr_u, 614
L4_vcon_i_flags, 612
L4_vcon_l_flags, 613
L4_vcon_o_flags, 613
l4_vcon_read, 615
l4_vcon_read_u, 616
l4_vcon_read_with_flags, 617
l4_vcon_send, 618
l4_vcon_send_u, 619
l4_vcon_set_attr, 620
l4_vcon_set_attr_u, 620
L4_vcon_size_consts, 613
l4_vcon_write, 621
l4_vcon_write_u, 622
Virtual Machines, 624
Virtual Registers (UTCBs), 625
l4_utcb_br, 627
l4_utcb_mr, 627
l4_utcb_t, 626
l4_utcb_tcr, 628

wait
L4::Ipc::Istream, 990
L4::Irq, 1075
wait_for_event
L4vcpu::Vcpu, 1595
word_index
cxx::Bitmap_base, 757
words
cxx::Bitmap_base, 757
write
L4::Vcon, 1243
L4virtio::Virtqueue::Desc::Flags, 1696
write_bfm_t
L4virtio::Virtqueue::Desc::Flags, 1695
writev
L4Re::Vfs::Regular_file, 1477

x86 Virtual Registers (UTCB), 641
L4_utcb_consts_x86, 641
x86/l4/sys/cache.h, 2067, 2068
x86/l4/sys/consts.h, 2081, 2082
x86/l4/sys/l4int.h, 2174, 2175
x86/l4/sys/linkage.h, 1780, 1781
x86/l4/sys/segment.h, 1762, 1763
L4_task_ldt_x86_consts, 1763
x86/l4/sys/utcb.h, 2231, 2233
x86/l4/util/apic.h, 1714, 1715
x86/l4/util/bitops_arch.h, 1789, 1790
x86/l4/util/cpu.h, 1796, 1797
x86/l4/util/idt.h, 1722, 1723
x86/l4/util/irq.h, 1893, 1894
l4util_irq_acknowledge, 1894
x86/l4/util/l4_macros.h, 1800, 1801
x86/l4/util/mbi_argv.h, 1803, 1804
x86/l4/util/perform.h, 1729, 1730
x86/l4/util/port_io.h, 1769, 1771
x86/l4/util/rdtsc.h, 1740, 1741
x86/l4/util/spin.h, 1746
x86/l4/util/stack_impl.h, 1807, 1808
l4util_stack_get_sp, 1808
x86/l4/util/util.h, 1750, 1752
l4_sleep, 1751
l4util_micros2l4to, 1751
x86/l4f/l4/sys/ipc.h, 2153, 2154
x86/l4f/l4/sys/segment.h, 1760, 1761
x86/l4f/l4/util/port_io.h, 1766, 1769
l4util_ioport_map, 1768
x86/l4f/l4/util/setjmp.h, 1776, 1778
l4_thread_longjmp, 1777
l4_thread_setjmp, 1778