

Contents

1	Fias	co.OC &	L4 Runtime Environment (L4Re)	1
	1.1	Preface		1
	1.2	Genera	System Structure	1
	1.3	The Fia	sco.OC Microkernel	3
		1.3.1	Communication	3
		1.3.2	Kernel Objects	3
	1.4	L4 Runt	time Environment (L4Re)	3
	1.5	Introduc	ction to L4Re's concepts	4
	1.6	Memory	y management - Data Spaces and the Region Map	4
		1.6.1	User-level paging	4
		1.6.2	Data spaces	4
		1.6.3	Virtual Memory Handling	4
		1.6.4	Memory Allocation	4
	1.7	Capabil	ities and Naming	5
	1.8	Initial E	nvironment and Application Bootstrapping	6
		1.8.1	Configuring an application before startup	6
		1.8.2	Connecting clients and servers	6
	1.9	Progran	n Input and Output	7
	1.10	Initial M	lemory Allocator and Factory	7
	1.11	Applica	tion and Server Building Blocks	8
		1.11.1	Creating Additional Application Threads	8
		1.11.2	Providing a Service	8
2	Getti	ing Stari	ted	9
		3		
3	L4Re	Server	s	11
	3.1	Sigma0	, the Root Pager	11
	3.2	Moe, th	e Root Task	11
	3.3	Ned, the	e Default Init Process	11
	3.4	Io, the F	Platform and Device Resource Manager	12
	3.5	Mag, th	e GUI Multiplexer	12
	3.6	fh-dry t	he Low-Level Graphics Driver	12

iv CONTENTS

	3.7	Rtc, the	e Real-Time Clock Server	12
	3.8	Moe, th	ne Root-Task	12
		3.8.1	Memory Allocator, Generic Factory	13
		3.8.2	Name-Space Provider	13
		3.8.3	Boot FS	13
		3.8.4	Log Subsystem	13
		3.8.5	Command-Line Options	13
			3.8.5.1debug= <debug flags=""></debug>	13
			3.8.5.2init= <init process=""></init>	14
			3.8.5.3l4re-dbg= <debug flags=""></debug>	14
			3.8.5.4Idr-flags= <loader flags=""></loader>	14
	3.9	Ned, th	ne Init Process	14
		3.9.1	Lua Bindings for L4Re	14
			3.9.1.1 Capabilities in Lua	14
			3.9.1.2 Access to L4Re::Env Capabilities	15
			3.9.1.3 Constants	15
			3.9.1.4 Application Startup Details	15
	3.10	lo, the	lo Server	16
	Dales	d C		10
4	Pull	ead Sup	pport	19
5	Mod	ule Inde	ex	21
	5.1	Module	es	21
6	Nam	espace	Index	25
Ĭ	6.1	_		25
7	Hiera	archical	Index	27
	7.1	Class I	Hierarchy	27
8	Doto	Ctructi	ure Index	33
0	8.1		tructures	33
	0.1	Dala 3	tructures	33
9	Mod	ule Doc	umentation	41
	9.1	C++ E	ceptions	41
		9.1.1	Detailed Description	42
	9.2	Small (C++ Template Library	43
		9.2.1	Detailed Description	44
		9.2.2	Function Documentation	44
			9.2.2.1 min	44
			9.2.2.2 max	44
			9.2.2.3 operator new	44

CONTENTS

9.3	Client/S	Server IPC	C Framework	 . 45
	9.3.1	Detailed	Description	 . 45
9.4	IPC Str	reams		 . 46
	9.4.1	Detailed	Description	 . 46
	9.4.2	Function	Documentation	 . 46
		9.4.2.1	operator>>	 . 46
		9.4.2.2	operator>>	 . 47
		9.4.2.3	operator>>	 . 47
		9.4.2.4	operator>>	 . 48
		9.4.2.5	operator>>	 . 49
		9.4.2.6	operator<<	 . 49
		9.4.2.7	operator<<	 . 50
		9.4.2.8	operator<<	 . 50
		9.4.2.9	operator<<	 . 51
9.5	IPC Me	essaging F	Framework	 . 53
	9.5.1	Detailed	Description	 . 53
	9.5.2	Function	Documentation	 . 53
		9.5.2.1	buf_cp_out	 . 53
		9.5.2.2	buf_cp_in	 . 54
		9.5.2.3	msg_ptr	 . 54
		9.5.2.4	buf_in	 . 54
9.6	L4Re C	C++ Interfa	ace	 . 55
	9.6.1	Detailed	Description	 . 56
9.7	L4Re U	Jtil C++ Int	nterface	 . 57
	9.7.1	Detailed	Description	 . 57
9.8	Datasp	ace interfa	face	 . 58
	9.8.1	Detailed	Description	 . 58
	9.8.2	Function	Documentation	 . 58
		9.8.2.1	l4re_ds_clear	 . 58
		9.8.2.2	l4re_ds_allocate	 . 59
		9.8.2.3	l4re_ds_copy_in	 . 59
		9.8.2.4	l4re_ds_size	 . 59
		9.8.2.5	l4re_ds_flags	 . 59
		9.8.2.6	l4re_ds_info	 . 59
		9.8.2.7	l4re_ds_phys	 . 59
9.9	Debug	interface		 . 61
	9.9.1	Detailed	Description	 . 61
	9.9.2	Function	Documentation	 . 61
		9.9.2.1	l4re_debug_obj_debug	 . 61
9.10	Event i	nterface		 . 62

vi CONTENTS

	9.10.1	Detailed Description	62
	9.10.2	Function Documentation	62
		9.10.2.1	62
		9.10.2.2 I4re_event_get_num_streams	63
		9.10.2.3	64
		9.10.2.4 l4re_event_get_stream_info_for_id	64
		9.10.2.5	64
9.11	Log into	erface	66
	9.11.1	Detailed Description	66
	9.11.2	Function Documentation	66
		9.11.2.1	66
		9.11.2.2	67
		9.11.2.3	67
		9.11.2.4	68
9.12	Memor	y allocator	69
	9.12.1	Detailed Description	69
	9.12.2	Enumeration Type Documentation	69
		9.12.2.1	69
	9.12.3	Function Documentation	70
		9.12.3.1	70
		9.12.3.2	71
		9.12.3.3	72
		9.12.3.4	73
		9.12.3.5	73
9.13	Names	pace interface	75
	9.13.1	Detailed Description	75
	9.13.2	Enumeration Type Documentation	75
		9.13.2.1	75
	9.13.3	Function Documentation	75
		9.13.3.1	75
		9.13.3.2	76
9.14	Region	map interface	77
	9.14.1	Detailed Description	78
	9.14.2	Enumeration Type Documentation	78
		9.14.2.1	78
	9.14.3	Function Documentation	78
		9.14.3.1	78
		9.14.3.2 l4re_rm_free_area	79
		9.14.3.3	79
		9.14.3.4	80

CONTENTS vii

		9.14.3.5	81
		9.14.3.6	81
		9.14.3.7	82
		9.14.3.8	83
		9.14.3.9	83
		9.14.3.10 l4re_rm_reserve_area_srv	84
		9.14.3.11 l4re_rm_free_area_srv	84
		9.14.3.12 l4re_rm_attach_srv	85
		9.14.3.13 l4re_rm_detach_srv	85
		9.14.3.14 l4re_rm_find_srv	85
9.15	Capabi	lity allocator	87
	9.15.1	Detailed Description	87
	9.15.2	Function Documentation	87
		9.15.2.1	87
9.16	Kumen	allocator utility	88
	9.16.1	Detailed Description	88
	9.16.2	Function Documentation	88
		9.16.2.1 l4re_util_kumem_alloc	88
9.17	Video A	API	89
	9.17.1	Detailed Description	90
	9.17.2	Typedef Documentation	90
		9.17.2.1	90
	9.17.3	Enumeration Type Documentation	90
		9.17.3.1 l4re_video_goos_info_flags_t	90
		9.17.3.2 l4re_video_view_info_flags_t	91
	9.17.4	Function Documentation	91
		9.17.4.1	91
		9.17.4.2	91
		9.17.4.3 l4re_video_goos_create_buffer	92
		9.17.4.4 l4re_video_goos_delete_buffer	92
		9.17.4.5 l4re_video_goos_get_static_buffer	92
		9.17.4.6 l4re_video_goos_create_view	92
		9.17.4.7 dec_video_goos_delete_view	93
		9.17.4.8	93
		9.17.4.9	93
		9.17.4.10 l4re_video_view_get_info	93
		9.17.4.11	94
		9.17.4.12 l4re_video_view_set_viewport	95
		9.17.4.13 l4re_video_view_stack	95
9.18	Consol	e API	96

viii CONTENTS

	9.18.1 Detailed Description	96
9.19	Data-Space API	97
	9.19.1 Detailed Description	97
9.20	Debugging API	98
	9.20.1 Detailed Description	98
9.21	L4Re ELF Auxiliary Information	99
	9.21.1 Detailed Description	100
	9.21.2 Macro Definition Documentation	100
	9.21.2.1 L4RE_ELF_AUX_ELEM	100
	9.21.2.2 L4RE_ELF_AUX_ELEM_T	100
	9.21.3 Enumeration Type Documentation	100
	9.21.3.1 anonymous enum	100
9.22	Initial Environment	101
	9.22.1 Detailed Description	101
	9.22.2 Typedef Documentation	102
	9.22.2.1	102
	9.22.3 Function Documentation	102
	9.22.3.1 l4re_env	102
	9.22.3.2	103
	9.22.3.3	103
	9.22.3.4	104
	9.22.3.5	105
9.23	Event API	107
	9.23.1 Detailed Description	107
9.24	Auxiliary data	108
	9.24.1 Detailed Description	108
9.25	Logging interface	109
	9.25.1 Detailed Description	109
9.26	Memory allocator API	110
	9.26.1 Detailed Description	110
9.27	Name-space API	111
	9.27.1 Detailed Description	111
9.28	Parent API	112
	9.28.1 Detailed Description	112
9.29	L4Re Protocol identifiers	113
	9.29.1 Detailed Description	113
	9.29.2 Enumeration Type Documentation	113
	9.29.2.1 Protocols	113
9.30	Region map API	115
	9.30.1 Detailed Description	115

CONTENTS

9.31	L4Re Capability API	116
	9.31.1 Detailed Description	116
	9.31.2 Variable Documentation	116
	9.31.2.1 cap_alloc	116
9.32	Kumem utiltities	118
	9.32.1 Detailed Description	118
	9.32.2 Function Documentation	118
	9.32.2.1 kumem_alloc	118
9.33	Goos video API	119
	9.33.1 Detailed Description	119
9.34	L4Re C Interface	120
	9.34.1 Detailed Description	121
9.35	L4Re Util C Interface	122
9.36	Base API	123
	9.36.1 Detailed Description	125
9.37	IPC-Gate API	126
	9.37.1 Detailed Description	126
	9.37.2 Enumeration Type Documentation	127
	9.37.2.1 L4_ipc_gate_ops	127
	9.37.3 Function Documentation	127
	9.37.3.1 I4_ipc_gate_bind_thread	127
	9.37.3.2 I4_ipc_gate_get_infos	127
9.38	Basic Macros	129
	9.38.1 Detailed Description	130
	9.38.2 Macro Definition Documentation	130
	9.38.2.1 L4_DECLARE_CONSTRUCTOR	130
	9.38.2.2 L4_NOTHROW	
	9.38.2.3 L4_EXPORT	131
	9.38.2.4 L4_HIDDEN	131
9.39		132
	•	134
		134
	9.39.2.1 fiasco_tbuf_get_status	134
	9.39.2.2 fiasco_tbuf_get_status_phys	134
	9.39.2.3 fiasco_tbuf_log	134
		135
	,	136
		136
	9.39.2.7 fiasco_tbuf_dump	
	9.39.2.8 fiasco_watchdog_takeover	137

CONTENTS

		9.39.2.9 fiasco_watchdog_touch
		9.39.2.10 fiasco_ldt_set
		9.39.2.11 fiasco_gdt_set
		9.39.2.12 fiasco_gdt_get_entry_offset
9.40	Fiasco	real time scheduling extensions
9.41	Flex pa	ges
	9.41.1	Detailed Description
	9.41.2	Enumeration Type Documentation
		9.41.2.1 4_fpage_consts
		9.41.2.2 anonymous enum
		9.41.2.3 L4_fpage_rights
		9.41.2.4 L4_cap_fpage_rights
		9.41.2.5 I4_fpage_cacheability_opt_t
		9.41.2.6 anonymous enum
	9.41.3	Function Documentation
		9.41.3.1 l4_fpage
		9.41.3.2 4_fpage_all
		9.41.3.3 l4_fpage_invalid
		9.41.3.4 l4_iofpage
		9.41.3.5 l4_obj_fpage
		9.41.3.6 4_is_fpage_writable
		9.41.3.7 l4_fpage_rights
		9.41.3.8 l4_fpage_type
		9.41.3.9 l4_fpage_size
		9.41.3.10 l4_fpage_page
		9.41.3.11 l4_fpage_set_rights
		9.41.3.12 I4_fpage_contains
		9.41.3.13 l4_fpage_max_order
9.42	Messag	ge Items
	9.42.1	Detailed Description
	9.42.2	Enumeration Type Documentation
		9.42.2.1 I4_msg_item_consts_t
	9.42.3	Function Documentation
		9.42.3.1 4_map_control
		9.42.3.2 14_map_obj_control
		ıts
		Detailed Description
	9.43.2	Macro Definition Documentation
		9.43.2.1 L4_IPC_TIMEOUT_0
	9.43.3	Typedef Documentation

CONTENTS xi

		9.43.3.1 14_t	imeout_s			 	 	 . 154
		9.43.3.2 14_t	imeout_t			 	 	 . 154
	9.43.4	Enumeration ¹	Type Documenta	tion		 	 	 . 155
		9.43.4.1 4_t	imeout_abs_vali	dity		 	 	 . 155
	9.43.5	Function Docu	umentation			 	 	 . 155
		9.43.5.1 l4_t	imeout_rel			 	 	 . 155
		9.43.5.2 4_i	pc_timeout			 	 	 . 155
		9.43.5.3 l4_t	imeout			 	 	 . 155
		9.43.5.4 14_6	and_timeout			 	 	 . 156
		9.43.5.5 4_1	rcv_timeout			 	 	 . 157
		9.43.5.6 4_t	imeout_rel_get			 	 	 . 157
		9.43.5.7 4_t	imeout_is_absol	ute		 	 	 . 157
		9.43.5.8 l4_t	imeout_get			 	 	 . 158
		9.43.5.9 l4_t	imeout_abs			 	 	 . 159
		9.43.5.10 14_0	utcb_mr64_idx			 	 	 . 160
9.44	VM API	for SVM				 	 	 . 161
			ription					
9.45	VM API	for VMX				 	 	 . 162
			ription					
	9.45.2	Enumeration 7	Type Documenta	tion		 	 	 . 163
		9.45.2.1 L4_	vm_vmx_caps_r	egs		 	 	 . 163
			vm_vmx_dfl1_re	_				
		9.45.2.3 and	nymous enum			 	 	 . 164
	9.45.3		umentation					
		9.45.3.1 4_\	/m_vmx_get_cap	os		 	 	 . 164
		9.45.3.2 14_\	/m_vmx_get_cap	os_default	1	 	 	 . 164
		_	/m_vmx_field_le					
		9.45.3.4 14_\	/m_vmx_field_or	der		 	 	 . 165
		_	/m_vmx_clear					
		9.45.3.6 14_\	/m_vmx_ptr_load	d		 	 	 . 166
		9.45.3.7 14_\	/m_vmx_get_cr2	_index .		 	 	 . 167
		9.45.3.8 14_\	/m_vmx_read_na	at		 	 	 . 167
		9.45.3.9 14_\	/m_vmx_read_1	3		 	 	 . 168
		9.45.3.10 14_\	/m_vmx_read_3	2		 	 	 . 168
		9.45.3.11 14_\	/m_vmx_read_64	4		 	 	 . 169
		9.45.3.12 L4_	vm_vmx_read			 	 	 . 169
			/m_vmx_write_n					
			/m_vmx_write_1					
			/m_vmx_write_3					
		9.45.3.16 14_\	/m_vmx_write_6	4		 	 	 . 171

xii CONTENTS

		9.45.3.17 l4_vm_vmx_write	'2
9.46	Cache	Consistency	'3
	9.46.1	Detailed Description	'3
	9.46.2	Function Documentation	'3
		9.46.2.1 l4_cache_clean_data	'3
		9.46.2.2 l4_cache_flush_data	'3
		9.46.2.3 l4_cache_inv_data	'4
		9.46.2.4 I4_cache_coherent	'4
		9.46.2.5 I4_cache_dma_coherent	'4
9.47	Memor	y related	'5
	9.47.1	Detailed Description	'6
	9.47.2	Macro Definition Documentation	'6
		9.47.2.1 L4_PAGEMASK	'6
		9.47.2.2 L4_LOG2_PAGESIZE	'6
		9.47.2.3 L4_SUPERPAGESIZE	'6
		9.47.2.4 L4_SUPERPAGEMASK	'6
		9.47.2.5 L4_LOG2_SUPERPAGESIZE	7
	9.47.3	Enumeration Type Documentation	'7
		9.47.3.1	'7
	9.47.4	Function Documentation	'7
		9.47.4.1 l4_trunc_page	7
		9.47.4.2 l4_trunc_size	'7
		9.47.4.3 l4_round_page	'8
		9.47.4.4 l4_round_size	'8
9.48	Kernel	Debugger	30
	9.48.1	Detailed Description	31
	9.48.2	Macro Definition Documentation	31
		9.48.2.1 enter_kdebug	31
		9.48.2.2 asm_enter_kdebug	32
		9.48.2.3 kd_display	32
		9.48.2.4 ko	32
		9.48.2.5 enter_kdebug	32
		9.48.2.6 asm_enter_kdebug	33
		9.48.2.7 kd_display	33
		9.48.2.8 ko	33
	9.48.3	Function Documentation	34
		9.48.3.1	}4
		9.48.3.2 l4_debugger_global_id	35
		9.48.3.3 l4_debugger_kobj_to_id	35
		9.48.3.4 outchar	36

CONTENTS xiii

		9.48.3.5 outstring	86
		9.48.3.6 outnstring	86
		9.48.3.7 outhex32	87
		9.48.3.8 outhex20	87
		9.48.3.9 outhex16	87
		9.48.3.10 outhex12	87
		9.48.3.11 outhex8	87
		9.48.3.12 outdec	87
		9.48.3.13 l4kd_inchar	88
9.49	Error co	odes	89
	9.49.1	Detailed Description	89
	9.49.2	Enumeration Type Documentation	89
		9.49.2.1	89
9.50	Factory	/	91
		Detailed Description	
	9.50.2	Function Documentation	
		9.50.2.1 I4_factory_create_task	
		9.50.2.2 4_factory_create_thread	
		9.50.2.3 4_factory_create_factory	94
		9.50.2.4 4_factory_create_gate	94
		9.50.2.5 I4_factory_create_irq	95
		9.50.2.6 I4_factory_create_vm	96
9.51	Virtual	Machines	97
	9.51.1	Detailed Description	97
9.52		ot controller	
		Detailed Description	99
	9.52.2	Typedef Documentation	99
			99
	9.52.3	21	99
		0	99
	9.52.4		99
			99
		9.52.4.2 4_icu_unbind	00
			00
			01
			01
		9.52.4.6	
		9.52.4.7 4_icu_mask	
9.53	Object	Invocation	05
	9.53.1	Detailed Description	06

XIV

	9.53.2	Enumeration Type Documentation	
		9.53.2.1 4_syscall_flags_t	
	9.53.3	Function Documentation	207
		9.53.3.1 I4_ipc_send	207
		9.53.3.2 4_ipc_wait	208
		9.53.3.3 l4_ipc_receive	209
		9.53.3.4 4_ipc_call	210
		9.53.3.5	211
		9.53.3.6 4_ipc_send_and_wait	13
		9.53.3.7 l4_ipc	214
		9.53.3.8 l4_ipc_sleep	15
		9.53.3.9 14_sndfpage_add	216
9.54	Error H	andling	118
	9.54.1	Detailed Description	118
	9.54.2	Enumeration Type Documentation	118
		9.54.2.1 4_ipc_tcr_error_t	118
	9.54.3	Function Documentation	
		9.54.3.1 l4_ipc_error	19
		9.54.3.2 l4_error	220
		9.54.3.3 l4_ipc_is_snd_error	221
		9.54.3.4 14_ipc_is_rcv_error	222
		9.54.3.5 14_ipc_error_code	<u>2</u> 22
9.55	Realtim	e API	223
9.56	IRQs .		224
	9.56.1	Detailed Description	225
	9.56.2	Enumeration Type Documentation	225
		9.56.2.1 L4_irq_mode	225
	9.56.3	Function Documentation	225
		9.56.3.1 l4_irq_attach	225
		9.56.3.2 l4_irq_chain	226
		9.56.3.3 l4_irq_detach	226
		9.56.3.4 l4_irq_trigger	227
		9.56.3.5 14_irq_receive	228
		9.56.3.6 4_irq_wait	229
		9.56.3.7 l4_irq_unmask	229
9.57	Kernel	Objects	230
	9.57.1	Detailed Description	231
9.58	Kernel	Interface Page	232
	9.58.1	Detailed Description	233
	9.58.2	Function Documentation	233

CONTENTS xv

		9.58.2.1 l4_kip_version
		9.58.2.2 l4_kip_version_string
		9.58.2.3 14_kernel_info_version_offset
		9.58.2.4 l4_kip_clock
		9.58.2.5 4_kip_clock_lw
9.59	Memor	ry descriptors (C version)
	9.59.1	Detailed Description
	9.59.2	Typedef Documentation
		9.59.2.1 l4_kernel_info_mem_desc_t
	9.59.3	Enumeration Type Documentation
		9.59.3.1 l4_mem_type_t
	9.59.4	Function Documentation
		9.59.4.1 l4_kernel_info_get_num_mem_descs
		9.59.4.2 4_kernel_info_set_mem_desc
		9.59.4.3 l4_kernel_info_get_mem_desc_start
		9.59.4.4 4_kernel_info_get_mem_desc_end
		9.59.4.5
		9.59.4.6 l4_kernel_info_get_mem_desc_subtype
		9.59.4.7 l4_kernel_info_get_mem_desc_is_virtual
9.60	Schedu	uler
	9.60.1	Detailed Description
	9.60.2	Enumeration Type Documentation
		9.60.2.1 L4_scheduler_ops
	9.60.3	Function Documentation
		9.60.3.1 l4_sched_cpu_set
		9.60.3.2 l4_scheduler_info
		9.60.3.3 l4_scheduler_run_thread
		9.60.3.4 l4_scheduler_idle_time
		9.60.3.5 l4_scheduler_is_online
9.61	Task .	
	9.61.1	Detailed Description
	9.61.2	Enumeration Type Documentation
		9.61.2.1 4_unmap_flags_t
	9.61.3	Function Documentation
		9.61.3.1 l4_task_map
		9.61.3.2 l4_task_unmap
		9.61.3.3 l4_task_unmap_batch
		9.61.3.4 l4_task_delete_obj
		9.61.3.5 l4_task_release_cap
		9.61.3.6 l4_task_cap_valid

xvi CONTENTS

		9.61.3.7 l4_task_cap_has_child	251
		9.61.3.8 l4_task_cap_equal	252
		9.61.3.9 l4_task_add_ku_mem	253
9.62	Thread	1	255
	9.62.1	Detailed Description	256
	9.62.2	Enumeration Type Documentation	257
		9.62.2.1 L4_thread_ops	257
		9.62.2.2 L4_thread_control_flags	257
		9.62.2.3 L4_thread_control_mr_indices	258
		9.62.2.4 L4_thread_ex_regs_flags	258
	9.62.3	Function Documentation	258
		9.62.3.1 l4_thread_ex_regs	258
		9.62.3.2 14_thread_ex_regs_ret	259
		9.62.3.3 I4_thread_yield	259
		9.62.3.4 I4_thread_switch	260
		9.62.3.5 I4_thread_stats_time	260
		9.62.3.6 I4_thread_vcpu_resume_start	261
		9.62.3.7 I4_thread_vcpu_resume_commit	261
		9.62.3.8 I4_thread_vcpu_control	62
		9.62.3.9 I4_thread_vcpu_control_ext	263
		9.62.3.10 l4_thread_register_del_irq	263
		9.62.3.11 I4_thread_modify_sender_start	264
		9.62.3.12 l4_thread_modify_sender_add	264
		9.62.3.13 I4_thread_modify_sender_commit	265
		9.62.3.14 I4_thread_arm_set_tpidruro	266
9.63	Thread	control	267
	9.63.1	Detailed Description	267
	9.63.2	Function Documentation	268
		9.63.2.1 4_thread_control_start	268
		9.63.2.2 4_thread_control_pager	268
		9.63.2.3 l4_thread_control_exc_handler	269
		9.63.2.4 4_thread_control_bind	270
		9.63.2.5 4_thread_control_alien	271
		9.63.2.6 l4_thread_control_ux_host_syscall	72
		9.63.2.7 14_thread_control_commit	72
9.64	Messag	ge Tag	274
	9.64.1	Detailed Description	275
	9.64.2	Typedef Documentation	275
		9.64.2.1 l4_msgtag_t	275
	9.64.3	Enumeration Type Documentation	275

CONTENTS xvii

	9.64.3.1 4_msgtag_protocol
	9.64.3.2 l4_msgtag_flags
9.64.4	Function Documentation
	9.64.4.1 l4_msgtag
	9.64.4.2 4_msgtag_label
	9.64.4.3 l4_msgtag_words
	9.64.4.4 l4_msgtag_items
	9.64.4.5 4_msgtag_flags
	9.64.4.6
	9.64.4.7
	9.64.4.8 l4_msgtag_is_preemption
	9.64.4.9 l4_msgtag_is_sys_exception
	9.64.4.10 l4_msgtag_is_exception
	9.64.4.11 I4_msgtag_is_sigma0
	9.64.4.12 l4_msgtag_is_io_page_fault
9.65 Capal	bilities
9.65.1	Detailed Description
9.65.2	2 Macro Definition Documentation
	9.65.2.1 L4_DISABLE_COPY
	9.65.2.2 L4_KOBJECT_DISABLE_COPY
	9.65.2.3 L4_KOBJECT
9.65.3	3 Typedef Documentation
	9.65.3.1 l4_cap_idx_t
9.65.4	Enumeration Type Documentation
	9.65.4.1 4_cap_consts_t
	9.65.4.2 l4_default_caps_t
9.65.5	Function Documentation
	9.65.5.1 cap_cast
	9.65.5.2 cap_reinterpret_cast
	9.65.5.3 cap_dynamic_cast
	9.65.5.4 4_is_invalid_cap
	9.65.5.5 l4_is_valid_cap
	9.65.5.6 4_capability_equal
9.66 Virtua	Registers (UTCBs)
9.66.1	Detailed Description
9.66.2	2 Typedef Documentation
	9.66.2.1 l4_utcb_t
9.66.3	Function Documentation
	9.66.3.1 4_utcb_mr
	9.66.3.2 l4_utcb_br

xviii CONTENTS

9.66.3.3 4_utcb_tcr	294
9.67 Message Registers (MRs)	295
9.67.1 Detailed Description	295
9.68 Buffer Registers (BRs)	296
9.68.1 Detailed Description	296
9.68.2 Enumeration Type Documentation	296
9.68.2.1	296
9.69 Thread Control Registers (TCRs)	297
9.69.1 Detailed Description	297
9.70 Exception registers	298
9.70.1 Detailed Description	298
9.70.2 Function Documentation	298
9.70.2.1 l4_utcb_exc	298
9.70.2.2 l4_utcb_exc_pc	299
9.70.2.3 4_utcb_exc_pc_set	299
9.70.2.4 l4_utcb_exc_is_pf	299
9.71 Virtual Console	300
9.71.1 Detailed Description	301
9.71.2 Enumeration Type Documentation	301
9.71.2.1 L4_vcon_write_consts	301
9.71.2.2 L4_vcon_i_flags	301
9.71.2.3 L4_vcon_o_flags	301
9.71.2.4 L4_vcon_l_flags	301
9.71.2.5 L4_vcon_ops	302
9.71.3 Function Documentation	302
9.71.3.1 l4_vcon_send	302
9.71.3.2 l4_vcon_write	302
9.71.3.3 l4_vcon_read	303
9.71.3.4 I4_vcon_set_attr	303
9.71.3.5 4_vcon_get_attr	304
9.72 vCPU API	305
9.72.1 Detailed Description	306
9.72.2 Enumeration Type Documentation	306
9.72.2.1 L4_vcpu_state_flags	306
9.72.2.2 L4_vcpu_sticky_flags	306
9.72.2.3 L4_vcpu_state_offset	306
9.73 Fiasco-UX Virtual devices	307
9.73.1 Detailed Description	307
9.73.2 Enumeration Type Documentation	307
9.73.2.1 l4_vhw_entry_type	307

CONTENTS xix

9.74	Memor	y operations
	9.74.1	Detailed Description
	9.74.2	Enumeration Type Documentation
		9.74.2.1 L4_mem_op_widths
	9.74.3	Function Documentation
		9.74.3.1 I4_mem_read
		9.74.3.2 4_mem_write
9.75	ARM V	irtual Registers (UTCB)
	9.75.1	Detailed Description
9.76	VM AP	l for TZ
	9.76.1	Detailed Description
9.77	amd64	Virtual Registers (UTCB)
	9.77.1	Detailed Description
9.78	x86 Vir	tual Registers (UTCB)
	9.78.1	Detailed Description
	9.78.2	Enumeration Type Documentation
		9.78.2.1 L4_utcb_consts_x86
9.79	CPU re	elated functions
	9.79.1	Detailed Description
	9.79.2	Function Documentation
		9.79.2.1
		9.79.2.2
		9.79.2.3
9.80	Functio	ons to manipulate the local IDT
	9.80.1	Detailed Description
9.81	Timesta	amp Counter
	9.81.1	Detailed Description
	9.81.2	Function Documentation
		9.81.2.1 l4_rdtsc
		9.81.2.2 l4_rdtsc_32
		9.81.2.3 l4_rdpmc
		9.81.2.4 l4_rdpmc_32
		9.81.2.5 l4_tsc_to_ns
		9.81.2.6 l4_tsc_to_us
		9.81.2.7 l4_tsc_to_s_and_ns
		9.81.2.8 l4_ns_to_tsc
		9.81.2.9 l4_busy_wait_ns
		9.81.2.10 l4_busy_wait_us
		9.81.2.11 l4_calibrate_tsc
		9.81.2.12 l4_tsc_init

CONTENTS

	9.81.2.13 l4_get_hz
9.82 Atomic	Instructions
9.82.1	Detailed Description
9.82.2	Function Documentation
	9.82.2.1 l4util_cmpxchg64
	9.82.2.2
	9.82.2.3
	9.82.2.4
	9.82.2.5
	9.82.2.6
	9.82.2.7
	9.82.2.8
	9.82.2.9
	9.82.2.10 4util_add8
	9.82.2.11 4util_add8_res
	9.82.2.12 4util_inc8
	9.82.2.13 4util_inc8_res
	9.82.2.14 4util_atomic_add
	9.82.2.15
9.83 Interna	al functions
9.83.1	Detailed Description
9.84 Bit Ma	nipulation
9.84.1	Detailed Description
9.84.2	Function Documentation
	9.84.2.1
	9.84.2.2
	9.84.2.3
	9.84.2.4
	9.84.2.5
	9.84.2.6
	9.84.2.7
	9.84.2.8
	9.84.2.9
	9.84.2.10
	9.84.2.11
	9.84.2.12
9.85 ELF bi	inary format
9.85.1	Detailed Description
9.85.2	Macro Definition Documentation
	9.85.2.1 EI_CLASS

CONTENTS xxi

9.85.2.2 EI_CLASS	350
9.85.2.3 ELFCLASSNONE	350
9.85.2.4 ELFCLASSNONE	350
9.85.2.5 EI_DATA	350
9.85.2.6 EI_DATA	350
9.85.2.7 ELFDATANONE	350
9.85.2.8 ELFDATANONE	351
9.85.2.9 ELFDATA2LSB	351
9.85.2.10 ELFDATA2LSB	351
9.85.2.11 ELFDATA2MSB	351
9.85.2.12 ELFDATA2MSB	351
9.85.2.13 EI_VERSION	351
9.85.2.14 EI_VERSION	351
9.85.2.15 EI_OSABI	352
9.85.2.16 EI_OSABI	352
9.85.2.17 ELFOSABI_SYSV	352
9.85.2.18 ELFOSABI_SYSV	352
9.85.2.19 ELFOSABI_HPUX	352
9.85.2.20 ELFOSABI_HPUX	352
9.85.2.21 ELFOSABI_NETBSD	352
9.85.2.22 ELFOSABI_LINUX	352
9.85.2.23 ELFOSABI_SOLARIS	353
9.85.2.24 ELFOSABI_AIX	353
9.85.2.25 ELFOSABI_IRIX	353
9.85.2.26 ELFOSABI_FREEBSD	353
9.85.2.27 ELFOSABI_TRU64	353
9.85.2.28 ELFOSABI_MODESTO	353
9.85.2.29 ELFOSABI_OPENBSD	353
9.85.2.30 EI_PAD	353
9.85.2.31 EI_PAD	353
9.85.2.32 EM_ARC	354
9.85.2.33 SHT_NUM	354
9.85.2.34 SHF_GROUP	354
9.85.2.35 SHF_TLS	354
9.85.2.36 SHF_MASKOS	354
9.85.2.37 PT_LOOS	354
9.85.2.38 PT_HIOS	354
9.85.2.39 PT_LOPROC	354
9.85.2.40 PT_HIPROC	
9.85.2.41 PT_GNU_EH_FRAME	355

xxii CONTENTS

		9.85.2.42 PT_GNU_STACK
		9.85.2.43 PT_GNU_RELRO
		9.85.2.44 PT_L4_STACK
		9.85.2.45 PT_L4_KIP
		9.85.2.46 PT_L4_AUX
		9.85.2.47 NT_VERSION
		9.85.2.48 DT_NULL
		9.85.2.49 DT_LOPROC
		9.85.2.50 DT_HIPROC
		9.85.2.51 DF_1_NOW
		9.85.2.52 DF_1_GLOBAL
		9.85.2.53 DF_1_GROUP
		9.85.2.54 DF_1_NODELETE
		9.85.2.55 DF_1_LOADFLTR
		9.85.2.56 DF_1_NOOPEN
		9.85.2.57 DF_1_ORIGIN
		9.85.2.58 DF_1_DIRECT
		9.85.2.59 DF_1_INTERPOSE
		9.85.2.60 DF_1_NODEFLIB
		9.85.2.61 DF_1_NODUMP
		9.85.2.62 DF_1_CONFALT
		9.85.2.63 DF_1_ENDFILTEE
		9.85.2.64 DF_1_DISPRELDNE
		9.85.2.65 DF_1_DISPRELPND
		9.85.2.66 DF_P1_LAZYLOAD
		9.85.2.67 DF_P1_GROUPPERM
9.86	Kernel	Interface Page API
	9.86.1	Detailed Description
	9.86.2	Macro Definition Documentation
		9.86.2.1
	9.86.3	Function Documentation
		9.86.3.1
		9.86.3.2
		9.86.3.3
		9.86.3.4
9.87	Comfor	table Command Line Parsing
	9.87.1	Detailed Description
	9.87.2	Function Documentation
		9.87.2.1 parse_cmdline
9.88	Priority	related functions

CONTENTS xxiii

	9.88.1	Detailed Description	62
9.89	Randor	m number support	63
	9.89.1	Detailed Description	63
	9.89.2	Function Documentation	63
		9.89.2.1	63
		9.89.2.2	63
9.90	Machin	e Restarting Function	64
	9.90.1	Detailed Description	64
9.91	Low-Le	vel Thread Functions	65
9.92	Utility F	Functions	66
	9.92.1	Detailed Description	67
	9.92.2	Function Documentation	67
		9.92.2.1	67
		9.92.2.2	68
		9.92.2.3	69
9.93	IA32 Po	ort I/O API	70
	9.93.1	Detailed Description	70
	9.93.2	Function Documentation	70
		9.93.2.1	70
		9.93.2.2	71
		9.93.2.3	71
		9.93.2.4	71
		9.93.2.5	71
		9.93.2.6	72
		9.93.2.7	72
		9.93.2.8 4util_out16	72
		9.93.2.9	72
		9.93.2.10 4util_outs8	72
		9.93.2.11 4util_outs16	73
		9.93.2.12 4util_outs32	73
9.94	Bitmap	graphics and fonts	74
	9.94.1	Detailed Description	74
9.95	Functio	ons for rendering bitmap data in frame buffers	75
	9.95.1	Detailed Description	75
	9.95.2	Typedef Documentation	75
		9.95.2.1 gfxbitmap_color_t	76
		9.95.2.2 gfxbitmap_color_pix_t	76
	9.95.3	Function Documentation	76
		9.95.3.1 gfxbitmap_convert_color	76
		9.95.3.2 gfxbitmap_fill	76

xxiv CONTENTS

		9.95.3.3 gfxbitmap_bmap	'6
		9.95.3.4 gfxbitmap_set	7
		9.95.3.5 gfxbitmap_copy	7
9.96	Functio	ons for rendering bitmap fonts to frame buffers	'8
	9.96.1	Detailed Description	'9
	9.96.2	Enumeration Type Documentation	'9
		9.96.2.1 anonymous enum	'9
	9.96.3	Function Documentation	'9
		9.96.3.1 gfxbitmap_font_init	'9
		9.96.3.2 gfxbitmap_font_get	'9
		9.96.3.3 gfxbitmap_font_width	'9
		9.96.3.4 gfxbitmap_font_height	'9
		9.96.3.5 gfxbitmap_font_data	30
		9.96.3.6 gfxbitmap_font_text	30
		9.96.3.7 gfxbitmap_font_text_scale	30
9.97	IO inter	rface	32
	9.97.1	Detailed Description	3
	9.97.2	Typedef Documentation	3
		9.97.2.1 l4io_resource_t	3
	9.97.3	Enumeration Type Documentation	3
		9.97.3.1 l4io_iomem_flags_t	3
		9.97.3.2 l4io_device_types_t	33
		9.97.3.3 l4io_resource_types_t	33
	9.97.4	Function Documentation	34
		9.97.4.1 l4io_request_iomem	}4
		9.97.4.2 l4io_request_iomem_region	}4
		9.97.4.3 l4io_release_iomem	34
		9.97.4.4 l4io_search_iomem_region	35
		9.97.4.5 I4io_request_ioport	36
		9.97.4.6 l4io_release_ioport	36
		9.97.4.7	36
		9.97.4.8 l4io_lookup_resource	37
		9.97.4.9 I4io_request_resource_iomem	37
		9.97.4.10 l4io_has_resource	37
9.98	IRQ ha	indling library	39
	9.98.1	Detailed Description	39
9.99	Interfac	be using direct functionality)0
	9.99.1	Detailed Description)0
	9.99.2	Function Documentation)0
		9.99.2.1) 0

CONTENTS xxv

9.99.2.2	 391
9.99.2.3 l4irq_attach_thread	 391
9.99.2.4 l4irq_attach_thread_ft	 391
9.99.2.5	 392
9.99.2.6 l4irq_unmask_and_wait_any	 392
9.99.2.7	 392
9.99.2.8	 392
9.99.2.9	 393
9.100 Interface for asynchronous ISR handlers	 394
9.100.1 Detailed Description	 394
9.100.2 Function Documentation	 394
9.100.2.1 l4irq_request	 394
9.100.2.2 l4irq_release	 395
9.101 Interface using direct functionality.	 396
9.101.1 Detailed Description	 396
9.101.2 Function Documentation	 396
9.101.2.1 l4irq_attach_cap	 396
9.101.2.2 l4irq_attach_cap_ft	 396
9.101.2.3 l4irq_attach_thread_cap	 397
9.101.2.4 l4irq_attach_thread_cap_ft	 397
9.102Interface for asynchronous ISR handlers with a given IRQ capability	398
9.102Interface for asynchronous ISR handlers with a given IRQ capability	 398
9.102Interface for asynchronous ISR handlers with a given IRQ capability	 398 398
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description	 398 398 398
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description	398 398 398 399
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API	398 398 398 399
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API 9.103.1 Detailed Description	398 398 398 399 400
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API 9.103.1 Detailed Description 9.103.2 Enumeration Type Documentation	398 398 398 399 400 400
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API 9.103.1 Detailed Description 9.103.2 Enumeration Type Documentation 9.103.2.1 I4sigma0_return_flags_t	398 398 399 400 400 400
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API 9.103.1 Detailed Description 9.103.2 Enumeration Type Documentation 9.103.2.1 I4sigma0_return_flags_t 9.103.3 Function Documentation	398 398 398 400 400 400
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API 9.103.1 Detailed Description 9.103.2 Enumeration Type Documentation 9.103.2.1 I4sigma0_return_flags_t 9.103.3 Function Documentation 9.103.3.1 I4sigma0_map_kip	398 398 398 400 400 400 400
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API 9.103.1 Detailed Description 9.103.2 Enumeration Type Documentation 9.103.2.1 I4sigma0_return_flags_t 9.103.3 Function Documentation 9.103.3.1 I4sigma0_map_kip 9.103.3.2 I4sigma0_map_mem	398 398 398 400 400 400 400 400 400
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API 9.103.1 Detailed Description 9.103.2 Enumeration Type Documentation 9.103.2.1 I4sigma0_return_flags_t 9.103.3 Function Documentation 9.103.3.1 I4sigma0_map_kip 9.103.3.2 I4sigma0_map_mem 9.103.3.3 I4sigma0_map_iomem	398 398 398 400 400 400 400 400 400 401
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API 9.103.1 Detailed Description 9.103.2 Enumeration Type Documentation 9.103.2.1 I4sigma0_return_flags_t 9.103.3 Function Documentation 9.103.3.1 I4sigma0_map_kip 9.103.3.2 I4sigma0_map_mem 9.103.3.3 I4sigma0_map_iomem 9.103.3.4 I4sigma0_map_iomem 9.103.3.5 I4sigma0_map_anypage 9.103.3.5 I4sigma0_map_tbuf 9.103.3.6 I4sigma0_debug_dump	398 398 398 400 400 400 400 400 401
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API 9.103.1 Detailed Description 9.103.2 Enumeration Type Documentation 9.103.2.1 I4sigma0_return_flags_t 9.103.3 Function Documentation 9.103.3.1 I4sigma0_map_kip 9.103.3.2 I4sigma0_map_mem 9.103.3.3 I4sigma0_map_iomem 9.103.3.4 I4sigma0_map_iomem 9.103.3.5 I4sigma0_map_tbuf 9.103.3.6 I4sigma0_debug_dump 9.103.3.7 I4sigma0_new_client	398 398 398 400 400 400 400 401 401
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API 9.103.1 Detailed Description 9.103.2 Enumeration Type Documentation 9.103.2.1 I4sigma0_return_flags_t 9.103.3 Function Documentation 9.103.3.1 I4sigma0_map_kip 9.103.3.2 I4sigma0_map_mem 9.103.3.3 I4sigma0_map_iomem 9.103.3.4 I4sigma0_map_iomem 9.103.3.5 I4sigma0_map_tbuf 9.103.3.6 I4sigma0_debug_dump 9.103.3.7 I4sigma0_new_client 9.103.3.8 I4sigma0_map_errstr	398 398 398 400 400 400 400 401 401 401 402
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API 9.103.1 Detailed Description 9.103.2 Enumeration Type Documentation 9.103.2.1 I4sigma0_return_flags_t 9.103.3 Function Documentation 9.103.3.1 I4sigma0_map_kip 9.103.3.2 I4sigma0_map_mem 9.103.3.3 I4sigma0_map_iomem 9.103.3.4 I4sigma0_map_iomem 9.103.3.5 I4sigma0_map_tbuf 9.103.3.6 I4sigma0_map_tbuf 9.103.3.7 I4sigma0_debug_dump 9.103.3.7 I4sigma0_new_client 9.103.3.8 I4sigma0_map_errstr	398 398 398 400 400 400 400 401 401 401 402
9.102Interface for asynchronous ISR handlers with a given IRQ capability. 9.102.1 Detailed Description 9.102.2 Function Documentation 9.102.2.1 I4irq_request_cap 9.103Sigma0 API 9.103.1 Detailed Description 9.103.2 Enumeration Type Documentation 9.103.2.1 I4sigma0_return_flags_t 9.103.3 Function Documentation 9.103.3.1 I4sigma0_map_kip 9.103.3.2 I4sigma0_map_mem 9.103.3.3 I4sigma0_map_iomem 9.103.3.4 I4sigma0_map_iomem 9.103.3.5 I4sigma0_map_tbuf 9.103.3.6 I4sigma0_debug_dump 9.103.3.7 I4sigma0_new_client 9.103.3.8 I4sigma0_map_errstr	398 398 398 400 400 400 400 401 401 401 402 402

XXVI

9.105.1 Detailed Description
9.105.2 Enumeration Type Documentation
9.105.2.1 l4vcpu_irq_state_t
9.105.3 Function Documentation
9.105.3.1 l4vcpu_state
9.105.3.2 l4vcpu_irq_disable
9.105.3.3 l4vcpu_irq_disable_save
9.105.3.4 l4vcpu_irq_enable
9.105.3.5 l4vcpu_irq_restore
9.105.3.6 l4vcpu_wait_for_event
9.105.3.7 4vcpu_print_state
9.105.3.8 l4vcpu_is_irq_entry
9.105.3.9 l4vcpu_is_page_fault_entry
9.106Extended vCPU support
9.106.1 Detailed Description
9.106.2 Function Documentation
9.106.2.1 l4vcpu_ext_alloc
9.107Shared Memory Library
9.107.1 Detailed Description
9.107.2 Function Documentation
9.107.2.1 l4shmc_create
9.107.2.2 l4shmc_attach
9.107.2.3 l4shmc_attach_to
9.107.2.4 l4shmc_connect_chunk_signal
9.107.2.5 l4shmc_area_size
9.107.2.6 l4shmc_area_size_free
9.107.2.7 4shmc_area_overhead
9.107.2.8 l4shmc_chunk_overhead
9.108Chunks
9.108.1 Detailed Description
9.108.2 Function Documentation
9.108.2.1 l4shmc_add_chunk
9.108.2.2 l4shmc_get_chunk
9.108.2.3 l4shmc_get_chunk_to
9.108.2.4 l4shmc_iterate_chunk
9.108.2.5 l4shmc_chunk_ptr
9.108.2.6 l4shmc_chunk_capacity
9.108.2.7 4shmc_chunk_signal
9.109Producer
9.109.1 Detailed Description

CONTENTS xxvii

9.109.2 Function Documentation	22
9.109.2.1 4shmc_chunk_try_to_take	22
9.109.2.2 4shmc_chunk_ready	22
9.109.2.3 l4shmc_chunk_ready_sig	23
9.109.2.4 4shmc_is_chunk_clear	23
9.110Consumer	24
9.110.1 Detailed Description	24
9.110.2 Function Documentation	24
9.110.2.1 4shmc_enable_chunk	24
9.110.2.2 I4shmc_wait_chunk	24
9.110.2.3 l4shmc_wait_chunk_to	25
9.110.2.4 I4shmc_wait_chunk_try	25
9.110.2.5 4shmc_chunk_consumed	25
9.110.2.6 l4shmc_is_chunk_ready	26
9.110.2.7 4shmc_chunk_size	27
9.111Signals	28
9.111.1 Detailed Description	28
9.111.2 Function Documentation	28
9.111.2.1 4shmc_add_signal	28
9.111.2.2 l4shmc_attach_signal	29
9.111.2.3 4shmc_attach_signal_to	29
9.111.2.4 l4shmc_get_signal_to	30
9.111.2.5 4shmc_signal_cap	31
9.111.2.6 l4shmc_check_magic	31
9.112Producer	32
9.112.1 Detailed Description	32
9.112.2 Function Documentation	32
9.112.2.1 l4shmc_trigger	32
9.113Consumer	33
9.113.1 Detailed Description	33
9.113.2 Function Documentation	33
9.113.2.1 I4shmc_enable_signal	33
9.113.2.2 l4shmc_wait_any	33
9.113.2.3 l4shmc_wait_any_try	34
9.113.2.4 l4shmc_wait_any_to	34
9.113.2.5 4shmc_wait_signal	34
9.113.2.6 4shmc_wait_signal_to	35
9.113.2.7 l4shmc_wait_signal_try	36
9.114Integer Types	37
9.114.1 Detailed Description	38

xxviii CONTENTS

	9.114.2	2 Typedef Documentation	438
		9.114.2.1 l4_int8_t	438
		9.114.2.2 l4_uint8_t	438
		9.114.2.3 l4_int16_t	438
		9.114.2.4 l4_uint16_t	439
		9.114.2.5 l4_int32_t	439
		9.114.2.6 l4_uint32_t	439
		9.114.2.7 l4_int64_t	439
		9.114.2.8 l4_uint64_t	439
40 No.		De companyation	444
		Documentation Paterseas	441 441
10.		mespace Reference	
40		Detailed Description	
10.		s Namespace Reference	
		Detailed Description	
10.		nespace Reference	
		Detailed Description	
	10.3.2	Function Documentation	
		10.3.2.1 kobject_typeid	
10.		_svr Namespace Reference	
		Detailed Description	
	10.4.2	Enumeration Type Documentation	
		10.4.2.1 Reply_mode	
10.		lamespace Reference	
		Detailed Description	
10.		Vfs Namespace Reference	
	10.6.1	Detailed Description	447
11 Da	ta Structı	ire Documentation	449
11.	1 L4::Allo	oc list Class Reference	449
	11.1.1	Detailed Description	449
11.	2 L4::Thr	ead::Attr Class Reference	449
	11.2.1	Detailed Description	450
	11.2.2	Constructor & Destructor Documentation	
		11.2.2.1 Attr	
	11.2.3	Member Function Documentation	
		11.2.3.1 pager	451
		11.2.3.2 pager	
		11.2.3.3 exc_handler	
		11.2.3.4 exc_handler	
		11.2.3.5 bind	

CONTENTS xxix

	11.2.3.6 ux_host_syscall	453
11.3 L4Re:	:Util::Auto_cap< T > Struct Template Reference	453
11.3.1	Detailed Description	454
11.4 L4Re:	:Util::Auto_del_cap< T > Struct Template Reference	455
11.4.1	Detailed Description	456
11.5 cxx::A	uto_ptr< T > Class Template Reference	456
11.5.1	Detailed Description	458
11.5.2	Member Typedef Documentation	458
	11.5.2.1 Ref_type	458
11.5.3	Constructor & Destructor Documentation	458
	11.5.3.1 Auto_ptr	458
	11.5.3.2 Auto_ptr	458
	11.5.3.3 ~Auto_ptr	458
11.5.4	Member Function Documentation	458
	11.5.4.1 operator=	458
	11.5.4.2 operator*	459
	11.5.4.3 operator->	459
	11.5.4.4 get	459
	11.5.4.5 release	459
	11 F. 4.C. apparator Drive type :	460
	11.5.4.6 operator Priv_type *	700
11.6 cxx::A	vl_map< Key, Data, Compare, Alloc > Class Template Reference	
		460
11.6.1	vl_map< Key, Data, Compare, Alloc > Class Template Reference	460 463
11.6.1	vl_map< Key, Data, Compare, Alloc > Class Template Reference	460 463 463
11.6.1	vI_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation	460 463 463 463
11.6.1	vI_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node	460 463 463 464
11.6.1	vI_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node 11.6.2.2 lower_bound_node	460 463 463 464 464
11.6.1	vI_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node 11.6.2.2 lower_bound_node 11.6.2.3 find	460 463 463 464 464 464
11.6.1	vI_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node 11.6.2.2 lower_bound_node 11.6.2.3 find 11.6.2.4 remove	460 463 463 464 464 464 465
11.6.1	vI_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node 11.6.2.2 lower_bound_node 11.6.2.3 find 11.6.2.4 remove 11.6.2.5 erase	460 463 463 464 464 464 465
11.6.1 11.6.2	VI_map< Key, Data, Compare, Alloc > Class Template ReferenceDetailed DescriptionMember Function Documentation11.6.2.1 find_node11.6.2.2 lower_bound_node11.6.2.3 find11.6.2.4 remove11.6.2.5 erase11.6.2.6 operator[]11.6.2.7 operator[]	460 463 463 464 464 465 465
11.6.1 11.6.2	vI_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node 11.6.2.2 lower_bound_node 11.6.2.3 find 11.6.2.4 remove 11.6.2.5 erase 11.6.2.6 operator[] 11.6.2.7 operator[] Field Documentation	460 463 463 464 464 465 465 465
11.6.1 11.6.2	VI_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node 11.6.2.2 lower_bound_node 11.6.2.3 find 11.6.2.4 remove 11.6.2.5 erase 11.6.2.6 operator[] 11.6.2.7 operator[] Field Documentation 11.6.3.1pad0	460 463 463 464 464 465 465 465
11.6.1 11.6.2 11.6.3	vl_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node 11.6.2.2 lower_bound_node 11.6.2.3 find 11.6.2.4 remove 11.6.2.5 erase 11.6.2.6 operator[] 11.6.2.7 operator[] Field Documentation 11.6.3.1pad0 vl_set< Item, Compare, Alloc > Class Template Reference	460 463 463 464 464 465 465 465
11.6.1 11.6.2 11.6.3 11.7 cxx::A 11.7.1	vl_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node 11.6.2.2 lower_bound_node 11.6.2.3 find 11.6.2.4 remove 11.6.2.5 erase 11.6.2.6 operator[] 11.6.2.7 operator[] Field Documentation 11.6.3.1pad0 vl_set< Item, Compare, Alloc > Class Template Reference Detailed Description	460 463 463 464 464 465 465 465 465 465
11.6.1 11.6.2 11.6.3 11.7 cxx::A 11.7.1	vI_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node 11.6.2.2 lower_bound_node 11.6.2.3 find 11.6.2.4 remove 11.6.2.5 erase 11.6.2.6 operator[] 11.6.2.7 operator[] Field Documentation 11.6.3.1pad0 vI_set< Item, Compare, Alloc > Class Template Reference Detailed Description Constructor & Destructor Documentation	460 463 463 464 464 465 465 465 465 466 469
11.6.1 11.6.2 11.6.3 11.7 cxx::A 11.7.1	vI_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node 11.6.2.2 lower_bound_node 11.6.2.3 find 11.6.2.4 remove 11.6.2.5 erase 11.6.2.6 operator[] 11.6.2.7 operator[] Field Documentation 11.6.3.1pad0 vI_set< Item, Compare, Alloc > Class Template Reference Detailed Description Constructor & Destructor Documentation 11.7.2.1 AvI_set	460 463 463 464 464 465 465 465 465 466 469
11.6.1 11.6.2 11.6.3 11.7 exx::A 11.7.1 11.7.2	vI_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node 11.6.2.2 lower_bound_node 11.6.2.3 find 11.6.2.4 remove 11.6.2.5 erase 11.6.2.6 operator[] 11.6.2.7 operator[] Field Documentation 11.6.3.1pad0 vI_set< Item, Compare, Alloc > Class Template Reference Detailed Description Constructor & Destructor Documentation 11.7.2.1 AvI_set 11.7.2.2 AvI_set	460 463 463 464 464 465 465 465 465 466 469 469
11.6.1 11.6.2 11.6.3 11.7 exx::A 11.7.1 11.7.2	vI_map< Key, Data, Compare, Alloc > Class Template Reference Detailed Description Member Function Documentation 11.6.2.1 find_node 11.6.2.2 lower_bound_node 11.6.2.3 find 11.6.2.4 remove 11.6.2.5 erase 11.6.2.6 operator[] 11.6.2.7 operator[] Field Documentation 11.6.3.1pad0 vI_set< Item, Compare, Alloc > Class Template Reference Detailed Description Constructor & Destructor Documentation 11.7.2.1 AvI_set 11.7.2.2 AvI_set	460 463 463 464 464 465 465 465 466 469 469 470 470

CONTENTS

11.7.3.3 find_node	71
11.7.3.4 lower_bound_node	72
11.7.3.5 begin	73
11.7.3.6 end	73
11.7.3.7 begin	73
11.7.3.8 end	74
11.7.3.9 rbegin	74
11.7.3.10 rend	74
11.7.3.11 rbegin	74
11.7.3.12 rend	74
11.8 cxx::Avl_tree< Node, Get_key, Compare > Class Template Reference	75
11.8.1 Detailed Description	77
11.8.2 Member Typedef Documentation	77
11.8.2.1 Iterator	77
11.8.3 Member Function Documentation	77
11.8.3.1 insert	77
11.8.3.2 remove	78
11.9 cxx::Avl_tree_node Class Reference	78
11.9.1 Detailed Description	30
11.10L4::Base_exception Class Reference	31
11.10.1 Detailed Description	32
11.11cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference 48	33
11.11.1 Detailed Description	34
11.11.2 Member Enumeration Documentation	35
11.11.2.1 anonymous enum	35
11.11.3 Member Function Documentation	35
11.11.3.1 total_objects	85
11.11.3.2 free_objects	85
11.12cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference 48	86
11.12.1 Detailed Description	88
11.12.2 Member Enumeration Documentation	89
11.12.2.1 anonymous enum	89
11.12.3 Member Function Documentation	89
11.12.3.1 alloc	89
11.12.3.2 free	89
11.12.3.3 total_objects	89
11.12.3.4 free_objects	90
11.13L4::Basic_registry Class Reference	90
11.13.1 Detailed Description	91
11.13.2 Member Typedef Documentation	91

CONTENTS xxxi

11.13.2.1 Value	491
11.13.3 Member Function Documentation	491
11.13.3.1 dispatch	491
11.14L4Re::Vfs::Be_file Class Reference	492
11.14.1 Detailed Description	494
11.14.2 Member Function Documentation	495
11.14.2.1 unlock_all_locks	495
11.14.2.2 data_space	495
11.14.2.3 fstat64	495
11.15L4Re::Vfs::Be_file_system Class Reference	496
11.15.1 Detailed Description	497
11.15.2 Constructor & Destructor Documentation	498
11.15.2.1 Be_file_system	498
11.15.2.2 ~Be_file_system	498
11.15.3 Member Function Documentation	498
11.15.3.1 type	498
11.16cxx::Bitfield< T, LSB, MSB > Class Template Reference	498
11.16.1 Detailed Description	500
11.16.2 Member Typedef Documentation	500
11.16.2.1 Bits_type	500
11.16.2.2 Shift_type	500
11.16.2.3 Ref	501
11.16.2.4 Val	501
11.16.2.5 Ref_unshifted	501
11.16.2.6 Val_unshifted	501
11.16.3 Member Enumeration Documentation	501
11.16.3.1 anonymous enum	501
11.16.3.2 Masks	501
11.16.4 Member Function Documentation	501
11.16.4.1 get	501
11.16.4.2 get_unshifted	502
11.16.4.3 set_dirty	502
11.16.4.4 set_unshifted_dirty	503
11.16.4.5 set	504
11.16.4.6 set_unshifted	505
11.16.4.7 val_dirty	506
11.16.4.8 val	507
-	507
11.17cxx::Bitmap < BITS > Class Template Reference	508
11.17.1 Detailed Description	509

xxxii CONTENTS

11.17.2 Constructor & Destructor Documentation	510
11.17.2.1 Bitmap	510
11.17.3 Member Function Documentation	510
11.17.3.1 clear_all	510
11.18cxx::Bitmap_base Class Reference	510
11.18.1 Detailed Description	513
11.18.2 Member Function Documentation	513
11.18.2.1 words	513
11.18.2.2 chars	513
11.18.2.3 bit	513
11.18.2.4 clear_bit	513
11.18.2.5 set_bit	514
11.18.2.6 operator[]	514
11.18.2.7 scan_zero	515
11.19L4::Bounds_error Class Reference	515
11.19.1 Detailed Description	517
11.20cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference	518
11.20.1 Detailed Description	522
11.20.2 Member Function Documentation	522
11.20.2.1 dir	522
11.20.2.2 dir	523
11.20.2.3 begin	523
11.20.2.4 end	524
11.20.2.5 begin	525
11.20.2.6 end	525
11.20.2.7 rbegin	525
11.20.2.8 rend	525
11.20.2.9 rbegin	526
11.20.2.10rend	526
11.20.2.11find_node	526
11.20.2.12ower_bound_node	527
11.20.2.13find	528
11.21cxx::Bits::Bst_node Class Reference	529
11.21.1 Detailed Description	532
$11.22L4:: Ipc:: Buf_cp_in < T > Class \ Template \ Reference \ \dots $	532
11.22.1 Detailed Description	532
11.22.2 Constructor & Destructor Documentation	532
11.22.2.1 Buf_cp_in	533
$11.23L4:: Ipc:: Buf_cp_out < T > Class \ Template \ Reference \\ \ \ldots \\ \ \ldots \\ \ \ldots$	534
11.23.1 Detailed Description	534

CONTENTS xxxiii

11.23.2 Constructor & Destructor Documentation	535
11.23.2.1 Buf_cp_out	535
11.23.3 Member Function Documentation	535
11.23.3.1 size	535
11.23.3.2 buf	535
$11.24 L4::lpc::Buf_in < T > Class \ Template \ Reference \ $	536
11.24.1 Detailed Description	536
11.24.2 Constructor & Destructor Documentation	537
11.24.2.1 Buf_in	537
11.25L4::Cap< T > Class Template Reference	537
11.25.1 Detailed Description	539
11.25.2 Constructor & Destructor Documentation	539
11.25.2.1 Cap	539
11.25.2.2 Cap	539
11.25.2.3 Cap	540
11.25.3 Member Function Documentation	541
11.25.3.1 move	541
11.26L4Re::Cap_alloc Class Reference	541
11.26.1 Detailed Description	542
11.26.2 Member Function Documentation	542
11.26.2.1 alloc	542
11.26.2.2 alloc	542
11.26.2.3 free	543
11.26.2.4 get_cap_alloc	543
11.27L4Re::Util::Cap_alloc_base Class Reference	544
11.27.1 Detailed Description	545
11.28L4::Cap_base Class Reference	545
11.28.1 Detailed Description	547
11.28.2 Member Enumeration Documentation	547
11.28.2.1 No_init_type	547
11.28.2.2 Cap_type	547
11.28.3 Constructor & Destructor Documentation	548
11.28.3.1 Cap_base	548
11.28.3.2 Cap_base	548
11.28.4 Member Function Documentation	548
11.28.4.1 cap	548
	549
11.28.4.3 fpage	550
	551
11.28.4.5 validate	552

CONTENTS

11.28.4.6 validate	552
11.28.5 Field Documentation	553
11.28.5.1 _c	553
11.29cxx::Bitmap_base::Char < BITS > Class Template Reference	553
11.29.1 Detailed Description	553
11.30L4Re::Video::Color_component Class Reference	553
11.30.1 Detailed Description	554
11.30.2 Constructor & Destructor Documentation	555
11.30.2.1 Color_component	555
11.30.3 Member Function Documentation	556
11.30.3.1 size	556
11.30.3.2 shift	556
11.30.3.3 operator==	556
11.30.3.4 get	556
11.30.3.5 set	556
11.30.3.6 dump	557
11.31L4::Com_error Class Reference	557
11.31.1 Detailed Description	560
11.31.2 Constructor & Destructor Documentation	560
11.31.2.1 Com_error	560
11.32L4::lpc_svr::Compound_reply Struct Reference	560
11.32.1 Detailed Description	561
11.33L4Re::Console Class Reference	561
11.33.1 Detailed Description	563
11.34L4Re::Util::Counting_cap_alloc< COUNTERTYPE > Class Template Reference	563
11.34.1 Detailed Description	564
11.35L4Re::Dataspace Class Reference	564
11 25 1 Detailed Description	567
11.35.1 Detailed Description	307
11.35.1 Detailed Description	
·	
11.35.2 Member Enumeration Documentation	567 567
11.35.2 Member Enumeration Documentation	567 567 568
11.35.2 Member Enumeration Documentation	567 567 568
11.35.2 Member Enumeration Documentation	567 567 568 568
11.35.2 Member Enumeration Documentation 11.35.2.1 Map_flags 11.35.3 Member Function Documentation 11.35.3.1 map 11.35.3.2 map_region	567 567 568 568 568
11.35.2 Member Enumeration Documentation 11.35.2.1 Map_flags 11.35.3 Member Function Documentation 11.35.3.1 map 11.35.3.2 map_region 11.35.3.3 clear	567 568 568 568 569
11.35.2 Member Enumeration Documentation 11.35.2.1 Map_flags 11.35.3 Member Function Documentation 11.35.3.1 map 11.35.3.2 map_region 11.35.3.3 clear 11.35.3.4 allocate	567 568 568 568 569 570
11.35.2 Member Enumeration Documentation 11.35.2.1 Map_flags 11.35.3 Member Function Documentation 11.35.3.1 map 11.35.3.2 map_region 11.35.3.3 clear 11.35.3.4 allocate 11.35.3.5 copy_in	567 568 568 568 569 570 570
11.35.2 Member Enumeration Documentation 11.35.2.1 Map_flags 11.35.3 Member Function Documentation 11.35.3.1 map 11.35.3.2 map_region 11.35.3.3 clear 11.35.3.4 allocate 11.35.3.5 copy_in 11.35.3.6 phys	567 568 568 568 569 570 571 572

CONTENTS XXXV

11.36L4Re::Util::Dataspace_svr Class Reference	574
11.36.1 Detailed Description	576
11.36.2 Member Function Documentation	576
11.36.2.1 map	576
11.36.2.2 map_hook	576
11.36.2.3 phys	577
11.36.2.4 take	577
11.36.2.5 release	577
11.36.2.6 copy	577
11.36.2.7 clear	578
11.36.2.8 allocate	578
11.36.2.9 page_shift	578
11.36.2.10s_static	579
11.37L4Re::Debug_obj Class Reference	579
11.37.1 Detailed Description	580
11.37.2 Member Function Documentation	581
11.37.2.1 debug	581
11.38L4::Debugger Class Reference	581
11.38.1 Detailed Description	584
11.38.2 Member Function Documentation	584
11.38.2.1 set_object_name	584
11.38.2.2 global_id	584
11.38.2.3 kobj_to_id	585
11.38.2.4 query_log_typeid	586
11.38.2.5 query_log_name	586
11.38.2.6 switch_log	586
11.38.2.7 get_object_name	587
11.39L4::lpc_svr::Default_loop_hooks Struct Reference	587
11.39.1 Detailed Description	588
11.40L4::lpc_svr::Default_setup_wait Struct Reference	588
11.40.1 Detailed Description	589
11.41L4::lpc_svr::Default_timeout Struct Reference	589
11.41.1 Detailed Description	590
11.42cxx::Bits::Direction Struct Reference	590
11.42.1 Detailed Description	592
11.42.2 Member Enumeration Documentation	592
11.42.2.1 Direction_e	592
11.42.3 Member Function Documentation	592
11.42.3.1 operator!	592
11.43L4Re::Vfs::Directory Class Reference	592

xxxvi CONTENTS

11.43.1 Detailed Description	594
11.43.2 Member Function Documentation	595
11.43.2.1 faccessat	595
11.43.2.2 mkdir	595
11.43.2.3 unlink	595
11.43.2.4 rename	595
11.43.2.5 link	596
11.43.2.6 symlink	596
11.43.2.7 rmdir	596
11.44L4::Element_already_exists Class Reference	597
11.44.1 Detailed Description	599
11.45L4::Element_not_found Class Reference	600
11.45.1 Detailed Description	602
11.46Elf32_Dyn Struct Reference	603
11.46.1 Detailed Description	603
11.46.2 Field Documentation	603
11.46.2.1 d_val	603
11.47Elf32_Ehdr Struct Reference	604
11.47.1 Detailed Description	605
11.47.2 Field Documentation	605
11.47.2.1 e_phnum	605
11.47.2.2 e_shnum	605
11.48Elf32_Phdr Struct Reference	605
11.48.1 Detailed Description	606
11.49Elf32_Shdr Struct Reference	607
11.49.1 Detailed Description	608
11.50Elf32_Sym Struct Reference	608
11.50.1 Detailed Description	609
11.51 Elf64_Dyn Struct Reference	609
11.51.1 Detailed Description	609
11.51.2 Field Documentation	609
11.51.2.1 d_val	609
11.52Elf64_Ehdr Struct Reference	610
11.52.1 Detailed Description	611
11.52.2 Field Documentation	611
11.52.2.1 e_phnum	611
11.52.2.2 e_shnum	611
11.53Elf64_Phdr Struct Reference	611
11.53.1 Detailed Description	612
11.54Elf64_Shdr Struct Reference	613

CONTENTS xxxvii

11.54.1 Detailed Description	314
11.55Elf64_Sym Struct Reference	314
11.55.1 Detailed Description	315
11.56L4Re::Env Class Reference	315
11.56.1 Detailed Description	317
11.56.2 Member Function Documentation	317
11.56.2.1 env	317
11.56.2.2 parent	317
11.56.2.3 mem_alloc	317
11.56.2.4 rm	318
11.56.2.5 log	318
11.56.2.6 main_thread	318
11.56.2.7 task	318
11.56.2.8 factory	318
11.56.2.9 first_free_cap	319
11.56.2.10utcb_area	319
11.56.2.11first_free_utcb	319
11.56.2.12nitial_caps	319
11.56.2.13get	319
11.56.2.14get_cap	320
11.56.2.15get_cap	320
11.56.2.1@parent	320
11.56.2.17mem_alloc	321
11.56.2.18m	321
11.56.2.19og	321
11.56.2.20main_thread	321
11.56.2.21factory	321
11.56.2.22first_free_cap	321
11.56.2.23utcb_area	322
11.56.2.24first_free_utcb	322
11.56.2.25scheduler	322
11.56.2.26scheduler	322
11.56.2.27/initial_caps	322
11.57L4Re::Event Class Reference	523
11.57.1 Detailed Description	626
11.57.2 Member Function Documentation	626
11.57.2.1 get_buffer	326
11.58L4Re::Event_buffer_t< PAYLOAD >::Event Struct Reference	626
11.58.1 Detailed Description	627
11.59L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference	627

xxxviii CONTENTS

11.59.1 Detailed Description	629
11.59.2 Member Function Documentation	630
11.59.2.1 foreach_available_event	630
11.59.2.2 process	630
11.60L4Re::Event_buffer_t< PAYLOAD > Class Template Reference	631
11.60.1 Detailed Description	633
11.60.2 Constructor & Destructor Documentation	633
11.60.2.1 Event_buffer_t	633
11.60.3 Member Function Documentation	634
11.60.3.1 next	634
11.60.3.2 put	634
11.61L4Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference	635
11.61.1 Detailed Description	636
11.61.2 Member Function Documentation	636
11.61.2.1 buf	636
11.61.2.2 attach	637
11.61.2.3 detach	637
11.62L4Re::Util::Event_t< PAYLOAD > Class Template Reference	637
11.62.1 Detailed Description	638
11.62.2 Member Enumeration Documentation	638
11.62.2.1 Mode	638
11.62.3 Member Function Documentation	639
11.62.3.1 init	639
11.62.3.2 buffer	639
11.62.3.3 irq	639
11.63L4::Exception_tracer Class Reference	640
11.63.1 Detailed Description	641
11.64L4::Factory Class Reference	641
11.64.1 Detailed Description	644
11.64.2 Member Function Documentation	644
11.64.2.1 create	644
11.64.2.2 create_task	645
11.64.2.3 create_thread	646
11.64.2.4 create_factory	647
11.64.2.5 create_gate	648
11.64.2.6 create_irq	649
11.64.2.7 create_vm	650
11.65L4Re::Vfs::File Class Reference	651
11.65.1 Detailed Description	653
11.66L4Re::Vfs::File_system Class Reference	653

CONTENTS xxxix

11.66.1 Detailed Description	655
11.66.2 Member Function Documentation	655
11.66.2.1 type	655
11.66.2.2 mount	655
11.67L4Re::Vfs::Fs Class Reference	656
11.67.1 Detailed Description	658
11.67.2 Member Function Documentation	659
11.67.2.1 get_file	659
11.67.2.2 alloc_fd	659
11.67.2.3 set_fd	659
11.67.2.4 free_fd	659
11.67.2.5 mount	660
11.68L4Re::Vfs::Generic_file Class Reference	660
11.68.1 Detailed Description	662
11.68.2 Member Function Documentation	662
11.68.2.1 unlock_all_locks	662
11.68.2.2 fstat64	663
11.68.2.3 fchmod	663
11.68.2.4 get_status_flags	663
11.68.2.5 set_status_flags	663
11.69gfxbitmap_offset Struct Reference	664
11.69.1 Detailed Description	664
11.70L4Re::Video::Goos Class Reference	
11.70L4neVideoGoos Class neletence	665
11.70.1 Detailed Description	
	667
11.70.1 Detailed Description	667 667
11.70.1 Detailed Description	667 667 667
11.70.1 Detailed Description	667 667 667
11.70.1 Detailed Description 11.70.2 Member Enumeration Documentation 11.70.2.1 Flags 11.70.3 Member Function Documentation	667 667 667 667
11.70.1 Detailed Description 11.70.2 Member Enumeration Documentation 11.70.2.1 Flags 11.70.3 Member Function Documentation 11.70.3.1 info	667 667 667 667 668
11.70.1 Detailed Description 11.70.2 Member Enumeration Documentation 11.70.2.1 Flags	667 667 667 667 668 668
11.70.1 Detailed Description 11.70.2 Member Enumeration Documentation 11.70.2.1 Flags 11.70.3 Member Function Documentation 11.70.3.1 info 11.70.3.2 get_static_buffer 11.70.3.3 create_buffer	667 667 667 667 668 668
11.70.1 Detailed Description 11.70.2 Member Enumeration Documentation 11.70.2.1 Flags 11.70.3 Member Function Documentation 11.70.3.1 info 11.70.3.2 get_static_buffer 11.70.3.3 create_buffer 11.70.3.4 delete_buffer	667 667 667 667 668 668 668
11.70.1 Detailed Description 11.70.2 Member Enumeration Documentation 11.70.2.1 Flags 11.70.3 Member Function Documentation 11.70.3.1 info 11.70.3.2 get_static_buffer 11.70.3.3 create_buffer 11.70.3.4 delete_buffer 11.70.3.5 create_view	667 667 667 668 668 668 668
11.70.1 Detailed Description 11.70.2 Member Enumeration Documentation 11.70.2.1 Flags 11.70.3 Member Function Documentation 11.70.3.1 info 11.70.3.2 get_static_buffer 11.70.3.3 create_buffer 11.70.3.4 delete_buffer 11.70.3.5 create_view 11.70.3.6 delete_view	667 667 667 667 668 668 668 669
11.70.1 Detailed Description 11.70.2 Member Enumeration Documentation 11.70.2.1 Flags 11.70.3 Member Function Documentation 11.70.3.1 info 11.70.3.2 get_static_buffer 11.70.3.3 create_buffer 11.70.3.4 delete_buffer 11.70.3.5 create_view 11.70.3.6 delete_view 11.70.3.7 view	667 667 667 667 668 668 668 669 669
11.70.1 Detailed Description 11.70.2 Member Enumeration Documentation 11.70.2.1 Flags	667 667 667 668 668 668 669 669 669
11.70.1 Detailed Description 11.70.2 Member Enumeration Documentation 11.70.2.1 Flags 11.70.3 Member Function Documentation 11.70.3.1 info 11.70.3.2 get_static_buffer 11.70.3.3 create_buffer 11.70.3.4 delete_buffer 11.70.3.5 create_view 11.70.3.6 delete_view 11.70.3.7 view 11.71L4Re::Util::Video::Goos_svr Class Reference 11.71.1 Detailed Description	667 667 667 668 668 668 669 669 671 671
11.70.1 Detailed Description 11.70.2 Member Enumeration Documentation 11.70.2.1 Flags 11.70.3 Member Function Documentation 11.70.3.1 info 11.70.3.2 get_static_buffer 11.70.3.3 create_buffer 11.70.3.4 delete_buffer 11.70.3.5 create_view 11.70.3.6 delete_view 11.70.3.7 view 11.71.1 Detailed Description 11.71.2 Member Function Documentation	667 667 667 668 668 668 669 669 671 671

xI CONTENTS

11.71.2.4 refresh	672
11.71.2.5 dispatch	672
11.71.2.6 init_infos	673
11.72L4::lcu Class Reference	673
11.72.1 Detailed Description	676
11.72.2 Member Function Documentation	676
11.72.2.1 bind	676
11.72.2.2 unbind	677
11.72.2.3 info	678
11.72.2.4 msi_info	679
11.72.2.5 mask	680
11.72.2.6 unmask	681
11.72.2.7 set_mode	681
11.73L4::lpc_svr::lgnore_errors Struct Reference	682
11.73.1 Detailed Description	683
11.74L4Re::Video::Goos::Info Struct Reference	683
11.74.1 Detailed Description	685
11.74.2 Member Function Documentation	685
11.74.2.1 auto_refresh	685
11.75L4Re::Video::View::Info Struct Reference	685
11.75.1 Detailed Description	687
11.75.2 Field Documentation	687
11.75.2.1 flags	687
11.76L4::lcu::lnfo Class Reference	688
11.76.1 Detailed Description	689
11.77L4::Invalid_capability Class Reference	689
11.77.1 Detailed Description	691
11.77.2 Constructor & Destructor Documentation	692
11.77.2.1 Invalid_capability	692
11.77.3 Member Function Documentation	692
11.77.3.1 cap	692
11.78L4::IOModifier Class Reference	692
11.78.1 Detailed Description	692
11.79L4::lpc::lostream Class Reference	693
11.79.1 Detailed Description	696
11.79.2 Constructor & Destructor Documentation	696
11.79.2.1 lostream	696
11.79.3 Member Function Documentation	696
11.79.3.1 reset	696
11.79.3.2 call	697

CONTENTS xli

11.79.3.3 reply_and_wait	698
11.79.3.4 reply_and_wait	698
11.80L4::lpc_gate Class Reference	699
11.80.1 Detailed Description	701
11.80.2 Member Function Documentation	702
11.80.2.1 bind_thread	702
11.80.2.2 get_infos	702
11.81L4::lrq Class Reference	702
11.81.1 Detailed Description	705
11.81.2 Member Function Documentation	705
11.81.2.1 attach	705
11.81.2.2 chain	706
11.81.2.3 detach	707
11.81.2.4 receive	707
11.81.2.5 wait	708
11.81.2.6 unmask	709
11.81.2.7 trigger	709
11.82L4::lpc::lstream Class Reference	710
11.82.1 Detailed Description	713
11.82.2 Constructor & Destructor Documentation	713
11.82.2.1 lstream	713
11.82.3 Member Function Documentation	714
11.82.3.1 reset	714
11.82.3.2 get	714
11.82.3.3 skip	715
11.82.3.4 get	715
11.82.3.5 get	715
11.82.3.6 tag	715
11.82.3.7 tag	716
11.82.3.8 wait	716
11.82.3.9 wait	717
11.82.3.10 receive	717
11.83L4Re::Util::Item_alloc_base Class Reference	718
11.83.1 Detailed Description	718
11.84cxx::List_item::Iter Class Reference	718
11.84.1 Detailed Description	720
11.84.2 Member Function Documentation	720
11.84.2.1 remove_me	720
11.85cxx::List< D, Alloc >::Iter Class Reference	721
11.85.1 Detailed Description	721

XIII CONTENTS

11.86L4::Kobject Class Reference	722
11.86.1 Detailed Description	723
11.86.2 Member Function Documentation	723
11.86.2.1 cap	723
11.86.2.2 dec_refcnt	723
11.86.3 Friends And Related Function Documentation	724
11.86.3.1 kobject_typeid	724
11.87L4::Kobject_2t< Derived, Base1, Base2, PROTO > Class Template Reference	724
11.87.1 Detailed Description	726
11.87.2 Friends And Related Function Documentation	726
11.87.2.1 kobject_typeid	726
11.88L4::Kobject_t< Derived, Base, PROTO > Class Template Reference	726
11.88.1 Detailed Description	727
11.88.2 Friends And Related Function Documentation	728
11.88.2.1 kobject_typeid	728
11.89I4_buf_regs_t Struct Reference	728
11.89.1 Detailed Description	729
11.90I4_exc_regs_t Struct Reference	729
11.90.1 Detailed Description	731
11.90.2 Field Documentation	731
11.90.2.1 flags	731
11.91I4_fpage_t Union Reference	731
11.91.1 Detailed Description	732
11.92I4_icu_info_t Struct Reference	732
11.92.1 Detailed Description	734
11.92.2 Field Documentation	734
11.92.2.1 features	734
11.93I4_kernel_info_mem_desc_t Struct Reference	734
11.93.1 Detailed Description	734
11.94I4_kernel_info_t Struct Reference	735
11.94.1 Detailed Description	737
11.95I4_msg_regs_t Union Reference	737
11.95.1 Detailed Description	737
11.96I4_msgtag_t Struct Reference	738
11.96.1 Detailed Description	739
11.96.2 Member Function Documentation	739
11.96.2.1 flags	739
11.97l4_sched_cpu_set_t Struct Reference	740
11.97.1 Detailed Description	740
11.98I4_sched_param_t Struct Reference	740

CONTENTS xliii

11.98.1 Detailed Description	741
11.99I4_snd_fpage_t Struct Reference	741
11.99.1 Detailed Description	742
11.10@4_thread_regs_t Struct Reference	742
11.100. Detailed Description	743
11.1014_timeout_s Struct Reference	744
11.101. Detailed Description	744
11.1024_timeout_t Union Reference	744
11.102. Detailed Description	745
11.1084_tracebuffer_status_t Struct Reference	745
11.103. Detailed Description	748
11.103. Field Documentation	748
11.103.2.1tracebuffer0	748
11.103.2.2size0	748
11.103.2.3version0	748
11.103.2.4tracebuffer1	748
11.103.2.5size1	748
11.103.2.6version1	748
11.103.2.7cnt_iobmap_tlb_flush	749
11.10#4_tracebuffer_status_window_t Struct Reference	749
11.104. Detailed Description	749
11.1054_vcon_attr_t Struct Reference	750
11.105. Detailed Description	750
11.1064_vcpu_ipc_regs_t Struct Reference	750
11.106. Detailed Description	751
11.1014_vcpu_regs_t Struct Reference	751
11.107. Detailed Description	753
11.107. Field Documentation	753
11.107.2.1di	753
11.107.2.2si	753
11.107.2.3bp	754
11.107.2.4bx	754
11.107.2.5dx	754
11.107.2.6cx	754
11.107.2.7ax	754
11.1084_vcpu_state_t Struct Reference	754
11.108. Detailed Description	757
11.1094_vhw_descriptor Struct Reference	757
11.109. Detailed Description	758
11.109. Field Documentation	759

XIIV CONTENTS

11.109.2.1magic	59
11.109.2.2version	59
11.109.2.3count	59
11.109.2.4descs	59
11.11@4_vhw_entry Struct Reference	59
11.110. Detailed Description	30
11.110.2Field Documentation	30
11.110.2.1type	30
11.110.2.2provider_pid	31
11.110.2.3mem_start	31
11.110.2.4mem_size	31
11.110.2.5rq_no	31
11.110.2.6fd	31
11.11 M_vm_svm_vmcb_control_area Struct Reference	32
11.111. Detailed Description	32
11.1124_vm_svm_vmcb_state_save_area Struct Reference	32
11.112. Detailed Description	33
11.1184_vm_svm_vmcb_state_save_area_seg Struct Reference	34
11.113. Detailed Description	34
11.11##_vm_svm_vmcb_t Struct Reference	34
11.114. Detailed Description	35
11.1154_vm_tz_state Struct Reference	36
11.115. Detailed Description	36
11.1164re_aux_t Struct Reference	36
11.116. Detailed Description	37
11.11 Are_ds_stats_t Struct Reference	37
11.117. Detailed Description	38
11.1184re_elf_aux_mword_t Struct Reference	36
11.118. Detailed Description	39
11.1194re_elf_aux_t Struct Reference	39
11.119. Detailed Description	39
11.1204re_elf_aux_vma_t Struct Reference	39
11.120. Detailed Description	
11.12lure_env_cap_entry_t Struct Reference	70
11.121. Detailed Description	
11.121. Constructor & Destructor Documentation	
11.121.2.1l4re_env_cap_entry_t	
11.121. Field Documentation	
11.121.3.1flags	71
11.1224re_env_t Struct Reference	72

CONTENTS xiv

11.122. Detailed Description
11.1284re_event_t Struct Reference
11.123. Detailed Description
11.12#re_video_color_component_t Struct Reference
11.124. Detailed Description
11.1264re_video_goos_info_t Struct Reference
11.125. Detailed Description
11.12@4re_video_pixel_info_t Struct Reference
11.126. Detailed Description
11.1214re_video_view_info_t Struct Reference
11.127. Detailed Description
11.1284re_video_view_t Struct Reference
11.128. Detailed Description
11.1294util_idt_desc_t Struct Reference
11.129. Detailed Description
11.13@4util_idt_header_t Struct Reference
11.130. Detailed Description
11.13 Mutil_mb_addr_range_t Struct Reference
11.131. Detailed Description
11.1324util_mb_apm_t Struct Reference
11.132. Detailed Description
11.1384util_mb_drive_t Struct Reference
11.133. Detailed Description
11.133. Field Documentation
11.133.2.1drive_number
11.133.2.2drive_mode
11.133.2.3drive_cylinders
11.13#util_mb_info_t Struct Reference
11.134. Detailed Description
11.13 Hutil_mb_mod_t Struct Reference
11.135. Detailed Description
11.135. Field Documentation
11.135.2.1mod_start
11.135.2.2mod_end
11.13@Hutil_mb_vbe_ctrl_t Struct Reference
11.136. Detailed Description
11.13 Hutil_mb_vbe_mode_t Struct Reference
11.137. Detailed Description
11.138xx::List< D, Alloc > Class Template Reference
11.138. Detailed Description

XIVI

11.138.2Member Function Documentation	792
11.138.2.1push_back	792
11.138.2.2push_front	792
11.138.2.3remove	792
11.138.2.4size	792
11.138.2.5operator[]	792
11.138.2.6operator[]	792
11.138.2.7items	793
11.13@xx::List_alloc Class Reference	793
11.139. Detailed Description	793
11.139. Constructor & Destructor Documentation	793
11.139.2.1List_alloc	794
11.139.3Member Function Documentation	794
11.139.3.1free	794
11.139.3.2alloc	794
11.139.3.3avail	794
11.140xx::List_item Class Reference	794
11.140. Detailed Description	796
11.140.2Member Function Documentation	796
11.140.2.1get_prev_item	796
11.140.2.2get_next_item	796
11.140.2.3nsert_prev_item	796
11.140.2.4insert_next_item	796
11.140.2.5remove_me	796
11.140.2.6push_back	797
11.140.2.7push_front	797
11.140.2.8remove	797
11.14L4Re::Log Class Reference	798
11.141. Detailed Description	801
11.141.2Member Function Documentation	801
11.141.2.1printn	801
11.141.2.2print	801
11.1424::Factory::Lstr Struct Reference	801
11.142. Detailed Description	802
$\textbf{11.148} \textbf{xx::} \textbf{Lt_functor} < \textbf{Obj} > \textbf{Struct Template Reference} \dots \dots$	802
11.143. Detailed Description	802
11.1444Re::Mem_alloc Class Reference	802
11.144. Detailed Description	805
11.144.2Member Enumeration Documentation	805
11.144.2.1Mem_alloc_flags	805

CONTENTS xIvii

11.144.3Member Function Documentation	805
11.144.3.1alloc	805
11.144.3.2free	806
11.1454::Kip::Mem_desc Class Reference	806
11.145. Detailed Description	808
11.145. Constructor & Destructor Documentation	808
11.145.2.1Mem_desc	808
11.145.3Member Function Documentation	808
11.145.3.1first	808
11.145.3.2count	808
11.145.3.3count	809
11.145.3.4start	809
11.145.3.5end	810
11.145.3.6size	810
11.145.3.7type	811
11.145.3.8sub_type	811
11.145.3.9s_virtual	811
11.145.3.1 s @et	811
11.1464::Meta Class Reference	812
11.146. Detailed Description	813
11.146.2Member Function Documentation	814
11.146.2.1num_interfaces	814
11.146.2.2interface	814
11.146.2.3supports	815
11.14ℤ4Re::Vfs::Mman Class Reference	815
11.147. Detailed Description	817
11.1484::Thread::Modify_senders Class Reference	817
11.148. Detailed Description	818
11.148.2Member Function Documentation	818
11.148.2.1add	
11.1484::lpc::Msg_ptr< T > Class Template Reference	
11.149. Detailed Description	819
11.149. Constructor & Destructor Documentation	819
11.149.2.1Msg_ptr	819
11.15 0 4Re::Util::Names::Name Class Reference	819
11.150. Detailed Description	820
11.15L4Re::Namespace Class Reference	820
11.151. Detailed Description	823
11.151.2Member Enumeration Documentation	
11.151.2.1Register_flags	823

xlviii CONTENTS

11.151.3Member Function Documentation	823
11.151.3.1query	823
11.151.3.2query	824
11.151.3.3register_obj	825
11.152xx::New_allocator< _Type > Class Template Reference	826
11.152. Detailed Description	826
11.15&4::Factory::Nil Struct Reference	826
11.153. Detailed Description	827
11.15 4 xx::AvI_set< Item, Compare, Alloc >::Node Class Reference	827
11.154. Detailed Description	828
11.154.2Member Function Documentation	828
11.154.2.1valid	828
11.15 5 xx::Nothrow Class Reference	828
11.155. Detailed Description	829
11.1564Re::Vfs::Ops Class Reference	829
11.156. Detailed Description	830
11.15E4::lpc::Ostream Class Reference	830
11.157. Detailed Description	833
11.157.2Member Function Documentation	833
11.157.2.1put	833
11.157.2.2put	834
11.157.2.3tag	834
11.157.2.4tag	834
11.157.2.5send	835
11.15&4::Out_of_memory Class Reference	835
11.158. Detailed Description	838
11.15@xx::Pair< First, Second > Struct Template Reference	838
11.159. Detailed Description	839
11.159. Constructor & Destructor Documentation	839
11.159.2.1Pair	839
$11.16 \mbox{\bf 0} xx:: \mbox{Pair_first_compare} < \mbox{Cmp, Typ} > \mbox{Class Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	839
11.160. Detailed Description	840
11.160. Constructor & Destructor Documentation	840
11.160.2.1Pair_first_compare	840
11.160.3Member Function Documentation	840
11.160.3.1operator()	840
11.16L4Re::Parent Class Reference	840
11.161. Detailed Description	842
11.161.2Member Function Documentation	843
11.161.2.1signal	843

CONTENTS xlix

11.1624Re::Video::Pixel_info Class Reference	843
11.162. Detailed Description	844
11.162. 2Constructor & Destructor Documentation	844
11.162.2.1Pixel_info	844
11.162.2.2Pixel_info	845
11.162.3Member Function Documentation	845
11.162.3.1r	845
11.162.3.2g	845
11.162.3.36	845
11.162.3.4a	846
11.162.3.5bytes_per_pixel	846
11.162.3.6bits_per_pixel	846
11.162.3.7has_alpha	846
11.162.3.8	846
11.162.3.9g	846
11.162.3.110	847
11.162.3.1a	847
11.162.3.1/20/tes_per_pixel	847
11.162.3.1@perator==	847
11.162.3.1dump	847
11.16&4Re::Util::Ref_cap< T > Struct Template Reference	848
11.163. Detailed Description	848
11.1644Re::Util::Ref_del_cap< T > Struct Template Reference	849
11.164. Detailed Description	849
11.1654Re::Vfs::Regular_file Class Reference	850
11.165. Detailed Description	852
11.165.2Member Function Documentation	853
11.165.2.1data_space	
44.40F.0.0vvvd.	853
11.165.2.2readv	
11.165.2.3writev	853
	853 853
11.165.2.3writev	853 853 853
11.165.2.3writev	853 853 853 853
11.165.2.3writev 11.165.2.4seek64 11.165.2.5truncate64	853 853 853 853 854
11.165.2.3writev 11.165.2.4seek64 11.165.2.5truncate64 11.165.2.6sync	853 853 853 853 854 854
11.165.2.3writev	853 853 853 854 854 854
11.165.2.3writev	853 853 853 854 854 854 854
11.165.2.3writev	853 853 853 854 854 854 854 855
11.165.2.3writev	853 853 853 854 854 854 854 855 857

CONTENTS

11.166.2.2Region_flags	858
11.166.2.3Attach_flags	858
11.166.2.4Detach_flags	858
11.166.3Member Function Documentation	858
11.166.3.1reserve_area	858
11.166.3.2reserve_area	859
11.166.3.3free_area	860
11.166.3.4attach	861
11.166.3.5attach	862
11.166.3.6detach	862
11.166.3.7detach	863
11.166.3.8detach	864
11.166.3.9find	864
11.16E4::Runtime_error Class Reference	866
11.167. Detailed Description	867
11.1684::Factory::S Class Reference	868
11.168. Detailed Description	869
11.168. Constructor & Destructor Documentation	869
11.168.2.1S	869
11.168.3Member Function Documentation	869
11.168.3.1operator I4_msgtag_t	869
11.168.3.2perator<<	869
11.168.3.3operator <<	869
11.168.3.4operator <<	870
11.168.3.5perator<<	871
11.168.3.6operator <<	871
11.1694::Scheduler Class Reference	871
11.169. Detailed Description	874
11.169.2Member Function Documentation	874
11.169.2.1info	874
11.169.2.2run_thread	875
11.169.2.3dle_time	875
11.169.2.4is_online	876
11.1704::Server< LOOP_HOOKS > Class Template Reference	877
11.170. Detailed Description	879
11.170. Constructor & Destructor Documentation	879
11.170.2.1Server	879
11.170.3Member Function Documentation	879
11.170.3.1internal_loop	
11.17L4::Server_object Class Reference	880

CONTENTS

11.171. Detailed Description	880
11.171.2Member Function Documentation	880
11.171.2.1dispatch	880
11.172xx::Slab < Type, Slab_size, Max_free, Alloc > Class Template Reference	881
11.172. Detailed Description	884
11.172.2Member Function Documentation	884
11.172.2.1alloc	884
11.172.2.2free	884
$\textbf{11.178} \textbf{2xx::Slab_static} < \textbf{Type}, \textbf{Slab_size}, \textbf{Max_free}, \textbf{Alloc} > \textbf{Class Template Reference} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	884
11.173. Detailed Description	886
11.173.2Member Function Documentation	888
11.173.2.1alloc	888
11.17&4::lpc::Small_buf Class Reference	888
11.174. Detailed Description	888
11.1754::Smart_cap< T, SMART > Class Template Reference	889
11.175. Detailed Description	891
11.175. Constructor & Destructor Documentation	891
11.175.2.1Smart_cap	891
11.1764Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference	891
11.176. Detailed Description	892
11.17E4Re::Smart_cap_auto< Unmap_flags > Class Template Reference	892
11.177. Detailed Description	892
11.17&4Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference	893
11.178. Detailed Description	893
11.1794Re::Vfs::Special_file Class Reference	893
11.179. Detailed Description	895
11.179.2Member Function Documentation	895
11.179.2.1ioctl	895
11.1804vcpu::State Class Reference	895
11.180. Detailed Description	896
11.180.2Constructor & Destructor Documentation	896
11.180.2.1State	896
11.180.3Member Function Documentation	896
11.180.3.1add	896
11.180.3.2clear	897
11.180.3.3set	897
11.18L4Re::Dataspace::Stats Struct Reference	897
11.181. Detailed Description	898
11.1824::String Class Reference	898
11.182. Detailed Description	898

lii CONTENTS

11.188xx::List_item::T_iter< T, Poly > Class Template Reference
11.183. Detailed Description
11.18 4 4::Task Class Reference
11.184. Detailed Description
11.184.2Member Function Documentation
11.184.2.1map
11.184.2.2unmap
11.184.2.3unmap_batch
11.184.2.4delete_obj
11.184.2.5release_cap
11.184.2.6cap_valid
11.184.2.7cap_has_child
11.184.2.≈_equal
11.184.2.9add_ku_mem
11.18 5 4::Thread Class Reference
11.185. Detailed Description
11.185.2Member Function Documentation
11.185.2.1ex_regs
11.185.2.2ex_regs
11.185.2.3control
11.185.2.4switch_to
11.185.2.5stats_time
11.185.2.6vcpu_resume_start
11.185.2.7vcpu_resume_commit
11.185.2.8vcpu_control
11.185.2.9vcpu_control_ext
11.185.2.1@gister_del_irq
11.185.2.1rhodify_senders
11.18 6 4::Type_info Struct Reference
11.186. Detailed Description
11.18E4::Unknown_error Class Reference
11.187. Detailed Description
11.188xx::Bitfield< T, LSB, MSB >::Value< TT > Class Template Reference
11.188. Detailed Description
11.18@xx::Bitfield< T, LSB, MSB >::Value_base< TT > Class Template Reference
11.189. Detailed Description
11.196xx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Class Template Reference 927
11.190. Detailed Description
11.19 L4::Vcon Class Reference
11.191. Detailed Description

CONTENTS

11.191.2Member Function Documentation	931
11.191.2.1send	931
11.191.2.2write	932
11.191.2.3read	933
11.191.2.4set_attr	934
11.191.2.5get_attr	934
11.1924Re::Util::Vcon_svr< SVR > Class Template Reference	935
11.192. Detailed Description	936
11.192.2Member Function Documentation	936
11.192.2.1dispatch	936
11.1984vcpu::Vcpu Class Reference	937
11.193. Detailed Description	941
11.193.2Member Function Documentation	941
11.193.2.1irq_disable_save	941
11.193.2.2state	941
11.193.2.3state	941
11.193.2.4saved_state	941
11.193.2.5saved_state	942
11.193.2.6rq_enable	942
11.193.2.7/rq_restore	942
11.193.2.8wait_for_event	943
11.193.2.9task	943
11.193.2.1i6_page_fault_entry	944
11.193.2.1is_irq_entry	944
11.193.2.1/2	944
11.193.2.1/3	945
11.193.2.1i4	945
11.193.2.1i5	945
11.193.2.1@ntry_sp	945
11.193.2.1 @ ntry_ip	946
11.193.2.1@xt_alloc	947
11.193.2.1 9 ast	947
11.193.2.2@ast	947
11.1944Re::Video::View Class Reference	948
11.194. Detailed Description	949
11.194.2Member Enumeration Documentation	949
11.194.2.1Flags	949
11.194.2.2V_flags	949
11.194.3Member Function Documentation	
11.194.3.1info	950

liv CONTENTS

	11.194.3.2set_info	950
	11.194.3.3set_viewport	950
	11.194.3.4stack	950
	11.194.3.5refresh	951
	11.1954::Vm Class Reference	952
	11.195. Detailed Description	954
	11.196xx::Bitmap_base::Word< BITS > Class Template Reference	955
	11.196. Detailed Description	955
10	Evernale Decumentation	957
12	Example Documentation	
	12.1 examples/clntsrv/client.cc	
	12.2 examples/clntsrv/clntsrv.cfg	
	12.3 examples/clntsrv/server.cc	
	12.4 examples/libs/l4re/c++/mem_alloc/ma+rm.cc	
	12.5 examples/libs/l4re/c++/shared_ds/ds_clnt.cc	
	12.7 examples/libs/l4re/c++/shared_ds/shared_ds.lua	964
	12.8 examples/libs/l4re/c/ma+rm.c	964
	12.9 examples/libs/l4re/streammap/client.cc	966
	12.10examples/libs/l4re/streammap/server.cc	967
	12.11 examples/libs/l4re/streammap/streammap.cfg	968
	12.12examples/libs/libirq/async_isr.c	968
	12.13examples/libs/libirq/loop.c	969
	12.14examples/libs/shmc/prodcons.c	969
	12.15examples/sys/aliens/main.c	971
	12.16examples/sys/ipc/ipc.cfg	973
	12.17examples/sys/ipc/ipc_example.c	973
	12.18examples/sys/isr/main.c	974
	12.19examples/sys/migrate/thread_migrate.cc	975
	12.20examples/sys/migrate/thread_migrate.cfg	977
	12.21 examples/sys/singlestep/main.c	977
	12.22examples/sys/start-with-exc/main.c	979
	12.23examples/sys/utcb-ipc/main.c	981
	12.24examples/sys/ux-vhw/main.c	982
	12.25hello/server/src/main.c	983
	12.26tmpfs/lib/src/fs.cc	983

Index

990

Chapter 1

Fiasco.OC & L4 Runtime Environment (L4Re)

1.1 Preface

The intention of this document is to provide a birds eye overview about L4Re and about the environment in which typical applications and servers run. We highlight here the principled functionality of the servers in the environment but do not discuss their specific interfaces. Detailed documentation about these interface is available in the modules section.

The document is meant as a general overview repeating many design concepts of L4-based systems and capability systems in general. We do though assume familiarity with C++ and an idea on the general concepts and terms of L4: threads — as an abstraction for execution —, tasks — holding the capabilities to kernel objects that are accessible by the threads executing in this task —, and IPC over IPC-gates to send messages and to transfer capabilities between tasks.

1.2 General 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.

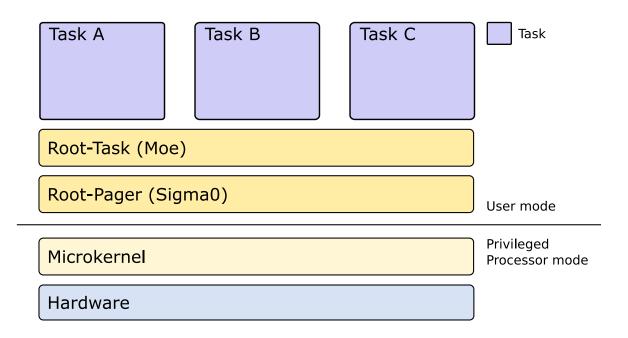


Figure 1.1: Basic Structure of an L4Re based system

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.

In the following sections, we discuss the basic concepts of our microkernel and its runtime environment in more depth.

1.3 The Fiasco.OC Microkernel

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 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.

1.3.1 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.

1.3.2 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 userlevel services.
- · Scheduler Implements scheduling policy and assignment of threads to CPUs, including CPU statistics.

1.4 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.

1.5 Introduction to L4Re's concepts

This section introduces basic concepts used by L4Re. Understanding of these concepts is a fundamental requirement to understand the inner workings of L4Re's software components and can dramatically help developers in efficiently developing L4Re-based software.

1.6 Memory management - Data Spaces and the Region Map

1.6.1 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.

1.6.2 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.

1.6.3 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.

1.6.4 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

Data-Space API and Region map API.

1.7 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.

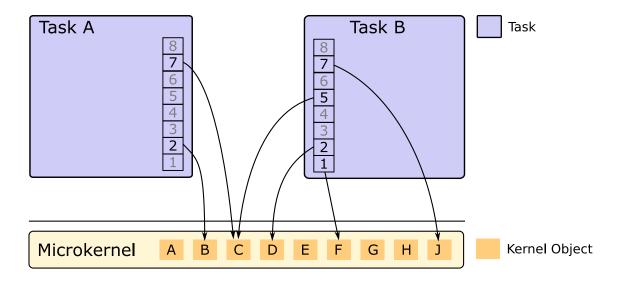


Figure 1.2: Capabilities and Local Naming in L4

So how does an application get access to 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.

1.8 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.

1.8.1 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");
```

1.8.2 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:full() }}, "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 <code>loader:new_channel()</code>. The capability to the gate is stored in the variable <code>svc</code>. Then the binary <code>my_server</code> is started in a new task, and full (:full()) 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:full() } }, "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.

1.9 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

1.10 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

Memory allocator API

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

1.11 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.

1.11.1 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.

1.11.2 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.

Chapter 2

Getting Started

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 14re
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 I4/conf/examples.

```
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.

10 Getting Started

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 grubliso 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.

Customizations

A basic set of bootable entries can be found in 14/conf/modules.list. This file is the default for any image creation as shown above. It is recommeded 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 14 directory.

Chapter 3

L4Re Servers

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

3.1 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.

3.2 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 an 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, this 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.

3.3 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.

12 L4Re Servers

For more information see Ned.

3.4 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 lo, the lo Server.

3.5 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.

3.6 fb-drv, the Low-Level Graphics Driver

The fb-drv server 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.

3.7 Rtc, the Real-Time Clock Server

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

3.8 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 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 (L4Re::Namespace) 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 (L4Re::Log) provides tagged log output for applications
- l4re_moe_scheduler (L4::Scheduler) provides simple scheduler objects for scheduling policy enforcement
- Memory Allocator, Generic Factory (L4Re::Mem_alloc, L4::Factory) provides allocation of physical RAM
 as data spaces, as well as allocation of the other L4Re objects provided by Moe

3.8 Moe, the Root-Task

3.8.1 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 gerneic factory interface alllows allocation of the following objects:

- L4Re::Namespace
- L4Re::Dataspace, RAM allocation
- L4Re::Rm, Virtual mamory 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). The granularity for memory allocation is the machine page size (L4_PAGESIZE).

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)

3.8.2 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.

3.8.3 Boot FS

The Boot FS subsystem provides read only access to the files loaded during the platform boot (or available in ROM). This 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.

3.8.4 Log Subsystem

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

3.8.5 Command-Line Options

Moe command-line syntax is:

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

3.8.5.1 --debug=<debug flags>

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

14 L4Re Servers

3.8.5.2 --init=<init process>

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

Note

command-line options to the init process are given after the - special option.

```
3.8.5.3 --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 sam as for --debug=.

```
3.8.5.4 --Idr-flags=<loader flags>
```

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

3.9 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.

3.9.1 Lua Bindings for L4Re

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

3.9.1.1 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.

3.9.1.2 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.

3.9.1.3 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 = 0x4005,
    Irq = -1,
    Sigma0 = -6,
    Log = -13,
    Scheduler = -14,
    Factory = -15,
    Ipc_gate = 0,
```

Debugging Flags

Debugging flags used for the applications L4Re core:

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
}
```

3.9.1.4 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

16 L4Re Servers

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.

3.10 lo, the lo Server

The lo 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 lo 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 statically.

Each lo 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.

3.10 lo, the lo Server

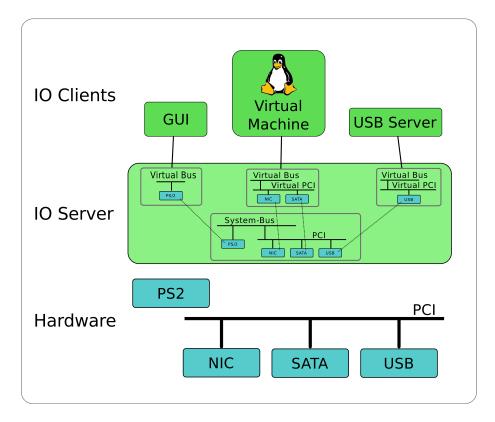


Figure 3.1: IO Service Architecture Overview

The lo 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 lo through the command line.

The platform configuration is stored in the structure called hw-root. It lists devices that are available on the platform. For the x86 architecture a basic set of platform devices is defined in the file x86-legacy.devs. There are configuration files for various ARM platforms available, as well. If the system contains a PCI bus, it is scanned automatically and the devices found on it are added automatically to the pool of available devices.

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

A very simple configuration for lo could look like this:

```
-- Example configuration for io

-- Configure two platform devices to be known to io
Io.hw_add_devices
{
   FOODEVICE = Io.Hw.Device
   {
      hid = "FOODEVICE";
      Io.Res.irq(17);
      Io.Res.mmio(0x6f000000, 0x6f007fff);
   },

BARDEVICE = Io.Hw.Device
   {
      hid = "BARDEVICE";
      Io.Res.irq(19);
      Io.Res.irq(20);
      Io.Res.irq(20);
      Io.Res.mmio(0x6f100000, 0x6f100fff);
   }
}

Io.add_vbusses
{
   -- Create a virtual bus for a client and give access to FOODEVICE
```

18 L4Re Servers

```
client1 = Io.Vi.System_bus(function ()
    dev = wrap(hw:match("FOODEVICE"));
end),

-- Create a virtual bus for another client and give it access to BARDEVICE
    client2 = Io.Vi.System_bus(function ()
        dev = wrap(hw:match("BARDEVICE"));
end)
}
```

Assigning clients PCI devices could look like this:

```
-- This is a configuration snippet for PCI device selection
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)
end)
```

The CC numbers are PCI class codes. You can also use REV_, VEN_, DEV_ and SUBSYS_ to specify revision, vendor, device and subsystem with a hex number.

Chapter 4

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-I4.h header file:

```
#include <pthread-14.h>
```

· Return the local thread capability of a pthread thread:

Use pthread_get14cap (pthread_t t) to get the capability index of the pthread t.

For example:

```
pthread_get14cap(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 = 14_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);
```

20 Pthread Support

Chapter 5

Module Index

5.1 Modules

Here	is a	list of	all	modu	ıles

1	se API	123
	Basic Macros	129
	Cache Consistency	173
	Capabilities	284
	Error codes	189
	Fiasco extensions	132
	Fiasco real time scheduling extensions	140
	Kernel Debugger	180
	Flex pages	141
	Integer Types	437
	Kernel Interface Page	232
	Fiasco-UX Virtual devices	307
	Memory descriptors (C version)	237
	Kernel Objects	230
	Factory	191
	IPC-Gate API	126
	IRQs	224
	Interrupt controller	198
	Scheduler	241
	Task	246
	Thread	255
	Thread control	267
	vCPU API	305
	Virtual Console	300
	Virtual Machines	197
	VM API for SVM	161
	VM API for TZ	312
	VM API for VMX	162
	Memory operations	308
	Memory related	175
	Object Invocation	205
	Error Handling	218
	Message Items	150
	Message Tag	274
	Realtime API	223
	Timeouts	153
	Virtual Registers (UTCRs)	

22 Module Index

ARM Virtual Registers (UTCB)
Buffer Registers (BRs)
Message Registers (MRs)
Exception registers
Thread Control Registers (TCRs)
amd64 Virtual Registers (UTCB)
x86 Virtual Registers (UTCB)
Client/Server IPC Framework
IPC Messaging Framework
IPC Streams
IRQ handling library
Interface for asynchronous ISR handlers
Interface for asynchronous ISR handlers with a given IRQ capability.
Interface using direct functionality
Interface using direct functionality
L4Re C Interface
Capability allocator
Dataspace interface
Debug interface
Event interface
Kumem allocator utility
L4Re Util C Interface
Log interface
Memory allocator
Namespace interface
Region map interface
Video API
L4Re C++ Interface
Auxiliary data
C++ Exceptions
Console API
Data-Space API
Debugging API
Event API
Goos video API
L4Re ELF Auxiliary Information
L4Re Protocol identifiers
L4Re Util C++ Interface
Kumem utilitities
L4Re Capability API
Logging interface
Memory allocator API
Name-space API
Parent API
Region map API
Shared Memory Library
Chunks
Consumer
Producer
Signals
Consumer
Producer
Sigma0 API
Internal constants

5.1 Modules 23

Small C++ Template Library
Utility Functions
Atomic Instructions
Bit Manipulation
Bitmap graphics and fonts
Functions for rendering bitmap data in frame buffers
Functions for rendering bitmap fonts to frame buffers
CPU related functions
Comfortable Command Line Parsing
ELF binary format
Functions to manipulate the local IDT
IA32 Port I/O API
Internal functions
Kernel Interface Page API
Low-Level Thread Functions
Machine Restarting Function
Priority related functions
Random number support
Timestamp Counter
vCPU Support Library
Extended vCPU support

24 **Module Index**

Chapter 6

Namespace Index

6.1 Namespace List

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

CXX		
	Our C++ library	4
cxx::Bits		
	Internal helpers for the cxx package	42
L4		
	L4 low-level kernel interface	42
L4::lpc_s	svr	
	Helper classes for L4::Server instantiation	4
L4Re		
	L4 Runtime Environment	46
L4Re::Vf	's	
	Virtual file system for interfaces POSIX libc	4

26 Namespace Index

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

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

L4::Alloc_list	449
	449
– '	453
= '	453
'	455
	455
-	456
- - · · · · ·	466 466
— · · · · · · · · · · · · · · · · · · ·	460
cxx::Avl_set< Pair< Region, Hdlr >, Pair_first_compare< cxx::Lt_functor< Region >, Pair< Region, Hdlr	
	466
cxx::AvI_map< Region, HdIr, cxx::Lt_functor, Alloc >	
— — — · — · — ·	483
	483
cxx::Slab< Type, Slab_size, Max_free, Alloc >	881
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >	486
cxx::Base_slab_static< sizeof(Type), Slab_size, Max_free, Alloc >	486
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >	884
L4::Basic_registry	490
cxx::Bitfield< T, LSB, MSB >	498
cxx::Bitmap_base	510
$cxx:: Bitmap < BITS > \dots $	508
cxx::Bits::Bst < Node, Get_key, Compare >	518
cxx::Avl_tree< Node, Get_key, Compare >	475
cxx::Bits::Bst< _Node, Get_key, Compare >	518
cxx::Avl_tree< _Node, Get_key, Compare >	475
	518
cxx::Avl_tree< _Node, Get_key, Pair_first_compare< Compare< Key >, Pair< Key, Data > >	475
cxx::Bits::Bst< _Node, Get_key, Pair_first_compare< cxx::Lt_functor< Region >, Pair< Region, Hdlr >	
	518
cxx::Avl_tree< _Node, Get_key, Pair_first_compare< cxx::Lt_functor< Region >, Pair< Region, Hdlr	
>>>	475
	518
	475

28 Hierarchical Index

cxx::Bits::Bst_node	529
cxx::Avl_tree_node	178
L4::lpc::Buf_cp_in< T >	532
·	534
L4::lpc::Buf_in< T >	
L4Re::Cap_alloc	
L4Re::Util::Cap_alloc_base	
L4::Cap_base	
L4::Cap< L4::Factory >	
L4::Cap< L4::Irq >	
L4::Cap< L4::Thread >	
L4::Cap< L4::Vcon >	
L4::Cap < L4Re::Dataspace >	
L4::Cap< L4Re::L4Re::Dataspace >	
L4::Cap< L4Re::L4Re::Namespace >	
L4::Cap< L4Re::Rm >	537
L4::Cap< L4Re::Video::Goos >	
L4::Cap< void >	
L4::Cap< T >	
L4::Smart_cap< T, SMART >	
cxx::Bitmap_base::Char< BITS >	
L4Re::Video::Color_component	
L4::lpc_svr::Compound_reply	
L4::lpc_svr::Default_loop_hooks	
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >	
L4Re::Util::Dataspace_svr	
L4::lpc_svr::Default_setup_wait	
L4::lpc_svr::Default_loop_hooks	
L4::lpc_svr::Default_timeout	
L4::lpc_svr::Default_loop_hooks	
cxx::Bits::Direction	
L4Re::Vfs::Directory	
L4Re::Vfs::File	
L4Re::Vfs::Be_file	
Elf32_Dyn	
	304
	305 307
-	30 <i>1</i> 308
— <i>•</i>	509
— ·	310
Elf64_Phdr	311
Elf64_Shdr	313
Elf64_Sym	314
	315
	326
L4Re::Event_buffer_t< PAYLOAD >	
L4Re::Util::Event_buffer_t< PAYLOAD >	
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >	
	337
	340
L4::Base_exception	
L4::Invalid_capability	
L4::Runtime_error	
L4::Bounds_error)15

7.1 Class Hierarchy 29

L4::Com_error	
L4::Element_already_exists	
L4::Element_not_found	
L4::Out_of_memory	
L4::Unknown_error	
L4Re::Vfs::File_system	
L4Re::Vfs::Be_file_system	. 496
L4Re::Vfs::Fs	656
L4Re::Vfs::Ops	. 829
L4Re::Vfs::Generic_file	660
L4Re::Vfs::File	. 651
gfxbitmap offset	664
L4Re::Util::Video::Goos_svr	
L4::lpc_svr::lgnore_errors	682
L4::lpc_svr::Default_loop_hooks	. 587
L4Re::Video::Goos::Info	683
L4Re::Video::View::Info	685
L4::IOModifier	692
L4::lpc::lstream	710
L4::lpc::lostream	. 693
L4Re::Util::Item_alloc_base	718
cxx::List_item::lter	718
cxx::List_item::T_iter< T, Poly >	. 898
$cxx::List_item::T_iter < E > \dots \dots$. 898
cxx::List< D, Alloc >::Iter	721
L4::Kobject	722
L4::Kobject_t< Dataspace, L4::Kobject, L4Re::Protocol::Dataspace >	. 726
L4Re::Dataspace	. 564
L4::Kobject_t< Debug_obj, L4::Kobject, Protocol::Debug >	. 726
L4Re::Debug_obj	. 579
L4::Kobject_t< Debugger, Kobject, L4_PROTO_DEBUGGER >	. 726
L4::Debugger	. 581
L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >	
L4::Factory	
lem:lem:lem:lem:lem:lem:lem:lem:lem:lem:	. 726
L4Re::Video::Goos	
lem:lem:lem:lem:lem:lem:lem:lem:lem:lem:	
L4::Ipc_gate	
$\label{eq:lambda} \text{L4::Kobject_t} < \text{Irq_eio, Kobject, L4_PROTO_IRQ} > \dots $	
L4::Kobject_t< Mem_alloc, L4::Kobject, L4Re::Protocol::Mem_alloc >	
L4Re::Mem_alloc	
L4::Kobject_t< Meta, Kobject, L4_PROTO_META >	
L4::Meta	
L4::Kobject_t< Namespace, L4::Kobject, L4Re::Protocol::Namespace >	
L4Re::Namespace	
L4::Kobject_t< Parent, L4::Kobject, L4Re::Protocol::Parent >	
L4Re::Parent	
L4::Kobject_t< Rm, L4::Kobject, L4Re::Protocol::Rm >	
L4Re::Rm	
L4::Kobject_t< Task, Kobject, L4_PROTO_TASK >	
L4::Task	
L4::Kobject_t< Vm, Task, L4_PROTO_VM >	
L4::Vm	
$\label{eq:L4::Kobject_t} \textbf{L4::Kobject_t} < \textbf{Thread, Kobject, L4_PROTO_THREAD} > \dots $. /26

30 Hierarchical Index

L4::Thread	. 910
L4::Kobject 2t< Derived, Base1, Base2, PROTO >	724
L4::Kobject_2t< Console, Video::Goos, Event >	724
L4Re::Console	
L4::Kobject_t< Derived, Base, PROTO >	
L4::Kobject_t< Icu, Irq_eio, L4_PROTO_IRQ >	
L4::lcu	
L4::Kobject_t< Event, L4::Icu, L4Re::Protocol::Event >	
L4Re::Event	
L4::Kobject_t< Scheduler, Icu, L4_PROTO_SCHEDULER >	
L4::Scheduler	. 871
L4::Kobject_t< Vcon, Icu, L4_PROTO_LOG >	. 726
L4::Vcon	. 928
L4::Kobject_t< Log, L4::Vcon, L4_PROTO_LOG >	726
L4Re::Log	798
L4::Kobject_t< Irq, Irq_eio, L4_PROTO_IRQ >	
L4::lrg	
·	
l4_buf_regs_t	
I4_exc_regs_t	729
I4_fpage_t	731
I4_icu_info_t	
L4::lcu::lnfo	. 688
$\label{lambda} \mbox{l4_kernel_info_mem_desc_t} \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	734
l4_kernel_info_t	735
l4_msg_regs_t	737
I4_msgtag_t	738
$I4_sched_cpu_set_t \ \dots $	740
l4_sched_param_t	740
l4_snd_fpage_t	741
l4_thread_regs_t	742
l4_timeout_s	744
l4_timeout_t	744
I4_tracebuffer_status_t	745
I4_tracebuffer_status_window_t	749 750
	750 750
I4_vcpu_ipc_regs_t	750 751
14_vcpu_state_t	751
L4vcpu::Vcpu	
I4_vhw_descriptor	757
l4_vhw_entry	759
I4_vm_svm_vmcb_control_area	762
I4_vm_svm_vmcb_state_save_area	762
I4_vm_svm_vmcb_state_save_area_seg	764
I4_vm_svm_vmcb_t	764
I4_vm_tz_state	766
l4re_aux_t	766
l4re_ds_stats_t	767 768
	769
ldre_elf_aux_t	769
I4re_elf_aux_vma_t	769
lare env t	772
l4re event t	773
l4re_video_color_component_t	774
l4re_video_goos_info_t	775

7.1 Class Hierarchy 31

l4re_video_pixel_info_t
l4re_video_view_info_t
14re_video_view_t
l4util_mb_addr_range_t
4util_mb_apm_t
4util_mb_mod_t
cxx::List< D, Alloc >
cxx::List_alloc
cxx::List_item
L4::Factory::Lstr
cxx::Lt_functor< Obj >
L4::Kip::Mem_desc
L4Re::Vfs::Mman
L4Re::Vfs::Ops
L4::Thread::Modify_senders
L4::lpc::Msg_ptr< T >
L4Re::Util::Names::Name
cxx::New_allocator< _Type >
L4::Factory::Nil
cxx::Avl_set< Item, Compare, Alloc >::Node
cxx::Nothrow
L4::lpc::Ostream
L4::lpc::lostream
cxx::Pair < First, Second >
cxx::Pair_first_compare< Cmp, Typ >
L4Re::Video::Pixel_info
L4Re::Util::Ref_cap< T >
L4Re::Util::Ref_del_cap< T >
L4Re::Vfs::Regular_file
L4Re::Vfs::File
L4::Factory::S
L4::Server < LOOP_HOOKS >
L4::Server object
L4::lpc::Small_buf
L4Re::Util::Smart_cap_auto< Unmap_flags >
L4Re::Smart_cap_auto< Unmap_flags >
L4Re::Util::Smart_count_cap< Unmap_flags >
L4Re::Vfs::Special_file
L4Re::Vfs::File
L4vcpu::State
L4Re::Dataspace::Stats
·
L4::String
L4::Type_info
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >
cxx::Bitfield< T, LSB, MSB >::Value< TT >
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >
L4Re::Util::Vcon_svr< SVR >
L4Re::Video::View
cxx::Bitmap_base::Word< BITS >
cxx::Bitmap_base::Word< Bits >
cxx::Bitmap_base::Word < Size >

32 Hierarchical Index

Chapter 8

Data Structure Index

8.1 Data Structures

Here are the data structures with brief descriptions:

L4::Alloc_list	
A simple list-based allocator	449
L4::Thread::Attr	
Thread attributes used for control_commit()	449
L4Re::Util::Auto_cap< T >	
Automatic capability that implements automatic free and unmap of the capability selector	453
L4Re::Util::Auto_del_cap< T >	
Automatic capability that implements automatic free and unmap+delete of the capability selector	455
cxx::Auto_ptr< T >	
Smart pointer with automatic deletion	456
cxx::Avl_map< Key, Data, Compare, Alloc >	
AVL tree based associative container	460
cxx::Avl_set< Item, Compare, Alloc >	
AVL Tree for simple comapreable items	466
cxx::Avl_tree< Node, Get_key, Compare >	
A generic AVL tree	475
cxx::Avl_tree_node	
Node of an AVL tree	478
L4::Base_exception	
Base class for all exceptions, thrown by the L4Re framework	481
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >	
Basic slab allocator	483
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >	
Merged slab allocator (allocators for objects of the same size are merged together)	486
L4::Basic_registry	
This registry returns the corresponding server object based on the label of an lpc_gate	490
L4Re::Vfs::Be_file	
Boiler plate class for implementing an open file for L4Re::Vfs	492
L4Re::Vfs::Be_file_system	
Boilerplate class for implementing a L4Re::Vfs::File_system	496
cxx::Bitfield< T, LSB, MSB >	
Definition for a member (part) of a bit field	498
cxx::Bitmap< BITS >	
A static bit map	508
cxx::Bitmap_base	
Basic bitmap abstraction	510
L4::Bounds_error	
Access out of bounds	515

34 Data Structure Index

cxx::Bits::Bst< Node, Get_key, Compare >	
Basic binary search tree (BST)	518
cxx::Bits::Bst_node	
Basic type of a node in a binary search tree (BST)	529
L4::lpc::Buf_cp_in< T > Abstraction for extracting array from an lpc::lstream	532
L4::lpc::Buf cp out< T >	332
Abstraction for inserting an array into an Ipc::Ostream	534
L4::lpc::Buf_in< T >	
Abstraction to extract an array from an Ipc::Istream	536
L4::Cap < T >	F07
Capability Selector a la C++	537
Capability allocator interface	541
L4Re::Util::Cap_alloc_base	
Capability allocator	544
L4::Cap_base	
Base class for all kinds of capabilities	545
cxx::Bitmap_base::Char < BITS > Helper abstraction for a byte contained in the bitmap	553
L4Re::Video::Color_component	550
A color component	553
L4::Com_error	
Error conditions during IPC	557
L4::lpc_svr::Compound_reply	500
Mix in for LOOP_HOOKS to always use compound reply and wait	560
Console class	561
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >	
Reference-counting cap allocator	563
L4Re::Dataspace	
This class represents a data space	564
L4Re::Util::Dataspace_svr Dataspace server class	574
L4Re::Debug_obj	0, .
Debug interface	579
L4::Debugger	
Debugger interface	581
L4::lpc_svr::Default_loop_hooks Default LOOP_HOOKS	587
L4::lpc_svr::Default_setup_wait	307
Mix in for LOOP_HOOKS for setup_wait no op	588
L4::lpc_svr::Default_timeout	
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout	589
cxx::Bits::Direction	
The direction to go in a binary search tree	590
Interface for a POSIX file that is a directory	592
L4::Element_already_exists	-
Exception for duplicate element insertions	597
L4::Element_not_found	
Exception for a failed lookup (element not found)	600
Elf32_Dyn ELF32 dynamic entry	603
Elf32 Ehdr	003
ELF32 header	604
Elf32_Phdr	
ELF32 program header	605

8.1 Data Structures 35

Elf32_Shdr	
ELF32 section header - figure 1-9, page 1-9	607
Elf32_Sym ELF32 symbol table entry	608
Elf64_Dyn ELF64 dynamic entry	609
Elf64_Ehdr ELF64 header	610
Elf64_Phdr	
ELF64 program header	611
ELF64 section header	613
ELF64 symbol table entry	614
L4Re::Env Initial Environment (C++ version)	615
L4Re::Event	
Event class	623
Event structure used in buffer	626
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >	607
An event buffer consumer	627
Event buffer class	631
L4Re::Util::Event_buffer_t< PAYLOAD > Event_buffer utility class	635
L4Re::Util::Event_t< PAYLOAD > Convenience wrapper for getting access to an event object	637
L4::Exception_tracer	
Back-trace support for exceptions	640
L4::Factory C++ L4 Factory, to create all kinds of kernel objects	641
L4Re::Vfs::File The basic interface for an open POSIX file	651
L4Re::Vfs::File_system Basic interface for an L4Re::Vfs file system	653
L4Re::Vfs::Fs	
POSIX File-system related functionality	656
The common interface for an open POSIX file	660
gfxbitmap_offset Offsets in pmap[] and bmap[]	664
L4Re::Video::Goos	
A goos	665
Goos server class	669
C++ version of an interrupt controller	673
L4::lpc_svr::lgnore_errors Mix in for LOOP_HOOKS to ignore IPC errors	682
L4Re::Video::Goos::Info	002
Information structure of a goos	683
Information structure of a view	685
L4::lcu::lnfo Info for an ICU	688
L4::Invalid_capability Indicates that an invalid object was invoked	689
maloates that an invalid object was involved	003

36 Data Structure Index

L4::IOModifier	
Modifier class for the IO stream	692
L4::lpc::lostream	
Input/Output stream for IPC [un]marshalling	693
L4::lpc_gate L4 IPC gate	000
L4:irg	699
C++ version of an L4 IRQ	702
L4::lpc::lstream	
Input stream for IPC unmarshalling	710
L4Re::Util::Item_alloc_base	
Item allocator	718
cxx::List_item::Iter	
Iterator for a list of ListItem-s	718
cxx::List< D, Alloc >::Iter Iterator	721
L4::Kobject	121
Base class for all kinds of kernel objects, referred to by capabilities	722
L4::Kobject_2t< Derived, Base1, Base2, PROTO >	
Helper class to create an L4Re interface class that is derived from two base classes	724
L4::Kobject_t< Derived, Base, PROTO >	
Helper class to create an L4Re interface class that is derived from a single base class	726
I4_buf_regs_t	700
Encapsulation of the buffer-registers block in the UTCB	728
I4_exc_regs_t UTCB structure for exceptions	729
I4_fpage_t	. 20
L4 flexpage type	731
I4_icu_info_t	
Info structure for an ICU	732
I4_kernel_info_mem_desc_t	70.4
Memory descriptor data structure	734
I4_kernel_info_t L4 Kernel Interface Page	735
I4_msg_regs_t	, 00
Encapsulation of the message-register block in the UTCB	737
I4_msgtag_t	
Message tag data structure	738
I4_sched_cpu_set_t	-
CPU sets	740
I4_sched_param_t Scheduler parameter set	740
14 snd fpage t	740
Send-flex-page types	741
I4_thread_regs_t	
Encapsulation of the thread-control-register block of the UTCB	742
I4_timeout_s	
Basic timeout specification	744
I4_timeout_t Timeout pair	744
I4_tracebuffer_status_t	,
Trace buffer status	745
I4_tracebuffer_status_window_t	
Trace-buffer status window descriptor	749
I4_vcon_attr_t	
Vcon attribute structure	750
I4_vcpu_ipc_regs_t	750
VCPU message registers	750

8.1 Data Structures 37

4_vcpu_regs_t	751
4 vcpu state t	731
State of a vCPU	754
I4_vhw_descriptor	
Virtual hardware devices description	757
I4_vhw_entry	750
•	759
I4_vm_svm_vmcb_control_area VMCB structure for SVM VMs	762
14 vm svm vmcb state save area	702
State save area structure for SVM VMs	762
I4_vm_svm_vmcb_state_save_area_seg	
State save area segment selector struct	764
I4_vm_svm_vmcb_t	
Control structure for SVM VMs	764
I4_vm_tz_state State structure for TrustZone VMs	766
l4re aux t	700
Auxiliary descriptor	766
l4re_ds_stats_t	
Information about the data space	767
l4re_elf_aux_mword_t	
Auxiliary vector element for a single unsigned data word	768
l4re_elf_aux_t	700
Generic header for each auxiliary vector element	769
l4re_elf_aux_vma_t Auxiliary vector element for a reserved virtual memory area	769
l4re_env_cap_entry_t	700
Entry in the L4Re environment array for the named inital objects	770
l4re_env_t	
Initial Environment structure (C version)	772
l4re_event_t	
Event structure used in buffer	773
	774
l4re_video_goos_info_t	117
Goos information structure	775
l4re_video_pixel_info_t	
Pixel_info structure	776
l4re_video_view_info_t	
View information structure	777
View information structure	
View information structure	777 779
View information structure l4re_video_view_t C representation of a goos view	779
View information structure I4re_video_view_t C representation of a goos view I4util_idt_desc_t IDT entry	
View information structure l4re_video_view_t C representation of a goos view	779
View information structure I4re_video_view_t C representation of a goos view I4util_idt_desc_t IDT entry I4util_idt_header_t	779 780
View information structure I4re_video_view_t C representation of a goos view I4util_idt_desc_t IDT entry I4util_idt_header_t Header of an IDT table I4util_mb_addr_range_t INT-15, AX=E820 style "AddressRangeDescriptor"with a "size" parameter on the front which	779 780
View information structure l4re_video_view_t	779 780 780
View information structure I4re_video_view_t C representation of a goos view I4util_idt_desc_t IDT entry I4util_idt_header_t Header of an IDT table I4util_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	779 780
View information structure I4re_video_view_t C representation of a goos view I4util_idt_desc_t IDT entry I4util_idt_header_t Header of an IDT table I4util_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 I4util_mb_apm_t	779 780 780 781
View information structure I4re_video_view_t C representation of a goos view I4util_idt_desc_t IDT entry I4util_idt_header_t Header of an IDT table I4util_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 I4util_mb_apm_t APM BIOS info	779 780 780
View information structure I4re_video_view_t C representation of a goos view I4util_idt_desc_t IDT entry I4util_idt_header_t Header of an IDT table I4util_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 I4util_mb_apm_t APM BIOS info I4util_mb_drive_t	779 780 780 781
View information structure	779 780 780 781 782

38 Data Structure Index

14util mb vbe ctrl t	
VBE controller information	788
I4util_mb_vbe_mode_t	
VBE mode information	788
cxx::List< D, Alloc >	- 0.4
Doubly linked list, with internal allocation	791
cxx::List_alloc Standard list-based allocator	793
cxx::List_item	790
Basic list item	794
L4Re::Log	
Log interface class	798
L4::Factory::Lstr	801
Special type to add a pascal string into the factory create stream	001
Generic comparator class that defaults to the less-than operator	802
L4Re::Mem alloc	
Memory allocator	802
L4::Kip::Mem_desc	
Memory descriptors stored in the kernel interface page	806
L4::Meta	
Meta interface that shall be implemented by each L4Re object and gives access to the dynamic	010
type information for L4Re objects	812
Interface for the POSIX memory management	815
L4::Thread::Modify senders	0.0
Wrapper class for modifying senders	817
L4::lpc::Msg_ptr< T >	
Pointer to an element of type T in an lpc::lstream	818
L4Re::Util::Names::Name Name class	819
L4Re::Namespace	019
Name-space interface	820
cxx::New_allocator< _Type >	
Standard allocator based on operator new ()	826
L4::Factory::Nil	
Special type to add a void argument into the factory create stream	826
cxx::Avl_set< Item, Compare, Alloc >::Node	007
A smart pointer to a tree item	827
cxx::Nothrow Helper type to distinguish the opprator new version that does not throw exceptions	828
L4Re::Vfs::Ops	020
Interface for the POSIX backends for an application	829
L4::Ipc::Ostream	
Output stream for IPC marshalling	830
L4::Out_of_memory	
Exception signalling insufficient memory	835
cxx::Pair < First, Second >	000
Pair of two values	838
Comparison functor for Pair	839
L4Re::Parent	550
Parent interface	840
L4Re::Video::Pixel_info	
Pixel information	843
L4Re::Util::Ref_cap< T >	0.40
Automatic capability that implements automatic free and unmap of the capability selector	848

8.1 Data Structures 39

L4Re::Util::Ref_del_cap< T >	
Automatic capability that implements automatic free and unmap+delete of the capability selector	849
L4Re::Vfs::Regular_file	
Interface for a POSIX file that provides regular file semantics	850
L4Re::Rm Region map	855
L4::Runtime_error	000
Exception for an abstract runtime error	866
L4::Factory::S	
Stream class for the create() argument stream	868
L4::Scheduler	
Scheduler object	871
Basic server loop for handling client requests	877
L4::Server_object	0,,
Abstract server object to be used with L4::Server and L4::Basic_registry	880
cxx::Slab< Type, Slab_size, Max_free, Alloc >	
Slab allocator for object of type	881
cxx::Slab_static < Type, Slab_size, Max_free, Alloc > Merged slab allocator (allocators for objects of the same size are merged together)	884
L4::lpc::Small buf	004
A receive item for receiving a single capability	888
L4::Smart_cap< T, SMART >	
Smart capability class	889
L4Re::Util::Smart_cap_auto< Unmap_flags >	004
Helper for Auto_cap and Auto_del_cap	891
Helper for Auto_cap and Auto_del_cap	892
L4Re::Util::Smart_count_cap< Unmap_flags >	
Helper for Ref_cap and Ref_del_cap	893
L4Re::Vfs::Special_file	
Interface for a POSIX file that provides special file semantics	893
L4vcpu::State C++ implementation of state word in the vCPU area	895
L4Re::Dataspace::Stats	000
Information about the data space	897
L4::String	
A null-terminated string container class	898
cxx::List_item::T_iter< T, Poly > Iterator for derived classes from ListItem	000
L4::Task	090
An L4 Task	901
L4::Thread	
L4 kernel thread	910
L4::Type_info	
Dynamic Type Information for L4Re Interfaces	920
Exception for an unknown condition	921
cxx::Bitfield< T, LSB, MSB >::Value< TT >	-
Internal helper type	924
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >	
Internal helper type	925
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Internal helper type	927
L4::Voon	521
C++ L4 Vcon	928
L4Re::Util::Vcon_svr< SVR >	
Console server template class	935

40 Data Structure Index

L4vcpu:	:Vepu	
	C++ implementation of the vCPU save state area	937
L4Re::V	ideo::View	
	View	948
L4::Vm		
	Virtual machine	952
cxx::Bitn	nap_base::Word< BITS >	
	Helper abstraction for a word contained in the hitman	955

Chapter 9

Module Documentation

9.1 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).

42 Module Documentation

• 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.

9.1.1 Detailed Description

9.2 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::Avl_map< Key, Data, Compare, Alloc >

AVL tree based associative container.

class cxx::Avl_set< Item, Compare, Alloc >

AVL Tree for simple comapreable items.

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 <code>oeprator</code> <code>new version</code> 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.

44 Module Documentation

9.2.1 Detailed Description

9.2.2 Function Documentation

9.2.2.1 template<typename T1 > T1 cxx::min (T1 a, T1 b) [inline]

Get the minimum of a and b.

Parameters

а	the first value.
b	the second value.

Examples:

tmpfs/lib/src/fs.cc.

Definition at line 35 of file minmax.

Referenced by operator>>().

Here is the caller graph for this function:



9.2.2.2 template<typename T1 > T1 cxx::max (T1 a, T1 b) [inline]

Get the maximum of a and b.

Parameters

а	the first value.
b	the second value.

Definition at line 45 of file minmax.

9.2.2.3 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.

9.3 Client/Server IPC Framework

Data Structures

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.

class L4::Basic_registry

This registry returns the corresponding server object based on the label of an <code>lpc_gate</code>.

9.3.1 Detailed Description

46 Module Documentation

9.4 IPC Streams

Functions

L4::lpc::lstream & operator>> (L4::lpc::lstream &s, bool &v)

Extract one element of type T from the stream s.

L4::lpc::lstream & operator>> (L4::lpc::lstream &s, l4_msgtag_t &v)

Extract the L4 message tag from the stream s.

• template<typename T >

```
L4::lpc::lstream & operator>> (L4::lpc::lstream &s, L4::lpc::Buf_in< T > const &v)
```

Extract an array of T elements from the stream s.

• template<typename T >

```
L4::lpc::lstream & operator>> (L4::lpc::lstream &s, L4::lpc::Msg_ptr< T > const &v)
```

Extract an element of type T from the stream s.

template<typename T >

```
L4::lpc::lstream & operator>> (L4::lpc::lstream &s, L4::lpc::Buf_cp_in< T > const &v)
```

Extract an array of T elements from the stream s.

• L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, bool v)

Insert an element to type T into the stream s.

L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, l4_msgtag_t const &v)

Insert the L4 message tag into the stream s.

• template<typename T >

```
L4::lpc::Ostream & operator << (L4::lpc::Ostream &s, L4::lpc::Buf_cp_out < T > const &v)
```

Insert an array with elements of type T into the stream s.

• L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, char const *v)

Insert a zero terminated character string into the stream s.

9.4.1 Detailed Description

9.4.2 Function Documentation

```
9.4.2.1 L4::lpc::lstream& operator>>( L4::lpc::lstream & s, bool & v ) [inline]
```

Extract one element of type T from the stream s.

Parameters

S	The stream to extract from.
V	Output: extracted value.

9.4 IPC Streams 47

Returns

the stream s.

Definition at line 1237 of file ipc_stream.

References L4::lpc::lstream::get().

Here is the call graph for this function:



9.4.2.2 L4::lpc::lstream& operator>> (L4::lpc::lstream & s, I4_msgtag_t & v) [inline]

Extract the L4 message tag from the stream s.

Parameters

S	The stream to extract from.
V	Output: the extracted tag.

Returns

the stream s.

Definition at line 1270 of file ipc_stream.

References L4::lpc::lstream::tag().

Here is the call graph for this function:



9.4.2.3 template < typename T > L4::lpc::lstream & s, L4::lpc::Buf_in < T > const & ν) [inline]

Extract an array of T elements from 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.

See lpc::Buf_in, lpc::Buf_cp_in, and lpc::Buf_cp_out.

48 Module Documentation

Parameters

S	The stream to extract from.
V	Output: pointer to the extracted array (ipc_buf_in()).

Returns

the stream s.

Definition at line 1292 of file ipc stream.

References L4::lpc::lstream::get().

Here is the call graph for this function:



9.4.2.4 template<typename T > L4::lpc::lstream & s, L4::lpc::Msg_ptr< T > const & ν) [inline]

Extract an element of type T from 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.

Parameters

s	The stream to extract from.
V	Output: pointer to the extracted element.

Returns

the stream s.

Definition at line 1317 of file ipc_stream.

References L4::lpc::lstream::get().

Here is the call graph for this function:



9.4 IPC Streams 49

9.4.2.5 template<typename T > L4::lpc::lstream & s, L4::lpc::Buf_cp_in< T > const & v) [inline]

Extract an array of *T* elements from the stream *s*.

This operator does a copy out of the data into the given buffer.

See lpc::Buf_in, lpc::Buf_cp_in, and lpc::Buf_cp_out.

Parameters

S	The stream to extract from.
V	buffer description to copy the array to (lpc::Buf_cp_out()).

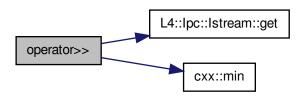
Returns

the stream s.

Definition at line 1338 of file ipc_stream.

References L4::lpc::lstream::get(), and cxx::min().

Here is the call graph for this function:



9.4.2.6 L4::lpc::Ostream& operator<<(L4::lpc::Ostream & s, bool v) [inline]

Insert an element to type *T* into the stream *s*.

Parameters

S	The stream to insert the element <i>v</i> .
V	The element to insert.

50 Module Documentation

Returns

the stream s.

Definition at line 1356 of file ipc_stream.

References L4::Ipc::Ostream::put().

Here is the call graph for this function:



9.4.2.7 L4::lpc::Ostream& operator<<(L4::lpc::Ostream & s, I4_msgtag_t const & v) [inline]

Insert the L4 message tag into the stream s.

Note

Only one message tag can be inserted into a stream. Multiple insertions simply overwrite previous insertions.

Parameters

S	The stream to insert the tag v .
V	The L4 message tag to insert.

Returns

the stream s.

Definition at line 1390 of file ipc_stream.

References L4::lpc::Ostream::tag().

Here is the call graph for this function:



9.4.2.8 template<typename T > L4::lpc::Ostream & operator<< (L4::lpc::Ostream & s, L4::lpc::Buf_cp_out< T > const & v) [inline]

Insert an array with elements of type T into the stream s.

9.4 IPC Streams 51

Parameters

S	The stream to insert the array <i>v</i> .
V	The array to insert (see lpc::Buf_cp_out()).

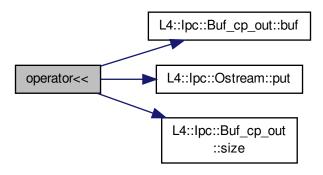
Returns

the stream s.

Definition at line 1406 of file ipc_stream.

 $References\ L4::lpc::Buf_cp_out < T > ::buf(),\ L4::lpc::Ostream::put(),\ and\ L4::lpc::Buf_cp_out < T > ::size().$

Here is the call graph for this function:



9.4.2.9 L4::lpc::Ostream& operator<<(L4::lpc::Ostream & s, char const * v) [inline]

Insert a zero terminated character string into the stream s.

Parameters

S	The stream to insert the string <i>v</i> .
V	The string to insert.

52 Module Documentation

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 1427 of file ipc_stream.

References L4::Ipc::Ostream::put().

Here is the call graph for this function:



9.5 IPC Messaging Framework

Data Structures

```
class L4::lpc::Buf_cp_out< T >
          Abstraction for inserting an array into an Ipc::Ostream.
    class L4::lpc::Buf_cp_in< T >
          Abstraction for extracting array from an Ipc::Istream.
    class L4::lpc::Msg_ptr< T >
          Pointer to an element of type T in an lpc::lstream.
    class L4::lpc::Buf in< T >
          Abstraction to extract an array from an Ipc::Istream.

    class L4::lpc::lstream

          Input stream for IPC unmarshalling.

    class L4::lpc::Ostream

          Output stream for IPC marshalling.
    • class L4::lpc::lostream
          Input/Output stream for IPC [un]marshalling.
Functions
    • template<typename T >
      Buf_cp_out < T > L4::lpc::buf_cp_out (T const *v, unsigned long size)
          Create an instance of Buf_cp_out for the given values.
    • template<typename T >
      Buf cp in < T > L4::lpc::buf cp in (T *v, unsigned long &size)
          Create an Buf_cp_in for the given values.
    • template<typename T >
      Msg_ptr < T > L4::lpc::msg_ptr (T *&p)
          Create an Msg_ptr to adjust the given pointer.
    template<typename T >
      Buf_in< T > L4::lpc::buf_in (T *&v, unsigned long &size)
          Create an Buf_in for the given values.
       Detailed Description
9.5.1
9.5.2 Function Documentation
9.5.2.1 template < typename T > Buf_cp_out < T > L4::lpc::buf_cp_out ( T const * v, unsigned long size )
Create an instance of Buf_cp_out for the given values.
This function makes it more convenient to insert arrays into an Ipc::Ostream (
See Also
```

Buf_cp_out.)

54 Module Documentation

Parameters

V	Pointer to the array that shall be inserted into an lpc::Ostream.
size	Number of elements in the array.

Definition at line 101 of file ipc stream.

9.5.2.2 template<typename T > Buf_cp_in<T> L4::lpc::buf_cp_in (T * v, unsigned long & size)

Create an Buf_cp_in for the given values.

This function makes it more convenient to extract arrays from an lpc::|stream (

See Also

Buf_cp_in.)

Parameters

V	Pointer to the array that shall receive the values from the lpc::lstream.
size	Input: the number of elements the array can take at most
	Output: the number of elements found in the stream.

See Also

buf_in() and buf_cp_out().

Definition at line 152 of file ipc_stream.

9.5.2.3 template<typename T > Msg_ptr<T> L4::lpc::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 lpc::lstream. This may be used to avoid copy out of large data structures. (See Msg_ptr.)

Definition at line 194 of file ipc_stream.

9.5.2.4 template<typename $T > Buf_in < T > L4::lpc::buf_in (<math>T * \& v$, unsigned long & size)

Create an Buf_in for the given values.

This function makes it more convenient to extract arrays from an lpc::lstream (See Buf_in.)

Parameters

V	Output: pointer to the array within the lpc::lstream.
size	Output: the number of elements found in the stream.

See buf_cp_in() and buf_cp_out().

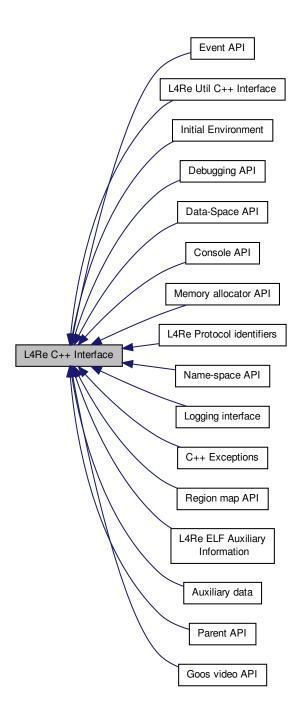
Definition at line 244 of file ipc_stream.

9.6 L4Re C++ Interface 55

9.6 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.

• Data-Space API

Data-Space API.

Debugging API

Debugging Interface.

Event API

Event interface.

- Goos video API
- Initial Environment

Environment that is initially provided to an L4 task.

L4Re ELF Auxiliary Information

API for embedding auxiliary information into binary programs.

· L4Re Protocol identifiers

Basic protocol identifiers used for L4Re.

• L4Re Util C++ Interface

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

· Logging interface

Interface for log output.

· Memory allocator API

Memory-allocator interface.

Name-space API

API for name spaces that store capabilities.

Parent API

Parent interface.

Region map API

Virtual address-space management.

9.6.1 Detailed Description

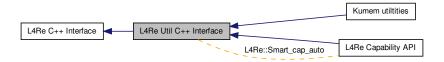
Documentation of the L4 Runtime Environment C++ API.

9.7 L4Re Util C++ Interface 57

9.7 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::Counting_cap_alloc< COUNTERTYPE >

Reference-counting cap allocator.

· class L4Re::Util::Dataspace_svr

Dataspace server class.

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.

9.7.1 Detailed Description

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

9.8 Dataspace interface

Dataspace C interface.

Collaboration diagram for Dataspace interface:



Data Structures

• struct I4re_ds_stats_t

Information about the data space.

Typedefs

• typedef I4_cap_idx_t I4re_ds_t

Dataspace type.

• typedef I4_cap_idx_t I4re_namespace_t

Dataspace type.

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
- 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_NO-THROW

Return physical address.

9.8.1 Detailed Description

Dataspace C interface.

9.8.2 Function Documentation

9.8.2.1 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
9.8.2.2 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
9.8.2.3 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
9.8.2.4 long l4re_ds_size ( const l4re_ds_t ds )
Returns
      size of dataspace, <0 on errors
See Also
      L4Re::Dataspace::size
9.8.2.5 long l4re_ds_flags ( const l4re_ds_t ds )
See Also
      L4Re::Dataspace::flags
9.8.2.6 int l4re_ds_info ( const l4re_ds_t ds, l4re_ds_stats_t * stats )
See Also
      L4Re::Dataspace::info
9.8.2.7 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 <i>phys_addr</i> (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

9.9 Debug interface 61

9.9 Debug interface

Collaboration diagram for Debug interface:



Functions

• void l4re_debug_obj_debug (l4_cap_idx_t srv, unsigned long function) L4_NOTHROW Call debug function of L4Re service.

9.9.1 Detailed Description

9.9.2 Function Documentation

9.9.2.1 void l4re_debug_obj_debug (I4_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

9.10 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_NO-THROW

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 *axis, l4re_event_absinfo_t *info) L4_NOTHROW

Get Axis information for a stream.

9.10.1 Detailed Description

Event C interface.

9.10.2 Function Documentation

9.10.2.1 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

9.10 Event interface 63

9.10.2.2 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

9.10.2.3 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

9.10.2.4 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

9.10.2.5 long l4re_event_get_axis_info (const l4_cap_idx_t server, l4_umword_t id, unsigned naxes, unsigned * axis, l4re_event_absinfo_t * info)

Get Axis information for a stream.

9.10 Event interface 65

Parameters

server	Server to talk to.	
naxes	Number of axes.	

Return values

axis	Number of axes.
info	Information buffer.

Returns

0 for success, <0 on error

See Also

L4Re::Event::get_axis_info

9.11 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.

9.11.1 Detailed Description

Log C interface.

9.11.2 Function Documentation

```
9.11.2.1 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

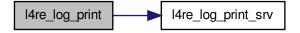
L4Re::Log::print

Definition at line 99 of file log.h.

References I4re_log_print_srv(), and I4re_env_t::log.

9.11 Log interface 67

Here is the call graph for this function:



9.11.2.2 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 I4re_log_printn_srv(), and I4re_env_t::log.

Here is the call graph for this function:



9.11.2.3 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 I4re_log_print().

Here is the caller graph for this function:



9.11.2.4 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

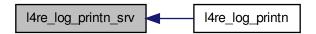
0 for success, <0 on error

See Also

L4Re::Log::printn

Referenced by I4re_log_printn().

Here is the caller graph for this function:



9.12 Memory allocator 69

9.12 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 (unsigned long size, l4re_ds_t const mem, unsigned long flags) L4_NOTHROW
 Allocate memory.
- long l4re_ma_alloc_align (unsigned long size, l4re_ds_t const mem, unsigned long flags, unsigned long align)
 L4_NOTHROW

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, unsigned 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.

9.12.1 Detailed Description

Memory allocator C interface.

9.12.2 Enumeration Type Documentation

9.12.2.1 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.

9.12.3 Function Documentation

9.12.3.1 long l4re_ma_alloc (unsigned 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 153 of file mem_alloc.h.

References I4re_ma_alloc_align_srv(), and I4re_env_t::mem_alloc.

Here is the call graph for this function:



9.12.3.2 long l4re_ma_alloc_align (unsigned 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 161 of file mem_alloc.h.

References I4re_ma_alloc_align_srv(), and I4re_env_t::mem_alloc.

Here is the call graph for this function:



9.12.3.3 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.

Examples:

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

Definition at line 169 of file mem_alloc.h.

References I4re_ma_free_srv(), and I4re_env_t::mem_alloc.

9.12 Memory allocator 73

Here is the call graph for this function:



9.12.3.4 long l4re_ma_alloc_align_srv (I4_cap_idx_t srv, unsigned long size, I4re_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

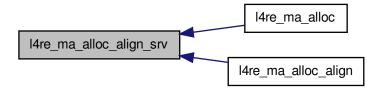
See Also

L4Re::Mem_alloc::alloc

The memory allocator returns a dataspace.

Referenced by I4re_ma_alloc(), and I4re_ma_alloc_align().

Here is the caller graph for this function:



9.12.3.5 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

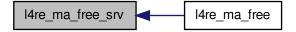
0 on success, <0 on error

See Also

L4Re::Mem_alloc::free

Referenced by I4re_ma_free().

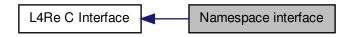
Here is the caller graph for this function:



9.13 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
- long l4re_ns_register_obj_srv (l4re_namespace_t srv, char const *name, l4_cap_idx_t const obj, unsigned flags) L4_NOTHROW

9.13.1 Detailed Description

Namespace C interface.

9.13.2 Enumeration Type Documentation

9.13.2.1 enum l4re_ns_register_flags

Namespace register flags.

See Also

L4Re::Namespace::Register_flags

Definition at line 39 of file namespace.h.

9.13.3 Function Documentation

 $9.13.3.1 \quad long \ l4re_ns_query_to_srv \ (\ l4re_namespace_t \ srv, \ char \ const * \textit{name}, \ l4_cap_idx_t \ const \ \textit{cap}, \ int \ \textit{timeout} \)$

Returns

0 on success, <0 on error

See Also

L4Re::Namespace::query

9.13.3.2 long l4re_ns_register_obj_srv (l4re_namespace_t srv, char const * name, l4_cap_idx_t const obj, unsigned flags)

Returns

0 on success, <0 on error

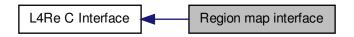
See Also

L4Re::Namespace::register_obj

9.14 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_NO-THROW
- 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.

9.14.1 Detailed Description

Region map C interface.

9.14.2 Enumeration Type Documentation

9.14.2.1 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.

9.14.3 Function Documentation

9.14.3.1 int l4re_rm_reserve_area (I4_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 I4re_rm_reserve_area_srv(), and I4re_env_t::rm.

Here is the call graph for this function:



9.14.3.2 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 I4re_rm_free_area_srv(), and I4re_env_t::rm.

Here is the call graph for this function:



9.14.3.3 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

start	Virtual start address
size	Size of the data space to attach (in bytes)
flags	Flags, see #Attach_flags and #Region_flags
mem	Data space
offs	Offset into the data space to use
align	Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT), Only meaningful
	if the #Search_addr flag is used.

Return values

start	Start of region if #Search_addr was used.

Returns

0 on success, <0 on error

- -L4_ENOENT
- -L4 EPERM
- -L4_EINVAL
- -L4_EADDRNOTAVAIL
- · 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 #find).

Returns

0 on success, <0 on error

See Also

L4Re::Rm::attach

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

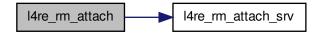
Examples:

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

Definition at line 243 of file rm.h.

References I4re_rm_attach_srv(), and I4re_env_t::rm.

Here is the call graph for this function:



```
9.14.3.4 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

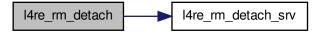
L4Re::Rm::detach

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

Definition at line 253 of file rm.h.

References L4_BASE_TASK_CAP, l4re_rm_detach_srv(), and l4re_env_t::rm.

Here is the call graph for this function:



9.14.3.5 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 L4_BASE_TASK_CAP, l4re_rm_detach_srv(), and l4re_env_t::rm.

Here is the call graph for this function:



9.14.3.6 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

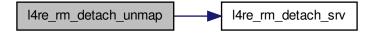
L4Re::Rm::detach

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

Definition at line 260 of file rm.h.

References I4re_rm_detach_srv(), and I4re_env_t::rm.

Here is the call graph for this function:



9.14.3.7 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.
auui	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 I4re_rm_detach_srv(), and I4re_env_t::rm.

Here is the call graph for this function:



9.14.3.8 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

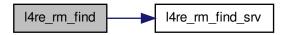
See Also

L4Re::Rm::find

Definition at line 280 of file rm.h.

References I4re_rm_find_srv(), and I4re_env_t::rm.

Here is the call graph for this function:



9.14.3.9 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 I4re_rm_show_lists_srv(), and I4re_env_t::rm.

Here is the call graph for this function:



9.14.3.10 int l4re_rm_reserve_area_srv (I4_cap_idx_t rm, I4_addr_t * start, unsigned long size, unsigned flags, unsigned char align)

See Also

L4Re::Rm::reserve_area

Referenced by I4re_rm_reserve_area().

Here is the caller graph for this function:



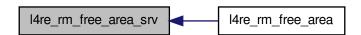
9.14.3.11 int l4re_rm_free_area_srv (I4_cap_idx_t rm, I4_addr_t addr)

See Also

L4Re::Rm::free_area

Referenced by I4re_rm_free_area().

Here is the caller graph for this function:



9.14.3.12 int l4re_rm_attach_srv (I4_cap_idx_t rm, void ** start, unsigned long size, unsigned long flags, l4re_ds_t const mem, I4_addr_t offs, unsigned char align)

See Also

L4Re::Rm::attach

Referenced by I4re_rm_attach().

Here is the caller graph for this function:



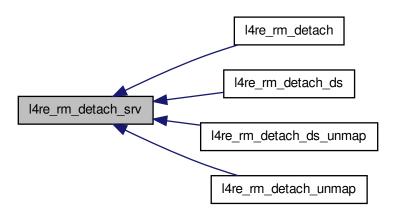
9.14.3.13 int l4re_rm_detach_srv (I4_cap_idx_t rm, I4_addr_t addr, I4re_ds_t * ds, I4_cap_idx_t task)

See Also

L4Re::Rm::detach

Referenced by I4re_rm_detach(), I4re_rm_detach_ds(), I4re_rm_detach_ds_unmap(), and I4re_rm_detach_unmap().

Here is the caller graph for this function:



9.14.3.14 int l4re_rm_find_srv (14_cap_idx_t rm, 14_addr_t * addr, unsigned long * size, 14_addr_t * offset, unsigned * flags, 14re_ds_t * m)

See Also

L4Re::Rm::find

Referenced by I4re_rm_find().

Here is the caller graph for this function:



9.15 Capability allocator

Capability allocator C interface.

Collaboration diagram for Capability allocator:



Functions

• I4_cap_idx_t I4re_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.

9.15.1 Detailed Description

Capability allocator C interface.

9.15.2 Function Documentation

9.15.2.1 long l4re_util_cap_last (void)

Return last capability index the allocator can return.

Returns

last/biggest capability index the allocator can return

9.16 Kumem allocator utility

Kumem allocator utility C interface.

Collaboration diagram for Kumem allocator utility:



Functions

• int l4re_util_kumem_alloc (l4_addr_t *mem, unsigned pages_order, l4_cap_idx_t task, l4_cap_idx_t regmgr) L4_NOTHROW

Get free capability index at capability allocator.

9.16.1 Detailed Description

Kumem allocator utility C interface.

9.16.2 Function Documentation

9.16.2.1 int l4re_util_kumem_alloc (I4_addr_t * mem, unsigned pages_order, I4_cap_idx_t task, I4_cap_idx_t regmgr)

Get free capability index at capability allocator.

Allocate state area.

Return values

mem	Pointer to memory that has been allocated.
pages_order	Size to allocate, in log2 pages.

Parameters

task	Task to use for allocation.
regmgr	Region manager to use for allocation.

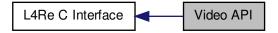
Returns

0 for success, error code otherwise

9.17 Video API 89

9.17 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

```
I4re_video_color_component_t I4re_video_color_component_t
```

Color component structure.

· typedef struct

l4re_video_pixel_info_t l4re_video_pixel_info_t

Pixel_info structure.

· typedef struct

I4re_video_view_info_t I4re_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_N-OTHROW

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_NOT-HROW

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_NOTH-ROW

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_N-OTHROW

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.

9.17.1 Detailed Description

9.17.2 Typedef Documentation

9.17.2.1 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.

9.17.3 Enumeration Type Documentation

9.17.3.1 enum l4re video goos info flags t

Flags of information on the goos.

9.17 Video API 91

Enumerator

F_I4re_video_goos_auto_refresh The graphics display is automatically refreshed.

F_l4re_video_goos_pointer We have a mouse pointer.

F_I4re_video_goos_dynamic_views Supports dynamically allocated views.

F_I4re_video_goos_dynamic_buffers Supports dynamically allocated buffers.

Definition at line 39 of file goos.h.

9.17.3.2 enum l4re_video_view_info_flags_t

Flags of information on a view.

Enumerator

F_I4re_video_view_none everything for this view is static (the VESA-FB case)

F_I4re_video_view_set_buffer buffer object for this view can be changed

F_I4re_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_I4re_video_view_set_position position on screen can be set

F_l4re_video_view_dyn_allocated View is dynamically allocated.

F_I4re_video_view_set_background Set view as background for session.

F_I4re_video_view_set_flags Set view property flags.

F_I4re_video_view_above Flag the view as stay on top.

F_I4re_video_view_flags_mask Mask containing all possible property flags.

Definition at line 33 of file view.h.

9.17.4 Function Documentation

9.17.4.1 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

9.17.4.2 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
У	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

9.17.4.3 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

9.17.4.4 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())

9.17.4.5 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

Γ	goos	the target goos object.
	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.

 $9.17.4.6 \quad int \ l4re_video_goos_create_view \ (\ l4re_video_goos_t \ goos, \ l4re_video_view_t * \textit{view} \)$

Create a new view (.

See Also

I4re_video_view_t)

Parameters

9.17 Video API 93

goos

Return values

view	the structure will be initialized for the new view.

9.17.4.7 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.

9.17.4.8 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 Identity also allows to access dynamic views created with Identity (also allows to access dynamic views created with Identity (also allows to access dynamic views created with Identity (also allows to access dynamic views created with Identity (also allows to access dynamic views created with Identity (also allows to access dynamic views created with Identity (also allows to access dynamic views created with Identity (also allows to access dynamic views created with Identity (also allows to access dynamic views created with Identity (also allows to access dynamic views created with Identity (also allows to access dynamic views created with Identity (also allows to access dynamic views (also allows to access the allows to access the access to a

9.17.4.9 int l4re_video_view_refresh (14re_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
у	y-coordinate of the upper left corner
W	the width of the rectangle
h	the height of the rectangle

 $9.17.4.10 \quad int \ l4re_video_view_get_info \ (\ l4re_video_view_t * \textit{view}, \ l4re_video_view_info_t * \textit{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.

9.17.4.11 int l4re_video_view_set_info (l4re_video_view_t * view, l4re_video_view_info_t * info) Set properties of the view.

9.17 Video API 95

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 Identification-view_get_info() and this depends on the concrete instance the view object.

9.17.4.12 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.
У	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.

9.17.4.13 int $l4re_video_view_t * view$, $l4re_video_view_t * pivot$, int behind)

Change the stacking order in the stack of visible views.

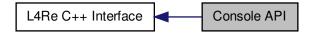
Parameters

view	the target view for the operation.
pivot	the neighbor view, relative to which <i>view</i> shall be stacked. a NULL value allows top (<i>behind</i> =
	1) and bottom (behind = 0) placement of the view.
behind	describes the placement of the view relative to the <i>pivot</i> view.

9.18 Console API

Console interface.

Collaboration diagram for Console API:



Data Structures

• class L4Re::Console Console class.

9.18.1 Detailed Description

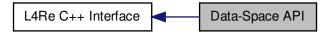
Console interface.

9.19 Data-Space API 97

9.19 Data-Space API

Data-Space API.

Collaboration diagram for Data-Space API:



Data Structures

class L4Re::Dataspace

This class represents a data space.

struct L4Re::Dataspace::Stats

Information about the data space.

9.19.1 Detailed Description

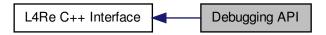
Data-Space API. Data spaces are a central abstraction provided by L4Re. A data space is an abstraction for any thing that is available via usual memory access instructions. A data space can be a file, as well as the memory-mapped registers of a device, or anonymous memory, such as a heap.

The data space interface defines a set of methods that allow any kind of data space to be attached (mapped) to the virtual address space of an L4 task and then be accessed via memory-access instructions. The region-map interface (L4Re::Rm) can be used to attach a data space to a virtual address space of a task paged by a certain instance of a region map (L4Re::Rm).

9.20 Debugging API

Debugging Interface.

Collaboration diagram for Debugging API:



Data Structures

class L4Re::Debug_obj
 Debug interface.

9.20.1 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.

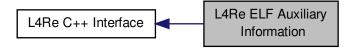
See Also

L4::Debug_obj for more information.

9.21 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 }

9.21.1 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.

9.21.2 Macro Definition Documentation

```
9.21.2.1 #define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".rol4re_elf_aux"), aligned(sizeof(I4_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_AU-X_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.

9.21.2.2 #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 stor-
	age 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:

Definition at line 67 of file elf_aux.h.

9.21.3 Enumeration Type Documentation

9.21.3.1 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.

9.22 Initial Environment 101

9.22 Initial Environment

Environment that is initially provided to an L4 task.

Collaboration diagram for Initial Environment:



Data Structures

· class L4Re::Env

Initial Environment (C++ version).

• struct l4re env cap entry t

Entry in the L4Re environment array for the named inital objects.

• struct l4re_env_t

Initial Environment structure (C version)

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 structure (C version)

Functions

I4re_env_t * I4re_env (void) L4_NOTHROW

Get L4Re initial environment (C version).

I4_kernel_info_t * I4re_kip (void) L4_NOTHROW

Get Kernel Info Page.

I4_cap_idx_t I4re_env_get_cap (char const *name) L4_NOTHROW

Get the capability selector for the object named name.

• I4_cap_idx_t I4re_env_get_cap_e (char const *name, I4re_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 I, l4re_env_t const *e) L4_-NOTHROW

Get the full I4re_env_cap_entry_t for the object named name.

9.22.1 Detailed Description

Environment that is initially 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).

The initial set of capabilities is:

- C[parent:L4Re::Parent] parent object
- C[mem_alloc:L4Re::Mem_alloc] initial memory allocator
- C[log:L4Re::Log] logging facility
- C[main_thread:L4::Thread] first application thread
- C[rm:L4::Rm] region manager
- · C[factory:L4::Factory] factory to create kernel objects
- C[task:L4::Task] the task itself

Additional information is:

- · First free entry in capability table
- The UTCB area (as flex page)
- · First free UTCB (address in the UTCB area)

See Also

L4Re::Env, l4re_env_t for more information.

9.22.2 Typedef Documentation

```
9.22.2.1 typedef struct I4re_env_t I4re_env_t
```

Initial Environment structure (C version)

See Also

Initial environment

9.22.3 Function Documentation

```
9.22.3.1 I4re_env_t * I4re_env(void) [inline]
```

Get L4Re initial environment (C version).

Returns

Pointer to L4Re initial environment (C version).

Examples:

examples/sys/aliens/main.c, examples/sys/isr/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 185 of file env.h.

Referenced by I4re_env_get_cap().

9.22 Initial Environment 103

Here is the caller graph for this function:



```
9.22.3.2 I4 kernel info t * I4re_kip ( void ) [inline]
```

Get Kernel Info Page.

Returns

Pointer to Kernel Info Page (KIP) structure.

Examples:

examples/sys/aliens/main.c, and examples/sys/ux-vhw/main.c.

Definition at line 189 of file env.h.

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 (I4_is_invalid_cap()).

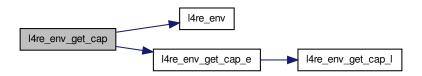
Examples:

examples/sys/isr/main.c.

Definition at line 227 of file env.h.

References I4re_env(), and I4re_env_get_cap_e().

Here is the call graph for this function:



9.22.3.4 I4_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*.

9.22 Initial Environment 105

Parameters

name	is the name of the object to lookup in the initial objects.
е	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 (I4_is_invalid_cap()).

Definition at line 214 of file env.h.

References I4re_env_cap_entry_t::cap, L4_INVALID_CAP, and I4re_env_get_cap_I().

Referenced by I4re_env_get_cap().

Here is the call graph for this function:



Here is the caller graph for this function:



9.22.3.5 | I4re_env_cap_entry_t const * I4re_env_get_cap_I (char const * name, unsigned I, I4re_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.
1	is the length of the name string, thus <i>name</i> might not be zero terminated.
е	is the environment structure to use for the operation.

Returns

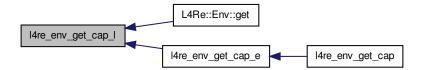
A pointer to an I4re_env_cap_entry_t if the object exists or NULL if not.

Definition at line 196 of file env.h.

References I4re_env_cap_entry_t::flags, and I4re_env_cap_entry_t::name.

Referenced by L4Re::Env::get(), and l4re_env_get_cap_e().

Here is the caller graph for this function:



9.23 Event API 107

9.23 Event API

Event interface.

Collaboration diagram for Event API:



Data Structures

class L4Re::Event

Event class.

class L4Re::Event_buffer_t< PAYLOAD >

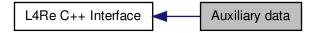
Event buffer class.

9.23.1 Detailed Description

Event interface.

9.24 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.

9.24.1 Detailed Description

9.25 Logging interface 109

9.25 Logging interface

Interface for log output.

Collaboration diagram for Logging interface:



Data Structures

class L4Re::Log

Log interface class.

9.25.1 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 an provide the possibility to tag output sent to a specific log object.

9.26 Memory allocator API

Memory-allocator interface.

Collaboration diagram for Memory allocator API:



Data Structures

class L4Re::Mem_alloc
 Memory allocator.

9.26.1 Detailed Description

Memory-allocator interface. The memory-allocator API is the basic API to allocate memory from the L4Re subsystem. The memory is allocated in terms of data spaces (see L4Re::Dataspace). The provided data spaces have at least the property that data written to such a data space is available as long as the data space is not freed or the data is not overwritten. In particular, the memory backing a data space from an allocator need not be allocated instantly, but may be allocated lazily on demand.

A memory allocator can provide data spaces with a 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.

The main interface is defined by the class L4Re::Mem_alloc.

9.27 Name-space API 111

9.27 Name-space API

API for name spaces that store capabilities.

Collaboration diagram for Name-space API:



Data Structures

class L4Re::Namespace

Name-space interface.

9.27.1 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 determine the policy which capabilities (which objects) are accessible to a client of a name space.

9.28 Parent API

Parent interface.

Collaboration diagram for Parent API:



Data Structures

class L4Re::Parent

Parent interface.

9.28.1 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.

9.29 L4Re Protocol identifiers

Basic protocol identifiers used for L4Re.

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::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::Protocol::Protocols {

L4Re::Protocol::Default = 0, L4Re::Protocol::Dataspace, L4Re::Protocol::Namespace, L4Re::Protocol::Parent,

 $L4Re:: Protocol:: Goos, \ L4Re:: Protocol:: Mem_alloc, \ L4Re:: Protocol:: Rm, \ L4Re:: Protocol:: Event, \ L4Re:: Protocol:: E$

L4Re::Protocol::Debug = ~0x7fffUL }

Protocols

These protocol IDs are used to distinguish requests for the different L4Re interfaces.

enum L4Re::Rm_::Opcodes

Region-map communication-protocol opcodes.

• enum L4Re::Video::Goos_::Opcodes

Frame buffer communication-protocol opcodes.

9.29.1 Detailed Description

Basic protocol identifiers used for L4Re.

9.29.2 Enumeration Type Documentation

9.29.2.1 enum L4Re::Protocol::Protocols

Protocols

These protocol IDs are used to distinguish requests for the different L4Re interfaces.

The interfaces use different protocol IDs to enable objects that realize a set of those interfaces at once.

Enumerator

Default Default protocol, used in message tag.

Dataspace ID for data space objects.

Namespace ID for name space objects.

Parent ID for parent objects.

Goos ID for goos objects.

Mem_alloc ID for memory allocator objects.

Rm ID for region map objects.

Event ID for event channel objects.

Debug ID for debug objects.

Definition at line 44 of file protocols.

9.30 Region map API

9.30 Region map API

Virtual address-space management.

Collaboration diagram for Region map API:



Data Structures

• class L4Re::Rm

Region map.

9.30.1 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

Data-Space API., L4Re::Dataspace, L4Re::Rm

9.31 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::Util::Smart_count_cap< Unmap_flags >

Helper for Ref_cap and Ref_del_cap.

struct L4Re::Util::Ref_cap< T >

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

struct L4Re::Util::Ref_del_cap< T >

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

Functions

virtual L4Re::Cap_alloc::~Cap_alloc ()=0
 Destructor.

Variables

 _Cap_alloc & L4Re::Util::cap_alloc Capability allocator.

9.31.1 Detailed Description

9.31.2 Variable Documentation

9.31.2.1 _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 a reference count 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.

9.32 Kumem utiltities

Collaboration diagram for Kumem utiltities:



Functions

• int L4Re::Util::kumem_alloc (I4_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.

9.32.1 Detailed Description

9.32.2 Function Documentation

9.32.2.1 int L4Re::Util::kumem_alloc (I4_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.

Return values

mem Pointer to memory that has been allocated.
--

Parameters

pages_order	Size to allocate, in log2 pages.
task	Task to use for allocation.
rm	Region manager to use for allocation.

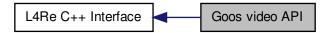
Returns

0 for success, error code otherwise

9.33 Goos video API

9.33 Goos video API

Collaboration diagram for Goos video API:



Data Structures

• class L4Re::Video::Color_component

A color component.

• class L4Re::Video::Pixel_info

Pixel information.

• class L4Re::Video::Goos

A goos.

• class L4Re::Video::View

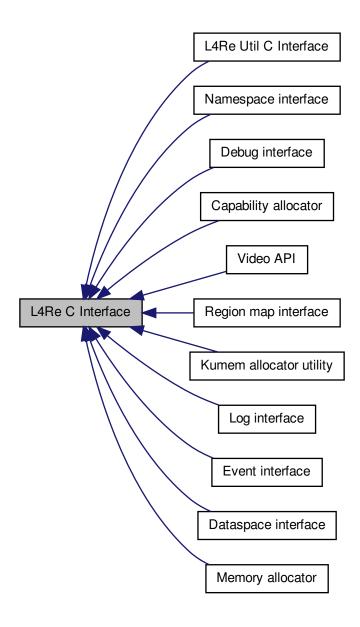
View.

9.33.1 Detailed Description

9.34 L4Re C Interface

Documentation for the L4Re C Interface.

Collaboration diagram for L4Re C Interface:



Modules

· Capability allocator

Capability allocator C interface.

• Dataspace interface

Dataspace C interface.

Debug interface

9.34 L4Re C Interface 121

· Event interface

Event C interface.

· 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

9.34.1 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.

9.35 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.

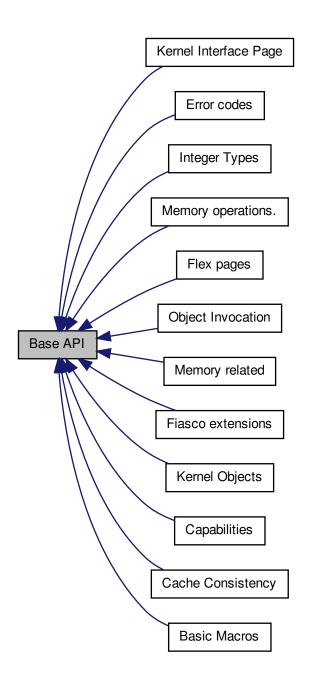
For new programs it is advised to use the C++ interface.

9.36 Base API 123

9.36 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.

· Cache Consistency

Various functions for cache consistency.

Capabilities

Functions and definitions related to capabilities.

Error codes

Common error codes.

· Fiasco extensions

Kernel debugger extensions of the Fiasco L4 implementation.

· Flex pages

Flex-page related API.

Integer Types

#include<14/sys/14int.h>

· 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

Common factory 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

Interrupt functionality.

• file irq.h

Interrupt functionality.

· file kip

L4::Kip class, memory descriptors.

• file kip.h

9.36 Base API 125

Kernel Info Page access functions.

· file memdesc.h

Memory description functions.

• file meta

Meta interface for getting dynamic type information about objects behind capabilities.

· 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.

9.36.1 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.

9.37 IPC-Gate API

Secure comminication object.

Collaboration diagram for IPC-Gate API:



Data Structures

class L4::lpc_gate
 L4 IPC gate.

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

- I4_msgtag_t I4_ipc_gate_bind_thread (I4_cap_idx_t gate, I4_cap_idx_t thread, I4_umword_t label)

 Bind the IPC-gate to the thread.
- I4_msgtag_t I4_ipc_gate_get_infos (I4_cap_idx_t gate, I4_umword_t *label)

 Get information on the IPC-gate.

9.37.1 Detailed Description

Secure comminication 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 (I4_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.

See Also

Object Invocation

9.37 IPC-Gate API

9.37.2 Enumeration Type Documentation

9.37.2.1 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 75 of file ipc_gate.h.

9.37.3 Function Documentation

9.37.3.1 I4_msgtag_t I4_ipc_gate_bind_thread (I4_cap_idx_t gate, I4_cap_idx_t thread, I4_umword_t label) [inline]

Bind the IPC-gate to the thread.

Parameters

t	Thread to bind the IPC-gate to
label	Label to use
utcb	UTCB to use.

Returns

System call return tag.

Definition at line 117 of file ipc_gate.h.

References I4_utcb().

Here is the call graph for this function:



9.37.3.2 I4_msgtag_t I4_ipc_gate_get_infos (I4_cap_idx_t gate, I4_umword_t * label) [inline]

Get information on the IPC-gate.

Return values

label	Label of the gate.
-------	--------------------

Parameters

utcb UTCb to use.

Returns

System call return tag.

Definition at line 124 of file ipc_gate.h.

References I4_utcb().

Here is the call graph for this function:



9.38 Basic Macros

9.38 Basic Macros

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

Collaboration diagram for Basic Macros:



Macros

#define L4_DECLARE_CONSTRUCTOR(func, prio)

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 EXPORT

Attribute to mark functions, variables, and data types as being exported from a library.

#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 EXPECT_TRUE(x)

Expression is likely to execute.

• #define EXPECT_FALSE(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 I4 barrier (void)

Memory barrier.

void I4_mb (void)

Memory barrier.

void I4_wmb (void)

Write memory barrier.

9.38.1 Detailed Description

L4 standard macros for header files, function definitions, and public APIs etc. #include <14/sys/compiler.-h>

9.38.2 Macro Definition Documentation

```
9.38.2.1 #define L4_DECLARE_CONSTRUCTOR( func, prio )
```

L4 Inline function attribute.

Handcoded version of attribute((constructor(xx))).

Parameters

func	function declaration (prototype)
prio	the prio must be 65535 - gcc_prio

Definition at line 84 of file compiler.h.

9.38.2.2 #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 equivalent to throw().

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

Definition at line 202 of file compiler.h.

9.38 Basic Macros

9.38.2.3 #define L4_EXPORT

Attribute to mark functions, variables, and data types as being exported from a library.

All data types, functions, and global variables that shall be exported from a library shall be marked with this attribute. The default may become to hide everything that is not marked as L4_EXPORT from the users of a library and provide the possibility for aggressive optimization of all those internal functionality of a library.

Usage:

Definition at line 232 of file compiler.h.

9.38.2.4 #define L4_HIDDEN

Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.

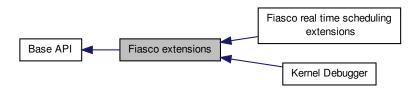
This attribute is intended for functions, data, and data types that shall never be visible outside of a library. In particular, for shared libraries this may result in much faster code within the library and short linking times.

Definition at line 229 of file compiler.h.

9.39 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

I4f specific fs/gs manipulation

• file segment.h

14f specific segment manipulation

Data Structures

• struct I4_tracebuffer_status_t

Trace buffer status.

• struct I4_tracebuffer_status_window_t

Trace-buffer status window descriptor.

Macros

• #define LOG_EVENT_CONTEXT_SWITCH 0

Event: context switch.

#define LOG_EVENT_IPC_SHORTCUT 1

Event: IPC shortcut.

#define LOG EVENT IRQ RAISED 2

Event: IRQ occurred.

• #define LOG_EVENT_TIMER_IRQ 3

Event: Timer IRQ occurred.

• #define LOG EVENT THREAD EX REGS 4

Event: thread_ex_regs.

• #define LOG_EVENT_MAX_EVENTS 16

9.39 Fiasco extensions 133

Maximum number of events.

#define LOG_EVENT_CONTEXT_SWITCH 0

Event: context switch.

• #define LOG EVENT IPC SHORTCUT 1

Event: IPC shortcut.

• #define LOG_EVENT_IRQ_RAISED 2

Event: IRQ occurred.

#define LOG_EVENT_TIMER_IRQ 3

Event: Timer IRQ occurred.

• #define LOG_EVENT_THREAD_EX_REGS 4

Event: thread_ex_regs.

#define LOG EVENT MAX EVENTS 16

Maximum number of events.

Enumerations

enum

Log event types.

Functions

I4_tracebuffer_status_t * fiasco_tbuf_get_status (void)

Return trace buffer status.

I4_addr_t fiasco_tbuf_get_status_phys (void)

Return the physical address of the trace buffer status struct.

I4_umword_t fiasco_tbuf_log (const char *text)

Create new trace buffer entry with describing < text>.

• I4_umword_t fiasco_tbuf_log_3val (const char *text, unsigned v1, unsigned v2, unsigned v3)

Create new trace buffer entry with describing < text> and three additional values.

• 14 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.

void fiasco_profile_start (void) L4_NOTHROW

Start profiling.

· void fiasco profile stop and dump (void) L4 NOTHROW

Stop profiling and dump result to console.

void fiasco_profile_stop (void) L4_NOTHROW

Stop profiling.

void fiasco_watchdog_enable (void) L4_NOTHROW

Enable Fiasco watchdog.

void fiasco watchdog disable (void) L4 NOTHROW

Disable Fiasco watchdog.

void fiasco_watchdog_takeover (void) L4_NOTHROW

Disable automatic resetting of watchdog.

void fiasco_watchdog_giveback (void) L4_NOTHROW

Reenable automatic resetting of watchdog.

void fiasco_watchdog_touch (void) L4_NOTHROW

Reset watchdog from user land.

long fiasco_ldt_set (l4_cap_idx_t task, void *ldt, unsigned int size, unsigned int entry_number_start, l4_utcb_t *utcb)

Set LDT segments descriptors.

long fiasco_gdt_set (I4_cap_idx_t thread, void *desc, unsigned int size, unsigned int entry_number_start,
 I4_utcb_t *utcb)

Set GDT segment descriptors.

unsigned fiasco gdt get entry offset (I4 cap idx t thread, I4 utcb t *utcb)

Return the offset of the entry in the GDT.

9.39.1 Detailed Description

Kernel debugger extensions of the Fiasco L4 implementation.

9.39.2 Function Documentation

```
9.39.2.1 I4_tracebuffer_status_t * fiasco_tbuf_get_status(void) [inline]
```

Return trace buffer status.

Return trace-buffer status.

Return tracebuffer status.

Returns

Pointer to trace buffer status struct.

Pointer to tracebuffer status struct.

Pointer to trace-buffer status struct.

Definition at line 183 of file ktrace.h.

```
9.39.2.2 I4_addr_t fiasco_tbuf_get_status_phys(void) [inline]
```

Return the physical address of the trace buffer status struct.

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

Return the physical address of the tracebuffer status struct.

Returns

physical address of status struct.

Definition at line 190 of file ktrace.h.

```
9.39.2.3 I4_umword_t fiasco_tbuf_log ( const char * text ) [inline]
```

Create new trace buffer entry with describing <text>.

Create new trace-buffer entry with describing <text>.

Create new tracebuffer entry with describing <text>.

9.39 Fiasco extensions

Parameters

text	Logging text
------	--------------

Returns

Pointer to trace buffer entry

Parameters

text	Logging text

Returns

Pointer to tracebuffer entry

Parameters

text	Logging text
------	--------------

Returns

Pointer to trace-buffer entry

Definition at line 197 of file ktrace.h.

9.39.2.4 I4_umword_t fiasco_tbuf_log_3val (const char * text, unsigned v1, unsigned v2, unsigned v3) [inline]

Create new trace buffer entry with describing <text> and three additional values.

Create new trace-buffer entry with describing <text> and three additional values.

Create new tracebuffer 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

Parameters

	text	Logging text
	v1	first value
	v2	second value
Ì	v3	third value

Returns

Pointer to tracebuffer entry

Parameters

text	Logging text
v1	first value
v2	second value
v3	third value

Returns

Pointer to trace-buffer entry

Definition at line 203 of file ktrace.h.

9.39.2.5 I4_umword_t fiasco_tbuf_log_binary (const unsigned char * data) [inline]

Create new trace buffer entry with binary data.

Create new trace-buffer entry with binary data.

Create new tracebuffer entry with binary data.

Parameters

data	binary data

Returns

Pointer to trace buffer entry

Parameters

data	binary data

Returns

Pointer to tracebuffer entry

Parameters

data	binary data
------	-------------

Returns

Pointer to trace-buffer entry

Definition at line 233 of file ktrace.h.

9.39.2.6 void fiasco_tbuf_clear (void) [inline]

Clear trace buffer.

Clear trace-buffer.

Clear tracebuffer.

Definition at line 209 of file ktrace.h.

9.39.2.7 void fiasco_tbuf_dump(void) [inline]

Dump trace buffer to kernel console.

9.39 Fiasco extensions 137

Dump trace-buffer to kernel console.

Dump tracebuffer to kernel console.

Definition at line 215 of file ktrace.h.

9.39.2.8 void fiasco_watchdog_takeover(void) [inline]

Disable automatic resetting of watchdog.

User is responsible to call fiasco_watchdog_touch from time to time to ensure that the watchdog does not trigger.

Definition at line 407 of file kdebug.h.

9.39.2.9 void fiasco_watchdog_touch (void) [inline]

Reset watchdog from user land.

This function **must** be called from time to time to prevent the watchdog from triggering if the watchdog is activated and if fiasco_watchdog_takeover was performed.

Definition at line 419 of file kdebug.h.

9.39.2.10 long fiasco_ldt_set (I4_cap_idx_t task, void * ldt, unsigned int size, unsigned int entry_number_start, I4_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	Entry number to start.
start	
utcb	UTCB of the caller.

Definition at line 123 of file segment.h.

References L4_EINVAL, I4_ipc_call(), L4_IPC_NEVER, I4_msgtag(), L4_PROTO_TASK, and I4_msg_regs_t::mr.

Here is the call graph for this function:



9.39.2.11 long fiasco_gdt_set (I4_cap_idx_t thread, void * desc, unsigned int size, unsigned int entry_number_start, I4_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	Entry number to start (valid values: 0-2).
start	
utcb	UTCB of the caller.

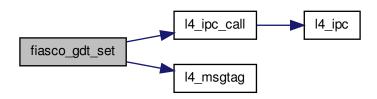
Returns

System call error

Definition at line 52 of file segment.h.

References L4_ENOSYS, I4_ipc_call(), L4_IPC_NEVER, I4_msgtag(), L4_PROTO_THREAD, L4_THREAD_X86-_GDT_OP, and I4_msg_regs_t::mr.

Here is the call graph for this function:



9.39.2.12 unsigned fiasco_gdt_get_entry_offset ($I4_cap_idx_t thread$, $I4_utcb_t*utcb$) [inline]

Return the offset of the entry in the GDT.

Parameters

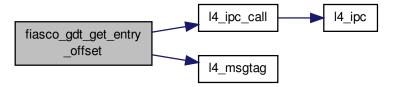
thread	Thread to get info from.
utcb	UTCB of the caller.

Definition at line 136 of file segment.h.

References I4_ipc_call(), L4_IPC_NEVER, I4_msgtag(), L4_PROTO_THREAD, L4_THREAD_X86_GDT_OP, and I4_msg_regs_t::mr.

9.39 Fiasco extensions

Here is the call graph for this function:



9.40 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.

9.41 Flex pages 141

9.41 Flex pages

Flex-page related API.

Collaboration diagram for Flex pages:



Data Structures

```
• union I4_fpage_t
```

L4 flexpage type.

• struct I4_snd_fpage_t

Send-flex-page types.

Enumerations

```
    enum I4_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 flexpage structure.

enum { L4_WHOLE_ADDRESS_SPACE = 63 }

Constants for flexpages.

• enum L4_fpage_rights { L4_FPAGE_RO = 4, L4_FPAGE_RW = 6 }

Flex-page rights.

enum L4_cap_fpage_rights { L4_CAP_FPAGE_R = 0x4, L4_CAP_FPAGE_RO = 0x4, L4_CAP_FPAGE_RW = 0x5 }

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

Flex-page map control for capabilities (snd_base)

enum I4_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

I4_fpage_t I4_fpage (unsigned long address, unsigned int size, unsigned char rights) L4_NOTHROW
 Create a memory flex page.

I4_fpage_t I4_fpage_all (void) L4_NOTHROW

Get a flex page, describing all address spaces at once.

• I4_fpage_t I4_fpage_invalid (void) L4_NOTHROW

Get an invalid flex page.

• I4_fpage_t I4_iofpage (unsigned long port, unsigned int size) L4_NOTHROW

Create an IO-port flex page.

I4_fpage_t I4_obj_fpage (I4_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 I4_fpage_rights (I4_fpage_t f) L4_NOTHROW

Return rights from a flex page.

unsigned I4_fpage_type (I4_fpage_t f) L4_NOTHROW

Return type from a flex page.

unsigned I4_fpage_size (I4_fpage_t f) L4_NOTHROW

Return size from a flex page.

unsigned long I4_fpage_page (I4_fpage_t f) L4_NOTHROW

Return page from a flex page.

- I4_fpage_t I4_fpage_set_rights (I4_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 I4_fpage_max_order (unsigned char order, I4_addr_t addr, I4_addr_t min_addr, I4_addr_t max-addr, I4_addr_t hotspot L4_DEFAULT_PARAM(0))

Determine maximum flex page size of a region.

9.41.1 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 I4_task_unmap() method, that can be used to describe all address spaces (all types) with a single flex page.

9.41.2 Enumeration Type Documentation

9.41.2.1 enum I4 fpage consts

L4 flexpage structure.

Enumerator

L4_FPAGE_RIGHTS_SHIFT Access permissions shift.

9.41 Flex pages 143

```
L4_FPAGE_TYPE_SHIFT Flexpage type shift (memory, IO port, obj...)
    L4_FPAGE_SIZE_SHIFT Flexpage size shift (log2-based)
    L4_FPAGE_ADDR_SHIFT Page address shift.
    L4_FPAGE_RIGHTS_BITS Access permissions size.
    L4_FPAGE_TYPE_BITS Flexpage type size (memory, IO port, obj...)
    L4_FPAGE_SIZE_BITS Flexpage size size (log2-based)
    L4_FPAGE_ADDR_BITS Page address size.
Definition at line 55 of file ___I4_fpage.h.
9.41.2.2 anonymous enum
Constants for flexpages.
Enumerator
     L4_WHOLE_ADDRESS_SPACE Whole address space size.
Definition at line 86 of file __l4_fpage.h.
9.41.2.3 enum L4 fpage rights
Flex-page rights.
Enumerator
    L4_FPAGE_RO Read-only flex page.
    L4_FPAGE_RW Read-write flex page.
Definition at line 104 of file ___l4_fpage.h.
9.41.2.4 enum L4 cap fpage rights
Cap-flex-page rights.
Enumerator
    L4_CAP_FPAGE_R Read-only cap.
    L4_CAP_FPAGE_RO Read-only cap.
    L4_CAP_FPAGE_RW Read-write cap.
Definition at line 117 of file ___l4_fpage.h.
9.41.2.5 enum I4 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 164 of file __l4_fpage.h.
```

9.41.2.6 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 183 of file __l4_fpage.h.

9.41.3 Function Documentation

9.41.3.1 I4_fpage_t I4_fpage (unsigned long 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 I4_fpage_rights

Returns

Memory flex page

Definition at line 453 of file __l4_fpage.h.

```
9.41.3.2 I4_fpage_t I4_fpage_all ( void ) [inline]
```

Get a flex page, describing all address spaces at once.

Returns

Special all-spaces flex page.

Definition at line 471 of file __l4_fpage.h.

References L4_WHOLE_ADDRESS_SPACE.

9.41.3.3 I4_fpage_t I4_fpage_invalid (void) [inline]

Get an invalid flex page.

Returns

Special invalid flex page.

Definition at line 477 of file __l4_fpage.h.

9.41.3.4 I4 fpage t I4_iofpage (unsigned long port, unsigned int size) [inline]

Create an IO-port flex page.

9.41 Flex pages 145

Parameters

port	I/O-flex-page port base
size	I/O-flex-page size, L4_WHOLE_IOADDRESS_SPACE to specify the whole I/O address
	space (with port 0)

Returns

I/O flex page

Definition at line 459 of file __l4_fpage.h.

References L4_FPAGE_ADDR_SHIFT, and L4_FPAGE_RW.

9.41.3.5 I4_fpage_t I4_obj_fpage (I4_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

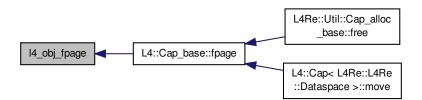
Returns

Flex page for a set of kernel objects.

Definition at line 465 of file __l4_fpage.h.

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

Here is the caller graph for this function:



9.41.3.6 int l4_is_fpage_writable (l4_fpage_t fp) [inline]

Test if the flex page is writable.

Parameters

fp Flex page.

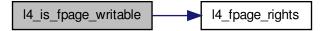
Returns

!= 0 if flex page is writable, 0 if not

Definition at line 484 of file __l4_fpage.h.

References I4_fpage_rights().

Here is the call graph for this function:



9.41.3.7 unsigned I4_fpage_rights (I4_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 403 of file __l4_fpage.h.

References L4_FPAGE_RIGHTS_SHIFT.

Referenced by I4_is_fpage_writable().

Here is the caller graph for this function:



9.41.3.8 unsigned I4_fpage_type(I4_fpage_t f) [inline]

Return type from a flex page.

9.41 Flex pages 147

Parameters

```
f | Flex page
```

Returns

Type part of the given flex page.

Definition at line 409 of file __l4_fpage.h.

References L4_FPAGE_TYPE_SHIFT.

9.41.3.9 unsigned I4_fpage_size(I4_fpage_t f) [inline]

Return size from a flex page.

Parameters

```
f | Flex page
```

Returns

Size part of the given flex page.

Definition at line 415 of file __l4_fpage.h.

References L4_FPAGE_SIZE_SHIFT.

9.41.3.10 unsigned long l4_fpage_page(l4_fpage_t f) [inline]

Return page from a flex page.

Parameters

```
f | Flex page
```

Returns

Page part of the given flex page.

Definition at line 421 of file __l4_fpage.h.

References L4_FPAGE_ADDR_SHIFT.

Referenced by I4_fpage_contains().

Here is the caller graph for this function:



9.41.3.11 I4_fpage_t I4_fpage_set_rights (I4_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 444 of file __l4_fpage.h.

References L4_FPAGE_RIGHTS_SHIFT, and I4_fpage_t::raw.

9.41.3.12 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.

Definition at line 503 of file __l4_fpage.h.

References L4_FPAGE_ADDR_SHIFT, and I4_fpage_page().

Here is the call graph for this function:



9.41.3.13 unsigned char I4_fpage_max_order (unsigned char order, I4_addr_t addr, I4_addr_t min_addr, I4_addr_t max_addr, I4_addr_t hotspot L4_DEFAULT_PARAMO) [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.

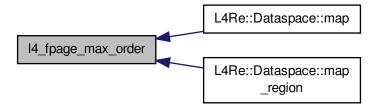
9.41 Flex pages 149

Note

The start address of the flex-page can be determined with I4_trunc_size(addr, returnvalue)

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

Here is the caller graph for this function:



9.42 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

- I4_umword_t I4_map_control (I4_umword_t spot, unsigned char cache, unsigned grant) L4_NOTHROW

 Create the first word for a map item for the memory space.
- I4_umword_t I4_map_obj_control (I4_umword_t spot, unsigned grant) L4_NOTHROW

 Create the first word for a map item for the object space.

9.42.1 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.

9.42.2 Enumeration Type Documentation

```
9.42.2.1 enum I4_msg_item_consts_t
```

Constants for message items.

Enumerator

L4_ITEM_MAP Identify a message item as map item.

L4_ITEM_CONT Donote 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.

9.42 Message Items 151

L4_RCV_ITEM_SINGLE_CAP Mark the receive buffer to be a small receive item that describes a buffer for a single capability.

L4_RCV_ITEM_LOCAL_ID The receiver requests to receive a local ID instead of a mapping whenever possible.

Definition at line 193 of file consts.h.

9.42.3 Function Documentation

9.42.3.1 I4_umword_t I4_map_control (I4_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 490 of file __l4_fpage.h.

References L4_ITEM_MAP.

Referenced by I4_map_obj_control().

Here is the caller graph for this function:



9.42.3.2 I4_umword_t I4_map_obj_control(I4_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 497 of file __l4_fpage.h.

References I4_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:

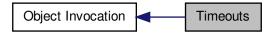


9.43 Timeouts 153

9.43 Timeouts

All kinds of timeouts and time related functions.

Collaboration diagram for Timeouts:



Data Structures

struct I4_timeout_s

Basic timeout specification.

• union I4_timeout_t

Timeout pair.

Macros

```
    #define L4_IPC_TIMEOUT_0 ((I4_timeout_s){0x0400})
```

Timeout constants.

• #define L4_IPC_TIMEOUT_NEVER ((I4_timeout_s){0})

never timeout

• #define L4_IPC_NEVER_INITIALIZER {0}

never timeout, init

#define L4_IPC_NEVER ((I4_timeout_t){0})

never timeout

#define L4_IPC_RECV_TIMEOUT_0 ((I4_timeout_t){0x00000400})

0 receive timeout

#define L4_IPC_SEND_TIMEOUT_0 ((I4_timeout_t){0x04000000})

0 send timeout

#define L4_IPC_BOTH_TIMEOUT_0 ((I4_timeout_t){0x04000400})

0 receive and send timeout

Typedefs

typedef struct I4_timeout_s I4_timeout_s

Basic timeout specification.

• typedef union I4_timeout_t I4_timeout_t

Timeout pair.

Enumerations

· enum I4_timeout_abs_validity

Intervals of validity for absolute timeouts Times are actually $2^{\wedge}x$ values (e.g.

Functions

I4_timeout_s I4_timeout_rel (unsigned man, unsigned exp) L4_NOTHROW

Get relative timeout consisting of mantissa and exponent.

I4_timeout_t I4_ipc_timeout (unsigned snd_man, unsigned snd_exp, unsigned rcv_man, unsigned rcv_exp)
 L4_NOTHROW

Convert explicit timeout values to I4_timeout_t type.

• 14 timeout t 14 timeout (14 timeout s snd, 14 timeout s rcv) L4 NOTHROW

Combine send and receive timeout in a timeout.

• void I4_snd_timeout (I4_timeout_s snd, I4_timeout_t *to) L4_NOTHROW

Set send timeout in given to timeout.

void I4_rcv_timeout (I4_timeout_s rcv, I4_timeout_t *to) L4_NOTHROW

Set receive timeout in given to timeout.

I4_kernel_clock_t I4_timeout_rel_get (I4_timeout_s to) L4_NOTHROW

Get clock value of out timeout.

unsigned I4_timeout_is_absolute (I4_timeout_s to) L4_NOTHROW

Return whether the given timeout is absolute or not.

• I4_kernel_clock_t I4_timeout_get (I4_kernel_clock_t cur, I4_timeout_s to) L4_NOTHROW

Get clock value for a clock + a timeout.

I4_timeout_s I4_timeout_abs (I4_kernel_clock_t pint, int br) L4_NOTHROW

Set an absolute timeout.

unsigned I4_utcb_mr64_idx (unsigned idx) L4_NOTHROW

Get index into 64bit message registers alias from native-sized index.

9.43.1 Detailed Description

All kinds of timeouts and time related functions.

9.43.2 Macro Definition Documentation

```
9.43.2.1 #define L4_IPC_TIMEOUT_0 ((I4_timeout_s){0x0400})
```

Timeout constants.

0 timeout

Definition at line 77 of file __timeout.h.

9.43.3 Typedef Documentation

```
9.43.3.1 typedef struct I4_timeout_s I4_timeout_s
```

Basic timeout specification.

Basically a floating point number with 10 bits mantissa and 5 bits exponent ($t = m*2^{\circ}e$).

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

9.43.3.2 typedef union I4_timeout_t I4_timeout_t

Timeout pair.

For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

9.43 Timeouts 155

9.43.4 Enumeration Type Documentation

9.43.4.1 enum I4_timeout_abs_validity

Intervals of validity for absolute timeouts

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

 $2ms -> 2048 \mu s$)

Definition at line 92 of file __timeout.h.

9.43.5 Function Documentation

9.43.5.1 I4_timeout_s I4_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.

9.43.5.2 I4_timeout_t I4_ipc_timeout (unsigned *snd_man*, unsigned *snd_exp*, unsigned *rcv_man*, unsigned *rcv_exp*) [inline]

Convert explicit timeout values to I4_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 I4_timeout_t::p, I4_timeout_t::rcv, I4_timeout_t::snd, and I4_timeout_s::t.

9.43.5.3 I4_timeout_t I4_timeout(I4_timeout_s snd, I4_timeout_s rcv) [inline]

Combine send and receive timeout in a timeout.

Parameters

ſ	snd	Send timeout
	rcv	Receive timeout

Returns

L4 timeout

Definition at line 221 of file __timeout.h.

References I4_timeout_t::p, I4_timeout_t::rcv, and I4_timeout_t::snd.

9.43.5.4 void I4_snd_timeout (I4_timeout_s snd, I4_timeout_t * to) [inline]

Set send timeout in given to timeout.

9.43 Timeouts

Parameters

snd | Send timeout

Return values

to L4 timeout

Definition at line 231 of file timeout.h.

9.43.5.5 void I4_rcv_timeout (I4_timeout_s rcv, I4_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.

9.43.5.6 I4_kernel_clock_t I4_timeout_rel_get(I4_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.

Referenced by I4_timeout_get().

Here is the caller graph for this function:



9.43.5.7 unsigned I4_timeout_is_absolute(I4_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.

Referenced by I4_timeout_get().

Here is the caller graph for this function:



9.43.5.8 I4_kernel_clock_t I4_timeout_get(I4_kernel_clock_t cur, I4_timeout_s to) [inline]

Get clock value for a clock + a timeout.

Parameters

cur	Clock value
to	L4 timeout

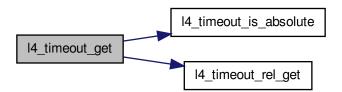
Returns

Clock sum

Definition at line 268 of file __timeout.h.

References I4_timeout_is_absolute(), and I4_timeout_rel_get().

Here is the call graph for this function:



9.43 Timeouts 159

9.43.5.9 I4_timeout_s I4_timeout_abs (I4_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.

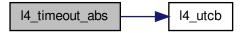
Returns

timeout value

Definition at line 373 of file utcb.h.

References I4_utcb().

Here is the call graph for this function:



9.43.5.10 unsigned I4_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 376 of file utcb.h.

9.44 VM API for SVM 161

9.44 VM API for SVM

Virtual machine API for SVM.

Collaboration diagram for VM API for SVM:



Data Structures

• struct I4_vm_svm_vmcb_control_area

VMCB structure for SVM VMs.

• struct I4 vm svm vmcb state save area seg

State save area segment selector struct.

struct I4_vm_svm_vmcb_state_save_area

State save area structure for SVM VMs.

• struct I4_vm_svm_vmcb_t

Control structure for SVM VMs.

Typedefs

· typedef struct

I4_vm_svm_vmcb_control_area I4_vm_svm_vmcb_control_area_t

VMCB structure for SVM VMs.

· typedef struct

I4_vm_svm_vmcb_state_save_area_seg I4_vm_svm_vmcb_state_save_area_seg_t

State save area segment selector struct.

· typedef struct

I4_vm_svm_vmcb_state_save_area I4_vm_svm_vmcb_state_save_area_t

State save area structure for SVM VMs.

typedef struct I4_vm_svm_vmcb_t I4_vm_svm_vmcb_t

Control structure for SVM VMs.

9.44.1 Detailed Description

Virtual machine API for SVM.

9.45 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_FIXED0-REG = 6, L4_VM_VMX_CR0_FIXED1_REG = 7,
    L4_VM_VMX_CR4_FIXED0_REG = 8, L4_VM_VMX_CR4_FIXED1_REG = 9, L4_VM_VMX_VMCS_ENUM_REG = 0xa, L4_VM_VMX_PROCBASED_CTLS2_REG = 0xb,
    L4_VM_VMX_EPT_VPID_CAP_REG = 0xc, L4_VM_VMX_NUM_CAPS_REGS }
```

Exported VMX capability registers.

enum L4_vm_vmx_dfl1_regs {
 L4_VM_VMX_PINBASED_CTLS_DFL1_REG = 0x1, L4_VM_VMX_PROCBASED_CTLS_DFL1_REG =
 0x2, L4_VM_VMX_EXIT_CTLS_DFL1_REG = 0x3, L4_VM_VMX_ENTRY_CTLS_DFL1_REG = 0x4,
 L4_VM_VMX_NUM_DFL1_REGS }

Exported VMX capability registers (default to 1 bits).

• enum { L4_VM_VMX_VMCS_CR2 = 0x683e }

Additional VMCS fields.

Functions

- I4_uint64_t I4_vm_vmx_get_caps (void const *vcpu_state, unsigned cap_msr) L4_NOTHROW Get a capability register for VMX.
- I4_uint32_t I4_vm_vmx_get_caps_default1 (void const *vcpu_state, unsigned cap_msr) L4_NOTHROW
 Get a default to one capability register for VMX.
- unsigned I4_vm_vmx_field_len (unsigned field) L4_NOTHROW

Return length in bytes of a VMCS field.

• unsigned I4_vm_vmx_field_order (unsigned field) L4_NOTHROW

Return length in power of two (bytes) of a VMCS field.

void I4 vm vmx clear (void *vmcs, void *user vmcs) L4 NOTHROW

Saves cached state from the kernel VMCS to the user VMCS.

void I4_vm_vmx_ptr_load (void *vmcs, void *user_vmcs) L4_NOTHROW

Loades the user_vmcs as the current VMCS.

• I4_uint32_t I4_vm_vmx_get_cr2_index (void const *vmcs) L4_NOTHROW

Get the VMCS field index of the virtual CR2 register.

• I4_umword_t I4_vm_vmx_read_nat (void *vmcs, unsigned field) L4_NOTHROW

Read a natural width VMCS field.

9.45 VM API for VMX 163

- I4_uint16_t I4_vm_vmx_read_16 (void *vmcs, unsigned field) L4_NOTHROW
 Read a 16bit VMCS field.
- I4_uint32_t I4_vm_vmx_read_32 (void *vmcs, unsigned field) L4_NOTHROW Read a 32bit VMCS field.
- I4_uint64_t I4_vm_vmx_read_64 (void *vmcs, unsigned field) L4_NOTHROW Read a 64bit VMCS field.
- I4_uint64_t L4_vm_vmx_read (void *vmcs, unsigned field) L4_NOTHROW
 Read any VMCS field.
- void I4_vm_vmx_write_nat (void *vmcs, unsigned field, I4_umword_t val) L4_NOTHROW
 Write to a natural width VMCS field.
- void I4_vm_vmx_write_16 (void *vmcs, unsigned field, I4_uint16_t val) L4_NOTHROW
 Write to a 16bit VMCS field.
- void I4_vm_vmx_write_32 (void *vmcs, unsigned field, I4_uint32_t val) L4_NOTHROW
 Write to a 32bit VMCS field.
- void I4_vm_vmx_write_64 (void *vmcs, unsigned field, I4_uint64_t val) L4_NOTHROW
 Write to a 64bit VMCS field.
- void I4_vm_vmx_write (void *vmcs, unsigned field, I4_uint64_t val) L4_NOTHROW
 Write to an arbitrary VMCS field.

9.45.1 Detailed Description

Virtual machine API for VMX.

9.45.2 Enumeration Type Documentation

9.45.2.1 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.

```
9.45.2.2 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.

9.45.2.3 anonymous enum

Additional VMCS fields.

Enumerator

L4_VM_VMX_VMCS_CR2 (virtual) VMCS offset for CR2. 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 version of the VMCS.

Note

You usually need to check this value against the value you get from I4_vm_vmx_get_cr2_index() to make sure you are running on a compatible kernel.

Definition at line 100 of file __vm-vmx.h.

9.45.3 Function Documentation

9.45.3.1 14 uint64 t 14 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 Also

```
L4_vm_vmx_caps_regs).
```

Returns

The value of the capability register.

Definition at line 506 of file __vm-vmx.h.

References L4_VCPU_OFFSET_EXT_INFOS.

9.45.3.2 14 uint32 t 14 vm vmx get caps default1 (void const * vcpu state, unsigned cap msr) [inline]

Get a default to one capability register for VMX.

9.45 VM API for VMX 165

Parameters

vcpu_state	Pointer to the VCPU state of the VCPU.
cap_msr	Default 1 caps register index (

See Also

L4_vm_vmx_dfl1_regs).

Returns

The value of the capability register.

Definition at line 514 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.

9.45.3.3 unsigned I4_vm_vmx_field_len (unsigned field) [inline]

Return length in bytes of a VMCS field.

Parameters

tield	Field number.
IICIU	i icia ilalibol.

Returns

Width of field in bytes.

Definition at line 335 of file __vm-vmx.h.

References I4_vm_vmx_field_order().

Here is the call graph for this function:



9.45.3.4 unsigned I4_vm_vmx_field_order (unsigned field) [inline]

Return length in power of two (bytes) of a VMCS field.

Parameters

field F	Field number.
---------	---------------

Returns

Width of field in power of two (bytes).

Definition at line 320 of file __vm-vmx.h.

Referenced by I4_vm_vmx_field_len().

Here is the caller graph for this function:



9.45.3.5 void I4_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 411 of file __vm-vmx.h.

Referenced by I4_vm_vmx_ptr_load().

Here is the caller graph for this function:



9.45.3.6 void I4_vm_vmx_ptr_load (void * vmcs, void * user_vmcs) [inline]

Loades 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 423 of file __vm-vmx.h.

References I4_vm_vmx_clear().

9.45 VM API for VMX 167

Here is the call graph for this function:



9.45.3.7 I4_uint32_t I4_vm_vmx_get_cr2_index (void const * vmcs) [inline]

Get the VMCS field index of the virtual CR2 register.

Parameters

vm	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 522 of file __vm-vmx.h.

9.45.3.8 I4_umword_t I4_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 439 of file __vm-vmx.h.

Referenced by L4_vm_vmx_read().

Here is the caller graph for this function:



9.45.3.9 I4_uint16_t I4_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 444 of file __vm-vmx.h.

Referenced by L4_vm_vmx_read().

Here is the caller graph for this function:



9.45.3.10 I4_uint32_t I4_vm_vmx_read_32 (void * vmcs, unsigned field) [inline]

Read a 32bit VMCS field.

9.45 VM API for VMX 169

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 449 of file __vm-vmx.h.

Referenced by L4_vm_vmx_read().

Here is the caller graph for this function:



9.45.3.11 I4_uint64_t I4_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 454 of file __vm-vmx.h.

Referenced by L4_vm_vmx_read().

Here is the caller graph for this function:



9.45.3.12 I4_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.

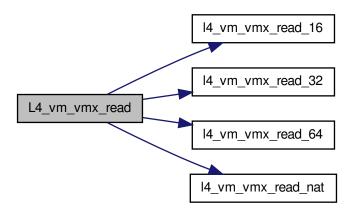
Returns

The value of the VMCS field with the given index.

Definition at line 459 of file __vm-vmx.h.

References I4_vm_vmx_read_16(), I4_vm_vmx_read_32(), I4_vm_vmx_read_64(), and I4_vm_vmx_read_nat().

Here is the call graph for this function:



9.45.3.13 void I4_vm_vmx_write_nat (void * vmcs, unsigned field, I4_umword_t val) [inline]

Write to a natural width VMCS field.

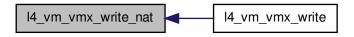
Parameters

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 472 of file __vm-vmx.h.

Referenced by I4_vm_vmx_write().

Here is the caller graph for this function:



9.45 VM API for VMX 171

9.45.3.14 void I4_vm_vmx_write_16 (void * vmcs, unsigned field, I4_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 477 of file __vm-vmx.h.

Referenced by I4_vm_vmx_write().

Here is the caller graph for this function:



9.45.3.15 void I4_vm_vmx_write_32 (void * vmcs, unsigned field, I4_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 482 of file __vm-vmx.h.

Referenced by I4_vm_vmx_write().

Here is the caller graph for this function:



9.45.3.16 void I4_vm_vmx_write_64 (void * vmcs, unsigned field, I4_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 487 of file __vm-vmx.h.

Referenced by I4_vm_vmx_write().

Here is the caller graph for this function:



9.45.3.17 void $I4_vm_vmx_write$ (void * vmcs, unsigned field, $I4_uint64_t$ val) [inline]

Write to an arbitrary VMCS field.

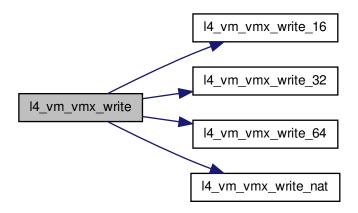
Parameters

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 493 of file __vm-vmx.h.

References I4_vm_vmx_write_16(), I4_vm_vmx_write_32(), I4_vm_vmx_write_64(), and I4_vm_vmx_write_nat().

Here is the call graph for this function:



9.46 Cache Consistency 173

9.46 Cache Consistency

Various functions for cache consistency.

Collaboration diagram for Cache Consistency:



Functions

- void l4_cache_clean_data (unsigned long start, unsigned long end) L4_NOTHROW
 Cache clean a range in D-cache.
- void l4_cache_flush_data (unsigned long start, unsigned long end) L4_NOTHROW
 Cache flush a range.
- void l4_cache_inv_data (unsigned long start, unsigned long end) L4_NOTHROW
 Cache invalidate a range.
- void I4_cache_coherent (unsigned long start, unsigned long end) L4_NOTHROW
 Make memory coherent between I-cache and D-cache.
- void I4_cache_dma_coherent (unsigned long start, unsigned long end) L4_NOTHROW
 Make memory coherent for use with external memory.
- void I4_cache_dma_coherent_full (void) L4_NOTHROW
 Make memory coherent for use with external memory.

9.46.1 Detailed Description

Various functions for cache consistency. #include <14/sys/cache.h>

9.46.2 Function Documentation

9.46.2.1 void I4_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)

Examples:

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

Definition at line 84 of file cache.h.

9.46.2.2 void I4_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)

Definition at line 91 of file cache.h.

9.46.2.3 void I4_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)

Definition at line 98 of file cache.h.

9.46.2.4 void I4_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)

Definition at line 105 of file cache.h.

9.46.2.5 void I4_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)

Definition at line 112 of file cache.h.

9.47 Memory related 175

9.47 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 20

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 I4_addr_consts_t { L4_INVALID_ADDR = ~0UL }

Address related constants.

Functions

I4_addr_t I4_trunc_page (I4_addr_t address) L4_NOTHROW

Round an address down to the next lower page boundary.

• I4_addr_t I4_trunc_size (I4_addr_t address, unsigned char bits) L4_NOTHROW

Round an address down to the next lower flex page with size bits.

• I4_addr_t I4_round_page (I4_addr_t address) L4_NOTHROW

Round address up to the next page.

• I4_addr_t I4_round_size (I4_addr_t address, unsigned char bits) L4_NOTHROW

Round address up to the next flex page with bits size.

9.47.1 Detailed Description

Memory related constants, data types and functions.

9.47.2 Macro Definition Documentation

9.47.2.1 #define L4_PAGEMASK

Mask for the page number.

Note

The most significant bits are set.

Definition at line 285 of file consts.h.

Referenced by I4_round_page(), and I4_trunc_page().

9.47.2.2 #define L4_LOG2_PAGESIZE

Number of bits used for page offset.

Size of page in log2.

Definition at line 294 of file consts.h.

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

9.47.2.3 #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 303 of file consts.h.

9.47.2.4 #define L4_SUPERPAGEMASK

Mask for the number of a large page.

Note

The most significant bits are set.

Definition at line 312 of file consts.h.

9.47 Memory related 177

9.47.2.5 #define L4_LOG2_SUPERPAGESIZE

Number of bits used as offset for a large page.

Size of large page in log2

Definition at line 320 of file consts.h.

9.47.3 Enumeration Type Documentation

```
9.47.3.1 enum I4_addr_consts_t
```

Address related constants.

Enumerator

L4_INVALID_ADDR Invalid address.

Definition at line 368 of file consts.h.

9.47.4 Function Documentation

```
9.47.4.1 I4_addr_t I4_trunc_page ( I4_addr_t address ) [inline]
```

Round an address down to the next lower page boundary.

Parameters

```
address The address to round.
```

Examples:

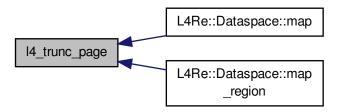
examples/libs/l4re/c++/mem_alloc/ma+rm.cc, and examples/libs/l4re/c/ma+rm.c.

Definition at line 329 of file consts.h.

References L4_PAGEMASK.

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

Here is the caller graph for this function:



9.47.4.2 I4_addr_t I4_trunc_size (I4_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 340 of file consts.h.

Referenced by L4Re::Dataspace::map_region().

Here is the caller graph for this function:



9.47.4.3 I4_addr_t I4_round_page (I4_addr_t address) [inline]

Round address up to the next page.

Parameters

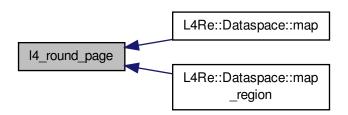
address	The address to round up.

Definition at line 350 of file consts.h.

References L4_PAGEMASK, and L4_PAGESIZE.

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

Here is the caller graph for this function:



9.47.4.4 I4_addr_t I4_round_size (I4_addr_t address, unsigned char bits) [inline]

Round address up to the next flex page with bits size.

9.47 Memory related 179

Parameters

	address	The address to round up to the next flex page.	
bits The size of the flex page (log2).		The size of the flex page (log2).	

Definition at line 361 of file consts.h.

Referenced by L4Re::Dataspace::map_region().

Here is the caller graph for this function:



9.48 Kernel Debugger

Kernel debugger related functionality.

Collaboration diagram for Kernel Debugger:



Data Structures

· class L4::Debugger

Debugger interface.

Macros

• #define enter_kdebug(text)

Enter L4 kernel debugger.

• #define asm_enter_kdebug(text)

Enter L4 kernel debugger (plain assembler version)

#define kd_display(text)

Show message with L4 kernel debugger, but do not enter debugger.

• #define ko(c)

Output character with L4 kernel debugger.

• #define enter_kdebug(text)

Enter L4 kernel debugger.

#define asm_enter_kdebug(text)

Enter L4 kernel debugger (plain assembler version)

#define kd display(text)

Show message with L4 kernel debugger, but do not enter debugger.

• #define ko(c)

Output character with L4 kernel debugger.

Functions

I4_msgtag_t I4_debugger_set_object_name (I4_cap_idx_t cap, const char *name) L4_NOTHROW
 The string name of kernel object.

• unsigned long I4_debugger_global_id (I4_cap_idx_t cap) L4_NOTHROW

Get the globally unique ID of the object behind a capability.

• unsigned long I4_debugger_kobj_to_id (I4_cap_idx_t cap, I4_addr_t kobjp) L4_NOTHROW

Get the globally unique ID of the object behind the kobject pointer.

• void outchar (char c) L4_NOTHROW

Print character.

void outstring (const char *text) L4_NOTHROW

9.48 Kernel Debugger 181

Print character string.

void outnstring (char const *text, unsigned len) L4_NOTHROW

Print character string.

void outhex32 (int number) L4_NOTHROW

Print 32 bit number (hexadecimal)

• void outhex20 (int number) L4_NOTHROW

Print 20 bit number (hexadecimal)

void outhex16 (int number) L4 NOTHROW

Print 16 bit number (hexadecimal)

• void outhex12 (int number) L4_NOTHROW

Print 12 bit number (hexadecimal)

void outhex8 (int number) L4_NOTHROW

Print 8 bit number (hexadecimal)

· void outdec (int number) L4 NOTHROW

Print number (decimal)

· char l4kd inchar (void) L4 NOTHROW

Read character from console, non blocking.

9.48.1 Detailed Description

Kernel debugger related functionality.

Attention

This API is subject to change!

This is a debugging factility, any call to any function might be invalid. Do not rely on it in any real code.

```
#include <14/sys/debugger.h>
```

9.48.2 Macro Definition Documentation

9.48.2.1 #define enter_kdebug(text)

Value:

```
asm(\
    "int $3 \n\t"\
    "jmp 1f \n\t"\
    ".ascii\\"" text "\"\n\t"\
    "1: \n\t"\
```

Enter L4 kernel debugger.

Parameters

text | Text to be shown at kernel debugger prompt

Examples:

examples/sys/singlestep/main.c.

Definition at line 41 of file kdebug.h.

9.48.2.2 #define asm_enter_kdebug(text)

Value:

```
"int $3 \n\t"\
   "jmp lf \n\t"\
   ".ascii\"" text "\"\n\t"\
   "1: \n\t"
```

Enter L4 kernel debugger (plain assembler version)

Parameters

```
text Text to be shown at kernel debugger prompt
```

Definition at line 63 of file kdebug.h.

```
9.48.2.3 #define kd_display( text )
```

Value:

```
asm(\
    "int $3 \n\t"\
    "nop \n\t"\
    "jmp 1f \n\t"\
    ".ascii \"" text "\"\n\t"\
    "1: \n\t"\
)
```

Show message with L4 kernel debugger, but do not enter debugger.

Parameters

```
text | Text to be shown
```

Definition at line 76 of file kdebug.h.

```
9.48.2.4 #define ko( c )
```

Value:

```
asm(
    "int $3 \n\t" \
    "cmpb %0,%%al \n\t" \
    : /* No output */
    : "N" (c)
```

Output character with L4 kernel debugger.

Parameters

```
c Character to be shown
```

Definition at line 92 of file kdebug.h.

9.48.2.5 #define enter_kdebug(text)

Value:

```
asm(\
   "int $3 \n\t"\
   "jmp 1f \n\t"\
   ".ascii \"" text "\"\n\t"\
   "1: \n\t"\
   )
```

Enter L4 kernel debugger.

9.48 Kernel Debugger 183

Parameters

text | Text to be shown at kernel debugger prompt

Definition at line 41 of file kdebug.h.

9.48.2.6 #define asm_enter_kdebug(text)

Value:

```
"int $3 \n\t"\
   "jmp lf \n\t"\
   ".ascii \"" text "\"\n\t"\
   "1: \n\t"
```

Enter L4 kernel debugger (plain assembler version)

Parameters

```
text | Text to be shown at kernel debugger prompt
```

Definition at line 63 of file kdebug.h.

```
9.48.2.7 #define kd_display( text )
```

Value:

```
asm(\
    "int $3 \n\t"\
    "nop \n\t"\
    "jmp 1f \n\t"\
    ".ascii \"" text "\"\n\t"\
    "1: \n\t"\
)
```

Show message with L4 kernel debugger, but do not enter debugger.

Parameters

```
text Text to be shown
```

Definition at line 76 of file kdebug.h.

```
9.48.2.8 #define ko( c )
```

Value:

```
asm(
    "int $3 \n\t" \
    "cmpb %0,%%al \n\t" \
    : /* No output */ \
    : "N" (c) \
}
```

Output character with L4 kernel debugger.

Parameters

```
c Character to be shown
```

Definition at line 92 of file kdebug.h.

9.48.3 Function Documentation

9.48.3.1 I4_msgtag_t I4_debugger_set_object_name (I4_cap_idx_t cap, const char * name) [inline]

The string name of kernel object.

9.48 Kernel Debugger 185

Parameters

сар	Capability
name	Name

This is a debugging factility, the call might be invalid.

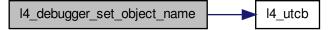
Examples:

examples/sys/aliens/main.c.

Definition at line 290 of file debugger.h.

References I4_utcb().

Here is the call graph for this function:



9.48.3.2 unsigned long I4_debugger_global_id (I4_cap_idx_t cap) [inline]

Get the globally unique ID of the object behind a capability.

Parameters

сар	Capability

Returns

 \sim 0UL on non-valid capability, ID otherwise

This is a debugging factility, the call might be invalid.

Definition at line 297 of file debugger.h.

References I4_utcb().

Here is the call graph for this function:



9.48.3.3 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

сар	Capability
kobjp	Kobject pointer

Returns

 \sim 0UL on non-valid capability or invalid kobject pointer, ID otherwise

This is a debugging factility, the call might be invalid.

Definition at line 303 of file debugger.h.

References I4_utcb().

Here is the call graph for this function:



9.48.3.4 void outchar (char c) [inline]

Print character.

Parameters

c Character	
-------------	--

9.48.3.5 void outstring (const char * text) [inline]

Print character string.

Parameters

text	Character string
text	String

Examples:

examples/sys/aliens/main.c.

9.48.3.6 void outnstring (char const * text, unsigned len) [inline]

Print character string.

Parameters

9.48 Kernel Debugger 187

text	Character string
len	Number of characters
text	String
len	Number of characters

Examples:

examples/sys/aliens/main.c.

9.48.3.7 void outhex32 (int number) [inline]

Print 32 bit number (hexadecimal)

Parameters

nui	mber	32 bit number

9.48.3.8 void outhex20 (int number) [inline]

Print 20 bit number (hexadecimal)

Parameters

number	20 bit number

9.48.3.9 void outhex16 (int number) [inline]

Print 16 bit number (hexadecimal)

Parameters

number 16 bit number

9.48.3.10 void outhex12 (int number) [inline]

Print 12 bit number (hexadecimal)

Parameters

number	12 bit number

9.48.3.11 void outhex8 (int number) [inline]

Print 8 bit number (hexadecimal)

Parameters

numl	ber 8 bit number	

9.48.3.12 void outdec (int number) [inline]

Print number (decimal)

Parameters

number	Number
--------	--------

9.48.3.13 charl4kd_inchar(void) [inline]

Read character from console, non blocking.

Returns

Input character, -1 if no character to read

9.49 Error codes 189

9.49 Error codes

Common error codes.

Collaboration diagram for Error codes:



Enumerations

```
    enum I4_error_code_t {
    L4_EOK = 0, L4_EPERM = 1, L4_ENOENT = 2, L4_EIO = 5,
    L4_EAGAIN = 11, L4_ENOMEM = 12, L4_EACCESS = 13, L4_EBUSY = 16,
    L4_EEXIST = 17, L4_ENODEV = 19, L4_EINVAL = 22, L4_ERANGE = 34,
    L4_ENAMETOOLONG = 36, L4_ENOSYS = 38, L4_EBADPROTO = 39, L4_EADDRNOTAVAIL = 99,
    L4_ERRNOMAX = 100, L4_ENOREPLY = 1000, L4_EIPC_LO = 2000, L4_EIPC_HI = 2000 + 0x1f }
```

9.49.1 Detailed Description

Common error codes. #include <14/sys/err.h>

9.49.2 Enumeration Type Documentation

```
9.49.2.1 enum I4_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_EAGAIN Try again.

L4_ENOMEM No memory.

L4_EACCESS Permission denied.

L4_EBUSY Object currently busy, try later.

L4_EEXIST Already exists.

L4_ENODEV No such thing.

L4_EINVAL Invalid argument.

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_EIPC_LO Communication error-range low.
- L4_EIPC_HI Communication error-range high.

Definition at line 41 of file err.h.

9.50 Factory 191

9.50 Factory

A factory is used to create all kinds of kernel objects.

Collaboration diagram for Factory:



Data Structures

· class L4::Factory

C++ L4 Factory, to create all kinds of kernel objects.

Functions

I4_msgtag_t I4_factory_create_task (I4_cap_idx_t factory, I4_cap_idx_t target_cap, I4_fpage_t const utcb_-area) L4_NOTHROW

Create a new task.

• I4_msgtag_t I4_factory_create_thread (I4_cap_idx_t factory, I4_cap_idx_t target_cap) L4_NOTHROW

• I4_msgtag_t I4_factory_create_factory (I4_cap_idx_t factory, I4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW

Create a new factory.

Create a new thread.

• I4_msgtag_t I4_factory_create_gate (I4_cap_idx_t factory, I4_cap_idx_t target_cap, I4_cap_idx_t thread_cap, I4_umword_t label) L4_NOTHROW

Create a new IPC gate.

- I4_msgtag_t I4_factory_create_irq (I4_cap_idx_t factory, I4_cap_idx_t target_cap) L4_NOTHROW
 Create a new IRQ.
- I4_msgtag_t I4_factory_create_vm (I4_cap_idx_t factory, I4_cap_idx_t target_cap) L4_NOTHROW
 Create a new virtual machine.

9.50.1 Detailed Description

A factory is used to create all kinds of kernel objects. #include <14/sys/factory.h>

A factory provides the means to create all kinds of kernel objects. 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.

9.50.2 Function Documentation

9.50.2.1 I4_msgtag_t I4_factory_create_task (I4_cap_idx_t factory, I4_cap_idx_t target_cap, I4_fpage_t const utcb_area) [inline]

Create a new task.

9.50 Factory 193

Parameters

factory	Capability selector for factory to use for creation.
target_cap	Capability selector for the root capability of the new task.
utcb_area	Flexpage that describes the area for the UTCBs of the new task

Note

The size of the UTCB area specifies indirectly the maximum number of UTCBs available for this task and cannot be changed afterwards.

Returns

Syscall return tag

See Also

Task

Definition at line 306 of file factory.h.

References I4_utcb().

Here is the call graph for this function:



9.50.2.2 I4_msgtag_t I4_factory_create_thread (I4_cap_idx_t factory, I4_cap_idx_t target_cap) [inline]

Create a new thread.

Parameters

	factory	Capability selector for factory to use for creation.
Ī	target_cap	Capability selector for the root capability of the new thread.

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 313 of file factory.h.

References I4_utcb().

Here is the call graph for this function:



9.50.2.3 I4_msgtag_t I4_factory_create_factory (I4_cap_idx_t factory, I4_cap_idx_t target_cap, unsigned long limit)
[inline]

Create a new factory.

Parameters

factory	Capability selector for factory to use for creation.
target_cap	Capability selector for the root capability of the new factory.
limit	Limit for the new factory in bytes

Note

The limit of the new factory is subtracted from the available amount of the factory used for creation.

Returns

Syscall return tag

Definition at line 320 of file factory.h.

References I4_utcb().

Here is the call graph for this function:



9.50.2.4 I4_msgtag_t I4_factory_create_gate (I4_cap_idx_t factory, I4_cap_idx_t target_cap, I4_cap_idx_t thread_cap, I4_umword_t label) [inline]

Create a new IPC gate.

9.50 Factory 195

Parameters

factory	Capability selector for factory to use for creation.
target_cap	Capability selector for the root capability of the new IPC gate.
thread_cap	Thread to bind the gate to
label	Label of the gate

Returns

Syscall return tag

See Also

IPC-Gate API

Definition at line 328 of file factory.h.

References I4_utcb().

Here is the call graph for this function:



9.50.2.5 I4_msgtag_t I4_factory_create_irq (I4_cap_idx_t factory, I4_cap_idx_t target_cap) [inline]

Create a new IRQ.

Parameters

factory	Capability selector for factory to use for creation.
target_cap	Capability selector for the root capability of the new IRQ.

Returns

Syscall return tag

See Also

IRQs

Examples:

examples/sys/isr/main.c.

Definition at line 336 of file factory.h.

References I4_utcb().

Here is the call graph for this function:



9.50.2.6 I4_msgtag_tI4_factory_create_vm(I4_cap_idx_t factory, I4_cap_idx_t target_cap) [inline]

Create a new virtual machine.

Parameters

factory	Capability selector for factory to use for creation.
target_cap	Capability selector for the root capability of the new VM.

Returns

Syscall return tag

See Also

Virtual Machines

Definition at line 343 of file factory.h.

References I4_utcb().

Here is the call graph for this function:

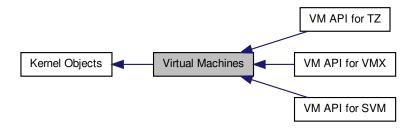


9.51 Virtual Machines

9.51 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.

Data Structures

class L4::Vm

Virtual machine.

9.51.1 Detailed Description

Virtual Machine API.

9.52 Interrupt controller

The ICU class.

Collaboration diagram for Interrupt controller:



Data Structures

• struct I4 icu info t

Info structure for an ICU.

· class L4::lcu

C++ version of an interrupt controller.

· class L4::lcu::Info

Info for an ICU.

Typedefs

typedef struct I4_icu_info_t I4_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

- I4_msgtag_t I4_icu_bind (I4_cap_idx_t icu, unsigned irqnum, I4_cap_idx_t irq) L4_NOTHROW
 Bind an interrupt vector of an interrupt controller to an interrupt object.
- I4_msgtag_t I4_icu_unbind (I4_cap_idx_t icu, unsigned irqnum, I4_cap_idx_t irq) L4_NOTHROW

 Remove binding of an interrupt vector from the interrupt controller object.
- I4_msgtag_t I4_icu_set_mode (I4_cap_idx_t icu, unsigned irqnum, I4_umword_t mode) L4_NOTHROW
 Set mode of interrupt.
- I4_msgtag_t I4_icu_info (I4_cap_idx_t icu, I4_icu_info_t *info) L4_NOTHROW

 Get info about capabilites of ICU.
- I4_msgtag_t I4_icu_msi_info (I4_cap_idx_t icu, unsigned irqnum, I4_umword_t *msg) L4_NOTHROW Get MSI info about IRQ.
- I4_msgtag_t I4_icu_unmask (I4_cap_idx_t icu, unsigned irqnum, I4_umword_t *label, I4_timeout_t to) L4_N-OTHROW

Unmask an IRQ vector.

• I4_msgtag_t I4_icu_mask (I4_cap_idx_t icu, unsigned irqnum, I4_umword_t *label, I4_timeout_t to) L4_NO-THROW

Mask an IRQ vector.

9.52.1 Detailed Description

The ICU class. #include <14/sys/icu.h>

9.52.2 Typedef Documentation

9.52.2.1 typedef struct I4_icu_info_t I4_icu_info_t

Info structure for an ICU.

This structure contains information about the features of an ICU.

See Also

I4_icu_info().

9.52.3 Enumeration Type Documentation

9.52.3.1 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 I4_icu_bind() and I4_icu_unbind() functions to denote that the IRQ number is meant to be an MSI.

Definition at line 44 of file icu.h.

9.52.4 Function Documentation

9.52.4.1 I4_msgtag_t I4_icu_bind (I4_cap_idx_t icu, unsigned irqnum, I4_cap_idx_t irq) [inline]

Bind an interrupt vector of an interrupt controller to an interrupt object.

Parameters

icu	ICU to use.
irqnum	IRQ vector at the ICU.
irq	IRQ capability to bind the IRQ to.

Returns

Syscall return tag

Examples:

examples/sys/isr/main.c.

Definition at line 423 of file icu.h.

References I4_utcb().

Here is the call graph for this function:



9.52.4.2 I4_msgtag_t I4_icu_unbind (I4_cap_idx_t icu, unsigned irqnum, I4_cap_idx_t irq) [inline]

Remove binding of an interrupt vector from the interrupt controller object.

Parameters

icu	ICU to use.
irqnum	IRQ vector at the ICU.
irq	IRQ object to remove from the ICU.

Returns

Syscall return tag

Definition at line 427 of file icu.h.

References I4_utcb().

Here is the call graph for this function:



9.52.4.3 I4_msgtag_t I4_icu_set_mode (I4_cap_idx_t icu, unsigned irqnum, I4_umword_t mode) [inline]

Set mode of interrupt.

Parameters

icu	ICU to use.
irqnum	IRQ vector at the ICU.
mode	Mode, see L4_irq_mode.

Returns

Syscall return tag

Definition at line 449 of file icu.h.

References I4_utcb().

Here is the call graph for this function:



9.52.4.4 I4_msgtag_t I4_icu_info (I4_cap_idx_t icu, I4_icu_info_t * info) [inline]

Get info about capabilites of ICU.

Parameters

icu	ICU to use.
info	Pointer to an info structure to be filled with information.

Returns

Syscall return tag

Definition at line 431 of file icu.h.

References I4_utcb().

Here is the call graph for this function:



9.52.4.5 I4_msgtag_t I4_icu_msi_info (I4_cap_idx_t icu, unsigned irqnum, I4_umword_t * msg) [inline]

Get MSI info about IRQ.

Parameters

icu	ICU to use.
irqnum	IRQ vector at the ICU.
msg	Pointer to a word to receive the message that must be used for the PCI devices MSI message.

Returns

Syscall return tag

Definition at line 435 of file icu.h.

References I4_utcb().

Here is the call graph for this function:



9.52.4.6 I4_msgtag_t I4_icu_unmask (I4_cap_idx_t icu, unsigned irqnum, I4_umword_t * label, I4_timeout_t to) [inline]

Unmask an IRQ vector.

Parameters

icu	ICU to use.
irqnum	IRQ vector 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 439 of file icu.h.

References I4_utcb().



9.52.4.7 | I4_msgtag_t I4_icu_mask (I4_cap_idx_t icu, unsigned irqnum, I4_umword_t * label, I4_timeout_t to) [inline]

Mask an IRQ vector.

Parameters

icu	ICU to use.
irqnum	IRQ vector 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 444 of file icu.h.

References I4_utcb().

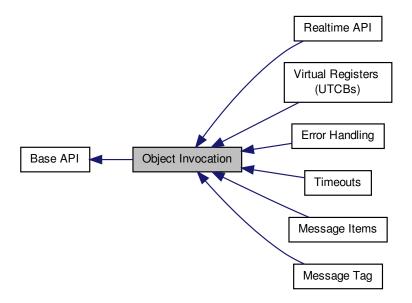


9.53 Object Invocation 205

9.53 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 I4_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

 I4_msgtag_t I4_ipc_send (I4_cap_idx_t dest, I4_utcb_t *utcb, I4_msgtag_t tag, I4_timeout_t timeout) L4_N-OTHROW

Send a message to an object (do not wait for a reply).

• I4_msgtag_t I4_ipc_wait (I4_utcb_t *utcb, I4_umword_t *label, I4_timeout_t timeout) L4_NOTHROW Wait for an incoming message from any possible sender.

- I4_msgtag_t I4_ipc_receive (I4_cap_idx_t object, I4_utcb_t *utcb, I4_timeout_t timeout) L4_NOTHROW

 Wait for a message from a specific source.
- I4_msgtag_t I4_ipc_call (I4_cap_idx_t object, I4_utcb_t *utcb, I4_msgtag_t tag, I4_timeout_t timeout) L4_N-OTHROW

Object call (usual invocation).

 I4_msgtag_t I4_ipc_reply_and_wait (I4_utcb_t *utcb, I4_msgtag_t tag, I4_umword_t *label, I4_timeout_t timeout) L4_NOTHROW

Reply and wait operation (uses the reply capability).

I4_msgtag_t I4_ipc_send_and_wait (I4_cap_idx_t dest, I4_utcb_t *utcb, I4_msgtag_t tag, I4_umword_t *label, I4_timeout t timeout) L4_NOTHROW

Send a message and do an open wait.

• I4_msgtag_t I4_ipc (I4_cap_idx_t dest, I4_utcb_t *utcb, I4_umword_t flags, I4_umword_t slabel, I4_msgtag_t tag, I4_umword_t *rlabel, I4_timeout_t timeout) L4_NOTHROW

Generic L4 object invocation.

• 14 msgtag t 14 ipc sleep (14 timeout t timeout) L4 NOTHROW

Sleep for an amount of time.

• int I4_sndfpage_add (I4_fpage_t const snd_fpage, unsigned long snd_base, I4_msgtag_t *tag) L4_NOTHR-OW

Add a flex-page to be sent to the UTCB.

9.53.1 Detailed Description

API for L4 object invocation. #include <14/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 (I4_ipc_call()). However, there are a lot more flavours available that have a semantics depending on the object.

See Also

IPC-Gate API

9.53.2 Enumeration Type Documentation

9.53.2.1 enum I4_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_REC-V) 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 45 of file consts.h.

9.53.3 Function Documentation

9.53.3.1 I4_msgtag_t I4_ipc_send (I4_cap_idx_t dest, I4_utcb_t * utcb, I4_msgtag_t tag, I4_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 I4_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 I4_ipc_call().

Examples:

examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 93 of file ipc.h.

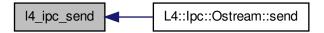
References I4_ipc(), L4_SYSF_SEND, and I4_msgtag_t::raw.

Referenced by L4::lpc::Ostream::send().

Here is the call graph for this function:



Here is the caller graph for this function:



 $\textbf{9.53.3.2} \quad \textbf{I4_msgtag_t I4_ipc_wait (I4_utcb_t*\textit{utcb}, I4_umword_t*\textit{label}, I4_timeout_t \textit{timeout}) \quad \texttt{[inline]}$

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 |4_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 I4_ipc_reply_and_wait().

Examples:

examples/sys/ipc/ipc_example.c.

Definition at line 101 of file ipc.h.

References L4_INVALID_CAP, I4_ipc(), L4_SYSF_WAIT, and I4_msgtag_t::raw.

9.53 Object Invocation 209

Referenced by L4::lpc::lstream::wait().

Here is the call graph for this function:



Here is the caller graph for this function:



9.53.3.3 I4_msgtag_t I4_ipc_receive (I4_cap_idx_t object, I4_utcb_t * utcb, I4_timeout_t timeout_) [inline]

Wait for a message from a specific source.

Parameters

object	Object to receive a message from.
timeout	Timeout pair (see I4_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:

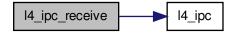
examples/sys/aliens/main.c, examples/sys/singlestep/main.c, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 110 of file ipc.h.

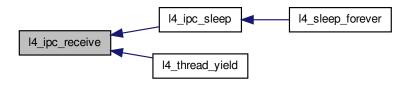
References I4_ipc(), L4_SYSF_RECV, and I4_msgtag_t::raw.

Referenced by I4_ipc_sleep(), and I4_thread_yield().

Here is the call graph for this function:



Here is the caller graph for this function:



9.53.3.4 I4_msgtag_t I4_ipc_call (I4_cap_idx_t object, I4_utcb_t * utcb, I4_msgtag_t tag, I4_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 I4_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 68 of file ipc.h.

References I4_ipc(), L4_SYSF_CALL, and I4_msgtag_t::raw.

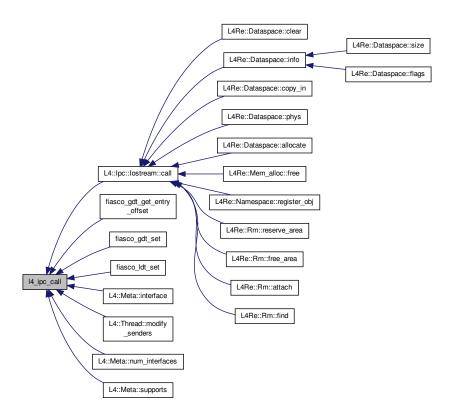
9.53 Object Invocation 211

Referenced by L4::lpc::lostream::call(), fiasco_gdt_get_entry_offset(), fiasco_gdt_set(), fiasco_ldt_set(), L4::Meta::interface(), L4::Thread::modify_senders(), L4::Meta::num_interfaces(), and L4::Meta::supports().

Here is the call graph for this function:



Here is the caller graph for this function:



9.53.3.5 I4_msgtag_t I4_ipc_reply_and_wait (I4_utcb_t * utcb, I4_msgtag_t tag, I4_umword_t * label, I4_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.

Return values

label	Label assigned to the source object of the received message.

Parameters

timed	out	Timeout pair (see I4_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.

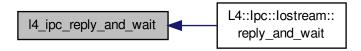
Definition at line 76 of file ipc.h.

References L4_INVALID_CAP, I4_ipc(), L4_SYSF_REPLY_AND_WAIT, and I4_msgtag_t::raw.

Referenced by L4::lpc::lostream::reply_and_wait().

Here is the call graph for this function:





9.53 Object Invocation 213

9.53.3.6 I4_msgtag_t I4_ipc_send_and_wait (I4_cap_idx_t dest, I4_utcb_t * utcb, I4_msgtag_t tag, I4_umword_t * label, I4_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.

Return values

label	Label assigned to the source object of the receive phase.

Parameters

timeout	Timeout pair (see I4_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 84 of file ipc.h.

References I4_ipc(), L4_SYSF_SEND_AND_WAIT, and I4_msgtag_t::raw.

Here is the call graph for this function:



9.53.3.7 I4_msgtag_t I4_ipc (I4_cap_idx_t dest, I4_utcb_t * utcb, I4_umword_t flags, I4_umword_t slabel, I4_msgtag_t tag, I4_umword_t * rlabel, I4_timeout_t timeout_) [inline]

Generic L4 object invocation.

Parameters

dest	Destination object.
utcb	UTCB of the caller.
flags	Invocation flags (see 4_syscall_flags_t).
slabel	Send label if applicable (may be seen by the receiver).
tag	Sending message tag.

Return values

icturii valacs

9.53 Object Invocation 215

	rlabel	Receiving label.
Parameters		
raiailleleis		
timeout	Timeout p	air (see 4 timeout t).

Returns

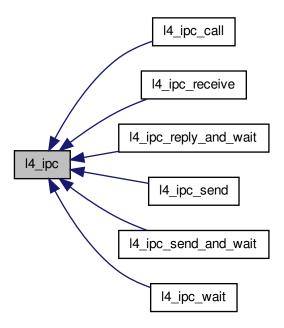
return tag

Definition at line 34 of file ipc.h.

References I4_msgtag_t::raw.

Referenced by I4_ipc_call(), I4_ipc_receive(), I4_ipc_reply_and_wait(), I4_ipc_send(), I4_ipc_send_and_wait(), and I4_ipc_wait().

Here is the caller graph for this function:



9.53.3.8 I4_msgtag_t I4_ipc_sleep (I4_timeout_t timeout) [inline]

Sleep for an amount of time.

Parameters

timeout	Timeout pair (see I4_timeout_t, the receive part matters).

Returns

error code:

- L4_IPC_RETIMEOUT: success
- L4_IPC_RECANCELED woken up by a different thread (I4_thread_ex_regs()).

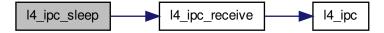
The invoking thread waits until the timeout is expired or the wait was aborted by another thread by I4_thread_ex_regs().

Definition at line 28 of file ipc-impl.h.

References L4_INVALID_CAP, and I4_ipc_receive().

Referenced by I4_sleep_forever().

Here is the call graph for this function:



Here is the caller graph for this function:



9.53.3.9 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

end fnage	Floy nago
snd_tpage	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.	

9.53 Object Invocation 217

Returns

0 on success, negative error code otherwise

Definition at line 486 of file ipc.h.

References I4_utcb().



9.54 Error Handling

Error handling for L4 object invocation.

Collaboration diagram for Error Handling:



Enumerations

```
    enum I4_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_RECVPFTO = 0x0d,
    L4_IPC_SERCVPFTO = 0x0c, L4_IPC_REABORTED = 0x0f, L4_IPC_SEABORTED = 0x0e, L4_IPC_REMSGCUT = 0x09,
    L4_IPC_SEMSGCUT = 0x08 }
```

Error codes in the error TCR.

Functions

- I4_umword_t I4_ipc_error (I4_msgtag_t tag, I4_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 I4_ipc_is_rcv_error (I4_utcb_t *utcb) L4_NOTHROW

Returns whether an error occurred in receive phase of an invocation.

• int I4_ipc_error_code (I4_utcb_t *utcb) L4_NOTHROW

Get the error condition of the last invocation from the TCR.

9.54.1 Detailed Description

Error handling for L4 object invocation. #include <14/sys/ipc.h>

9.54.2 Enumeration Type Documentation

9.54.2.1 enum I4_ipc_tcr_error_t

Error codes in the error TCR.

The error codes are accessible via the error TCR, see I4_thread_regs_t.error.

9.54 Error Handling 219

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.

9.54.3 Function Documentation

9.54.3.1 I4_umword_t I4_ipc_error(I4_msgtag_t tag, I4_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 14_ipc_tcr_error_t).

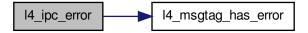
Examples:

examples/sys/ipc/ipc_example.c, examples/sys/isr/main.c, and examples/sys/start-with-exc/main.c.

Definition at line 430 of file ipc.h.

References I4_thread_regs_t::error, L4_IPC_ERROR_MASK, and I4_msgtag_has_error().

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



Here is the caller graph for this function:



9.54.3.2 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/client.cc, 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 447 of file ipc.h.

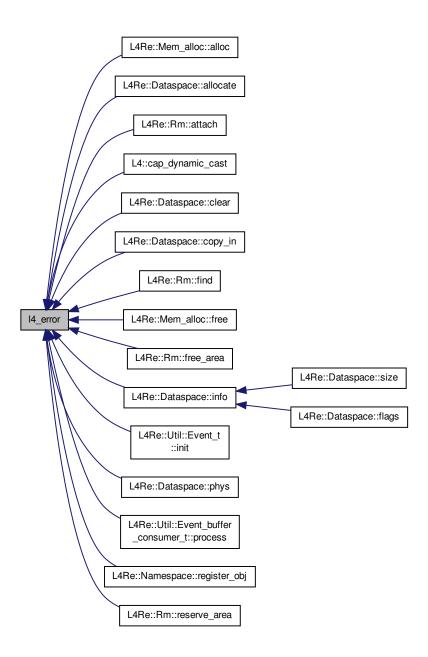
References I4_utcb().

Referenced by L4Re::Mem_alloc::alloc(), L4Re::Dataspace::allocate(), L4Re::Rm::attach(), L4::cap_dynamic_cast(), L4Re::Dataspace::clear(), L4Re::Dataspace::copy_in(), L4Re::Rm::find(), L4Re::Mem_alloc::free(), L4Re::Pm::free_area(), L4Re::Dataspace::info(), L4Re::Util::Event_t< PAYLOAD >::init(), L4Re::Dataspace::phys(), L4-Re::Util::Event_buffer_consumer_t< PAYLOAD >::process(), L4Re::Namespace::register_obj(), and L4Re::Rm::reserve_area().



9.54 Error Handling 221

Here is the caller graph for this function:



Returns whether an error occurred in send phase of an invocation.

Precondition

I4_msgtag_has_error(tag) == true

Parameters

utcb	UTCB to check.
------	----------------

Returns

Boolean value.

Definition at line 453 of file ipc.h.

References I4_thread_regs_t::error.

```
9.54.3.4 intl4_ipc_is_rcv_error(l4_utcb_t * utcb) [inline]
```

Returns whether an error occurred in receive phase of an invocation.

Precondition

```
I4_msgtag_has_error(tag) == true
```

Parameters

```
utcb UTCB to check.
```

Returns

Boolean value.

Definition at line 456 of file ipc.h.

References I4_thread_regs_t::error.

```
9.54.3.5 int l4_ipc_error_code ( l4_utcb_t * utcb ) [inline]
```

Get the error condition of the last invocation from the TCR.

Precondition

```
I4_msgtag_has_error(tag) == true
```

Parameters

utcb	UTCB to check.

Returns

Error condition of type I4_ipc_tcr_error_t.

Definition at line 459 of file ipc.h.

References I4_thread_regs_t::error, and L4_IPC_ERROR_MASK.

9.55 Realtime API 223

9.55 Realtime API

Collaboration diagram for Realtime API:



9.56 IRQs

The IRQ and ICU class.

Collaboration diagram for IRQs:



Data Structures

· class L4::Irq

C++ version of an L4 IRQ.

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 }
```

Functions

- I4_msgtag_t I4_irq_attach (I4_cap_idx_t irq, I4_umword_t label, I4_cap_idx_t thread) L4_NOTHROW
 Attach to an interrupt source.
- I4_msgtag_t I4_irq_chain (I4_cap_idx_t irq, I4_umword_t label, I4_cap_idx_t slave) L4_NOTHROW

 Chain an IRQ to another master IRQ source.
- I4_msgtag_t I4_irq_detach (I4_cap_idx_t irq) L4_NOTHROW

Detach from an interrupt source.

• I4_msgtag_t I4_irq_trigger (I4_cap_idx_t irq) L4_NOTHROW

Trigger an IRQ.

I4_msgtag_t I4_irq_receive (I4_cap_idx_t irq, I4_timeout_t to) L4_NOTHROW

Unmask and wait for specified IRQ.

- I4_msgtag_t I4_irq_wait (I4_cap_idx_t irq, I4_umword_t *label, I4_timeout_t to) L4_NOTHROW
 - Unmask IRQ and wait for any message.
- I4_msgtag_t I4_irq_unmask (I4_cap_idx_t irq) L4_NOTHROW

Unmask IRQ.

9.56 IRQs 225

9.56.1 Detailed Description

The IRQ and ICU class. #include <14/sys/irq.h>

The IRQ class provides access to abstract interrupts provided by the micro kernel. Interrupts may be hardware interrupts provided by the platform interrupt controller, virtual device interrupts provided by the micro kernel virtual devices (virtual serial or trace buffer), or IRQs (virtual interrupts that can be triggered by user programs).

IRQ objects can be created using a Factory, see Factory (I4_factory_create_irq()).

9.56.2 Enumeration Type Documentation

9.56.2.1 enum L4_irq_mode

Interrupt attributes.

Enumerator

L4_IRQ_F_NONE Flow types. None

L4_IRQ_F_LEVEL Level triggered.

L4_IRQ_F_EDGE Edge triggered.

L4_IRQ_F_POS Positive trigger.

L4_IRQ_F_NEG Negative trigger.

L4_IRQ_F_BOTH Both edges trigger.

L4_IRQ_F_LEVEL_HIGH Level high trigger.

L4_IRQ_F_LEVEL_LOW Level low trigger.

L4_IRQ_F_POS_EDGE Positive edge trigger.

L4_IRQ_F_NEG_EDGE Negative edge trigger.

L4_IRQ_F_BOTH_EDGE Both edges trigger.

L4_IRQ_F_MASK Mask.

L4_IRQ_F_SET_WAKEUP Wakeup source? Use irq as wakeup source

L4_IRQ_F_CLEAR_WAKEUP Do not use irq as wakeup source.

Definition at line 61 of file icu.h.

9.56.3 Function Documentation

9.56.3.1 I4_msgtag_t I4_irq_attach (I4_cap_idx_t irq, I4_umword_t label, I4_cap_idx_t thread) [inline]

Attach to an interrupt source.

Parameters

	irq	IRQ to attach to.
	label	Identifier of the IRQ.
Ì	thread	Thread to attach the interrupt to.

Returns

Syscall return tag

Examples:

examples/sys/isr/main.c.

Definition at line 281 of file irq.h.

References I4 utcb().

Here is the call graph for this function:



9.56.3.2 I4_msgtag_t I4_irq_chain (I4_cap_idx_t irq, I4_umword_t label, I4_cap_idx_t slave) [inline]

Chain an IRQ to another master IRQ source.

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.

Parameters

irq	The master IRQ object.
label	Identifier of the IRQ.
slave	The slave that shall be attached to the master.

Returns

Syscall return tag

Definition at line 288 of file irq.h.

References I4_utcb().

Here is the call graph for this function:



9.56.3.3 I4_msgtag_t I4_irq_detach (I4_cap_idx_t irq) [inline]

Detach from an interrupt source.

9.56 IRQs 227

Parameters

irq	IRQ to detach from.

Returns

Syscall return tag

Examples:

examples/sys/isr/main.c.

Definition at line 295 of file irq.h.

References I4_utcb().

Here is the call graph for this function:



9.56.3.4 I4_msgtag_t I4_irq_trigger (I4_cap_idx_t irq) [inline]

Trigger an IRQ.

Parameters

irq IRQ to trigger.

Precondition

irq must be a reference to an IRQ.

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 I4_error() will return an error value from the message tag which still contains the IRQ protocol used for the send operation.

Use I4_ipc_error() to check for (send) errors.

Definition at line 301 of file irq.h.

References I4_utcb().

Here is the call graph for this function:



9.56.3.5 I4_msgtag_t I4_irq_receive (I4_cap_idx_t irq, I4_timeout_t to) [inline]

Unmask and wait for specified IRQ.

Parameters

irq	IRQ to wait for.
to	Timeout.

Returns

Syscall return tag

Examples:

examples/sys/isr/main.c.

Definition at line 307 of file irq.h.

References I4_utcb().

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

Here is the call graph for this function:





9.56 IRQs 229

9.56.3.6 I4_msgtag_t I4_irq_wait (I4_cap_idx_t irq, I4_umword_t * label, I4_timeout_t to) [inline]

Unmask IRQ and wait for any message.

Parameters

irq	IRQ to wait for.
label	Receive label.
to	Timeout.

Returns

Syscall return tag

Definition at line 313 of file irq.h.

References I4_utcb().

Here is the call graph for this function:



9.56.3.7 I4_msgtag_t I4_irq_unmask(I4_cap_idx_t irq) [inline]

Unmask IRQ.

Parameters

irq	IRQ to unmask.

Returns

Syscall return tag

Note

I4_irq_wait and I4_irq_receive are doing the unmask themselves.

Definition at line 320 of file irq.h.

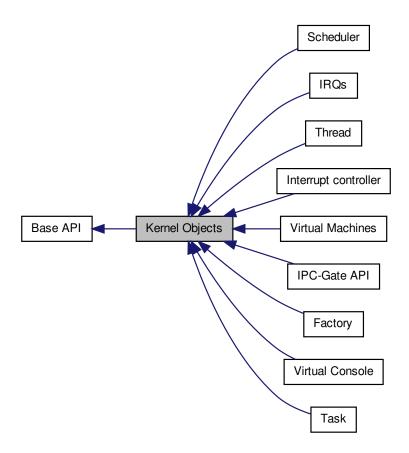
References I4_utcb().



9.57 Kernel Objects

API of kernel objects.

Collaboration diagram for Kernel Objects:



Modules

Factory

A factory is used to create all kinds of kernel objects.

IPC-Gate API

Secure comminication object.

• IRQs

The IRQ and ICU class.

· Interrupt controller

The ICU class.

Scheduler

Scheduler object.

Task

Class definition of the Task kernel object.

Thread

Thread object.

9.57 Kernel Objects 231

· 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, referred to by capabilities.

• class L4::Meta

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

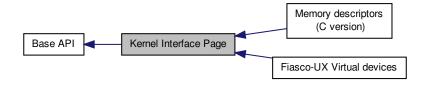
9.57.1 Detailed Description

```
API of kernel objects. #include <14/sys/capability> #include <14/sys/kernel_object.h>
```

9.58 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 I4_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 I4_kernel_info_t I4_kernel_info_t

L4 Kernel Interface Page.

• typedef struct I4_kernel_info_t I4_kernel_info_t

L4 Kernel Interface Page.

Functions

• I4_umword_t I4_kip_version (I4_kernel_info_t *kip) L4_NOTHROW

Get the kernel version.

const char * I4_kip_version_string (I4_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.

- I4_cpu_time_t I4_kip_clock (I4_kernel_info_t *kip) L4_NOTHROW Return clock value from the KIP.
- I4_umword_t I4_kip_clock_lw (I4_kernel_info_t *kip) L4_NOTHROW

Return least significant machine word of clock value from the KIP.

9.58.1 Detailed Description

Kernel Interface Page. C interface for the Kernel Interface Page:

```
#include <14/sys/kip.h>
```

C++ interface for the Kernel Interface Page:

```
#include <14/sys/kip>
```

9.58.2 Function Documentation

```
9.58.2.1 I4_umword_t I4_kip_version ( I4_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 122 of file kip.h.

```
9.58.2.2 const char * I4_kip_version_string ( I4_kernel_info_t * kip ) [inline]
```

Get the kernel version string.

Parameters

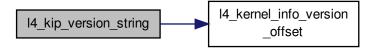
kip	Kernel Info Page.

Returns

Kernel version string.

Definition at line 126 of file kip.h.

References I4_kernel_info_version_offset().



9.58.2.3 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).
```

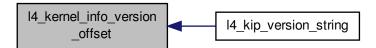
Returns

offset of version_strings relative to the KIP base address, in bytes.

Definition at line 130 of file kip.h.

Referenced by I4_kip_version_string().

Here is the caller graph for this function:



9.58.2.4 I4_cpu_time_t I4_kip_clock (I4_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 134 of file kip.h.

References I4_mb().



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 155 of file kip.h.

References I4_mb().

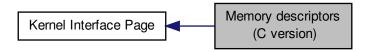
Here is the call graph for this function:



9.59 Memory descriptors (C version)

C Interface for KIP memory descriptors.

Collaboration diagram for Memory descriptors (C version):



Data Structures

struct I4_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 I4_mem_type_t {
 I4_mem_type_undefined = 0x0, I4_mem_type_conventional = 0x1, I4_mem_type_reserved = 0x2, I4_mem_type_dedicated = 0x3,
 I4_mem_type_shared = 0x4, I4_mem_type_bootloader = 0xe, I4_mem_type_archspecific = 0xf }
 Type of a memory descriptor.

Functions

- I4_kernel_info_mem_desc_t * I4_kernel_info_get_mem_descs (I4_kernel_info_t *kip) L4_NOTHROW
 Get pointer to memory descriptors from KIP.
- unsigned I4_kernel_info_get_num_mem_descs (I4_kernel_info_t *kip) L4_NOTHROW
 Get number of memory descriptors in KIP.
- void I4_kernel_info_set_mem_desc (I4_kernel_info_mem_desc_t *md, I4_addr_t start, I4_addr_t end, unsigned type, unsigned virt, unsigned sub_type) L4_NOTHROW

Populate a memory descriptor.

- I4_umword_t I4_kernel_info_get_mem_desc_start (I4_kernel_info_mem_desc_t *md) L4_NOTHROW

 Get start address of the region described by the memory descriptor.
- I4_umword_t I4_kernel_info_get_mem_desc_end (I4_kernel_info_mem_desc_t *md) L4_NOTHROW Get end address of the region described by the memory descriptor.
- I4_umword_t I4_kernel_info_get_mem_desc_type (I4_kernel_info_mem_desc_t *md) L4_NOTHROW Get type of the memory region.
- I4_umword_t I4_kernel_info_get_mem_desc_subtype (I4_kernel_info_mem_desc_t *md) L4_NOTHROW

Get sub-type of memory region.

• I4_umword_t I4_kernel_info_get_mem_desc_is_virtual (I4_kernel_info_mem_desc_t *md) L4_NOTHROW Get virtual flag of the memory descriptor.

9.59.1 Detailed Description

C Interface for KIP memory descriptors. #include <14/sys/memdesc.h>

This module contains the C functions to access the memory descriptor in the kernel interface page (KIP).

9.59.2 Typedef Documentation

```
9.59.2.1 typedef struct I4_kernel_info_mem_desc_t I4_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.

9.59.3 Enumeration Type Documentation

```
9.59.3.1 enum I4_mem_type_t
```

Type of a memory descriptor.

Enumerator

```
I4_mem_type_undefined Undefined, unused descriptor.
```

14_mem_type_conventional Conventional memory.

14_mem_type_reserved Reserved memory for kernel etc.

14_mem_type_dedicated Dedicated memory (some device memory)

14_mem_type_shared Shared memory (not implemented)

I4_mem_type_bootloader Memory owned by the boot loader.

I4_mem_type_archspecific Architecture specific memory (e.g., ACPI memory)

Definition at line 44 of file memdesc.h.

9.59.4 Function Documentation

```
9.59.4.1 unsigned I4_kernel_info_get_num_mem_descs ( I4_kernel_info_t * kip ) [inline]
```

Get number of memory descriptors in KIP.

Returns

Number of memory descriptors.

Definition at line 178 of file memdesc.h.

```
9.59.4.2 void I4_kernel_info_set_mem_desc ( I4_kernel_info_mem_desc_t * md, I4_addr_t start, I4_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 185 of file memdesc.h.

9.59.4.3 I4_umword_t I4_kernel_info_get_mem_desc_start(I4_kernel_info_mem_desc_t * md) [inline]

Get start address of the region described by the memory descriptor.

Returns

Start address.

Definition at line 200 of file memdesc.h.

9.59.4.4 I4_umword_t I4_kernel_info_get_mem_desc_end (I4_kernel_info_mem_desc_t * md) [inline]

Get end address of the region described by the memory descriptor.

Returns

End address.

Definition at line 207 of file memdesc.h.

9.59.4.5 I4_umword_t I4_kernel_info_get_mem_desc_type(I4_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 214 of file memdesc.h.

9.59.4.6 I4_umword_t I4_kernel_info_get_mem_desc_subtype (I4_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 I4_mem_type_archspecific) and has architecture specific meaning.

Definition at line 221 of file memdesc.h.

9.59.4.7 I4_umword_t I4_kernel_info_get_mem_desc_is_virtual (I4_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 228 of file memdesc.h.

9.60 Scheduler 241

9.60 Scheduler

Scheduler object.

Collaboration diagram for Scheduler:



Data Structures

• struct I4_sched_cpu_set_t

CPU sets.

· struct I4 sched param t

Scheduler parameter set.

Typedefs

- typedef struct I4_sched_cpu_set_t I4_sched_cpu_set_t
 - CPU sets
- typedef struct I4_sched_param_t I4_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

- I4_sched_cpu_set_t I4_sched_cpu_set (I4_umword_t offset, unsigned char granularity, I4_umword_t map L4-_DEFAULT_PARAM(1)) L4_NOTHROW
- I4_msgtag_t I4_scheduler_info (I4_cap_idx_t scheduler, I4_umword_t *cpu_max, I4_sched_cpu_set_t *cpus)
 L4_NOTHROW

Get scheduler information.

 I4_sched_param_t I4_sched_param (unsigned prio, I4_cpu_time_t quantum L4_DEFAULT_PARAM(0)) L4_-NOTHROW

Construct scheduler parameter.

• I4_msgtag_t I4_scheduler_run_thread (I4_cap_idx_t scheduler, I4_cap_idx_t thread, I4_sched_param_t const *sp) L4_NOTHROW

Run a thread on a Scheduler.

 I4_msgtag_t I4_scheduler_idle_time (I4_cap_idx_t scheduler, I4_sched_cpu_set_t const *cpus) L4_NOTHR-OW

Query idle time of a CPU, in µs.

• int I4_scheduler_is_online (I4_cap_idx_t scheduler, I4_umword_t cpu) L4_NOTHROW Query if a CPU is online.

9.60.1 Detailed Description

Scheduler object. #include <14/sys/scheduler.h>

9.60.2 Enumeration Type Documentation

9.60.2.1 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.
- $\textbf{\textit{L4_SCHEDULER_IDLE_TIME_OP}} \quad \text{Query idle time for the scheduler}.$

Definition at line 185 of file scheduler.h.

9.60.3 Function Documentation

9.60.3.1 I4_sched_cpu_set_t I4_sched_cpu_set (I4_umword_t offset, unsigned char granularity, I4_umword_t map L4_DEFAULT_PARAM1) [inline]

Parameters

offset	Offset.
granularity	Granularitry in log2 notation.
тар	Bitmap of CPUs, defaults to 1 in C++.

Returns

CPU set.

Examples:

examples/sys/migrate/thread_migrate.cc.

9.60.3.2 I4_msgtag_t I4_scheduler_info (I4_cap_idx_t scheduler, I4_umword_t * cpu_max, I4_sched_cpu_set_t * cpus) [inline]

Get scheduler information.

Parameters

scheduler	Scheduler object.

Return values

9.60 Scheduler 243

		сри_тах	maximum number of CPUs ever available.
Parameters			
	cpus	cpus.offse	et is first CPU of interest. cpus.granularity (see I4_sched_cpu_set_t).
Return values			
		cpus	cpus.map Bitmap of online CPUs.

Returns

0 on success, <0 error code otherwise.

Definition at line 284 of file scheduler.h.

References I4_utcb().

Here is the call graph for this function:



9.60.3.3 I4_msgtag_t I4_scheduler_run_thread (I4_cap_idx_t scheduler, I4_cap_idx_t thread, I4_sched_param_t const * sp) [inline]

Run a thread on a Scheduler.

Parameters

scheduler	Scheduler object.
thread	Thread to run.
sp	Scheduling parameters.

Returns

0 on success, <0 error code otherwise.

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 291 of file scheduler.h.

References I4_utcb().

Here is the call graph for this function:



9.60.3.4 I4_msgtag_t I4_scheduler_idle_time (I4_cap_idx_t scheduler, I4_sched_cpu_set_t const * cpus) [inline]

Query idle time of a CPU, in µs.

Parameters

scheduler	Scheduler object.
cpus	Set of CPUs to query.

The consumed time is returned as I4_kernel_clock_t at UTCB message register 0.

Definition at line 298 of file scheduler.h.

References I4_utcb().

Here is the call graph for this function:



9.60.3.5 int I4_scheduler_is_online (I4_cap_idx_t scheduler, I4_umword_t cpu) [inline]

Query if a CPU is online.

Parameters

scheduler	Scheduler object.
сри	CPU number.

Returns

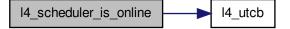
true if online, false if not (or any other query error).

Definition at line 304 of file scheduler.h.

References I4_utcb().

9.60 Scheduler 245

Here is the call graph for this function:



9.61 Task

Class definition of the Task kernel object.

Collaboration diagram for Task:



Data Structures

· class L4::Task

An L4 Task.

Enumerations

• enum I4_unmap_flags_t { L4_FP_ALL_SPACES, L4_FP_DELETE_OBJ, L4_FP_OTHER_SPACES } Flags for the unmap operation.

Functions

I4_msgtag_t I4_task_map (I4_cap_idx_t dst_task, I4_cap_idx_t src_task, I4_fpage_t const snd_fpage, I4_addr_t snd_base) L4_NOTHROW

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

 I4_msgtag_t I4_task_unmap (I4_cap_idx_t task, I4_fpage_t const fpage, I4_umword_t map_mask) L4_NOT-HROW

Revoke rights from the task.

• I4_msgtag_t I4_task_unmap_batch (I4_cap_idx_t task, I4_fpage_t const *fpages, unsigned num_fpages, unsigned long map_mask) L4_NOTHROW

Revoke rights from a task.

• I4_msgtag_t I4_task_delete_obj (I4_cap_idx_t task, I4_cap_idx_t obj) L4_NOTHROW

Release capability and delete object.

I4_msgtag_t I4_task_release_cap (I4_cap_idx_t task, I4_cap_idx_t cap) L4_NOTHROW
 Release capability.

• I4_msgtag_t I4_task_cap_valid (I4_cap_idx_t task, I4_cap_idx_t cap) L4_NOTHROW

Test whether a capability selector points to a valid capability.

• I4_msgtag_t I4_task_cap_has_child (I4_cap_idx_t task, I4_cap_idx_t cap) L4_NOTHROW

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

I4_msgtag_t I4_task_cap_equal (I4_cap_idx_t task, I4_cap_idx_t cap_a, I4_cap_idx_t cap_b) L4_NOTHRO-W

Test whether two capabilities point to the same object with the same rights.

I4_msgtag_t I4_task_add_ku_mem (I4_cap_idx_t task, I4_fpage_t const ku_mem) L4_NOTHROW
 Add kernel-user memory.

9.61 Task 247

9.61.1 Detailed Description

Class definition of the Task kernel object. #include <14/sys/task.h>

The L4 task class represents a combination of the address spaces provided by the L4 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.

A task object can be created using a Factory, see Factory (I4_factory_create_task()).

9.61.2 Enumeration Type Documentation

```
9.61.2.1 enum l4_unmap_flags_t
```

Flags for the unmap operation.

See Also

L4::Task::unmap() and I4_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() I4_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() I4_task_unmap()

L4_FP_OTHER_SPACES Counterpart to L4_FP_ALL_SPACES, unmap only child mappings.

See Also

L4::Task::unmap() I4_task_unmap()

Definition at line 163 of file consts.h.

9.61.3 Function Documentation

```
9.61.3.1 I4_msgtag_t I4_task_map ( I4_cap_idx_t dst_task, I4_cap_idx_t src_task, I4_fpage_t const snd_fpage, I4_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 359 of file task.h.

References I4 utcb().

Referenced by L4::Cap< L4Re::L4Re::Dataspace >::move().

Here is the call graph for this function:



Here is the caller graph for this function:



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 I4_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 366 of file task.h.

References I4_utcb().

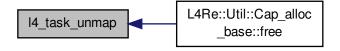
Referenced by L4Re::Util::Cap_alloc_base::free().

9.61 Task 249

Here is the call graph for this function:



Here is the caller graph for this function:



9.61.3.3 I4_msgtag_t I4_task_unmap_batch (I4_cap_idx_t task, I4_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 I4_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 373 of file task.h.

References I4_utcb().

Here is the call graph for this function:



9.61.3.4 I4_msgtag_t I4_task_delete_obj(I4_cap_idx_t task, I4_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

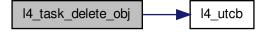
The object will be deleted if the obj has sufficient rights. No error will be reported if the rights are insufficient, however, the capability is removed in all cases.

This is operating calls I4_task_unmap() with L4_FP_DELETE_OBJ.

Definition at line 389 of file task.h.

References I4_utcb().

Here is the call graph for this function:



9.61.3.5 I4_msgtag_t I4_task_release_cap(I4_cap_idx_t task, I4_cap_idx_t cap) [inline]

Release capability.

Parameters

9.61 Task 251

task	Capability selector of destination task
сар	Capability selector to release

Returns

Syscall return tag

This operation unmaps the capability from the specified task.

Definition at line 404 of file task.h.

References I4_utcb().

Here is the call graph for this function:



9.61.3.6 I4_msgtag_t I4_task_cap_valid (I4_cap_idx_t task, I4_cap_idx_t cap) [inline]

Test whether a capability selector points to a valid capability.

Parameters

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

Returns

label contains >0 if valid, 0 if invalid

Definition at line 410 of file task.h.

References I4_utcb().

Here is the call graph for this function:



9.61.3.7 I4_msgtag_t I4_task_cap_has_child (I4_cap_idx_t task, I4_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
сар	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

Definition at line 416 of file task.h.

References I4_utcb().

Here is the call graph for this function:



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 422 of file task.h.

References I4_utcb().

Here is the call graph for this function:



9.61 Task 253

9.61.3.9 I4_msgtag_t I4_task_add_ku_mem (I4_cap_idx_t task, I4_fpage_t const 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 429 of file task.h.

References I4_utcb().

Here is the call graph for this function:

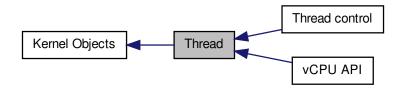


9.62 Thread 255

9.62 Thread

Thread object.

Collaboration diagram for Thread:



Modules

· Thread control

API for Thread Control method.

vCPU API

vCPU API

Data Structures

class L4::Thread

L4 kernel thread.

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_THREAD_MODIFY_SENDER_OP = 6UL, L4_THREAD_VCPU_CONTROL_OP = 7UL,
    L4_THREAD_X86_GDT_OP = 0x10UL, L4_THREAD_ARM_TPIDRURO_OP = 0x10UL, L4_THREAD_AMD64_SET_SEGMENT_BASE_OP = 0x12UL, L4_THREAD_OPCODE_MASK = 0xffff }
```

Operations on thread objects.

enum L4_thread_control_flags {
 L4_THREAD_CONTROL_SET_PAGER = 0x0010000, L4_THREAD_CONTROL_BIND_TASK = 0x0200000,
 L4_THREAD_CONTROL_ALIEN = 0x0400000, L4_THREAD_CONTROL_UX_NATIVE = 0x0800000,
 L4_THREAD_CONTROL_SET_EXC_HANDLER = 0x1000000 }

Flags for the thread control operation.

enum L4_thread_control_mr_indices {
 L4_THREAD_CONTROL_MR_IDX_FLAGS = 0, L4_THREAD_CONTROL_MR_IDX_PAGER = 1, L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER = 2, L4_THREAD_CONTROL_MR_IDX_FLAG_VALS = 4,
 L4_THREAD_CONTROL_MR_IDX_BIND_UTCB = 5, L4_THREAD_CONTROL_MR_IDX_BIND_TASK = 6
 }

Indices for the values in the message register for thread control.

enum L4_thread_ex_regs_flags { L4_THREAD_EX_REGS_CANCEL = 0x10000UL, L4_THREAD_EX_REGS_TRIGGER_EXCEPTION = 0x20000UL }

Flags for the thread ex-regs operation.

Functions

 I4_msgtag_t I4_thread_ex_regs (I4_cap_idx_t thread, I4_addr_t ip, I4_addr_t sp, I4_umword_t flags) L4_NO-THROW

Exchange basic thread registers.

I4_msgtag_t I4_thread_ex_regs_ret (I4_cap_idx_t thread, I4_addr_t *ip, I4_addr_t *sp, I4_umword_t *flags)
 L4_NOTHROW

Exchange basic thread registers and return previous values.

• I4_msgtag_t I4_thread_yield (void) L4_NOTHROW

Yield current time slice.

• 14 msgtag t 14 thread switch (14 cap idx t to thread) L4 NOTHROW

Switch to another thread (and donate the remaining time slice).

I4_msgtag_t I4_thread_stats_time (I4_cap_idx_t thread) L4_NOTHROW

Get consumed timed of thread in us.

• I4_msgtag_t I4_thread_vcpu_resume_start (void) L4_NOTHROW

vCPU return from event handler.

- I4_msgtag_t I4_thread_vcpu_resume_commit (I4_cap_idx_t thread, I4_msgtag_t tag) L4_NOTHROW Commit vCPU resume.
- I4_msgtag_t I4_thread_vcpu_control (I4_cap_idx_t thread, I4_addr_t vcpu_state) L4_NOTHROW Enable or disable the vCPU feature for the thread.
- I4_msgtag_t I4_thread_vcpu_control_ext (I4_cap_idx_t thread, I4_addr_t ext_vcpu_state) L4_NOTHROW Enable or disable the extended vCPU feature for the thread.
- I4_msgtag_t I4_thread_register_del_irq (I4_cap_idx_t thread, I4_cap_idx_t irq) L4_NOTHROW Register an IRQ that will trigger upon deletion events.
- I4_msgtag_t I4_thread_modify_sender_start (void) L4_NOTHROW

Start a thread sender modifiction sequence.

 int I4_thread_modify_sender_add (I4_umword_t match_mask, I4_umword_t match, I4_umword_t del_bits, I4umword_t add_bits, I4_msgtag_t *tag) L4_NOTHROW

Add a modification pattern to a sender modification sequence.

- I4_msgtag_t I4_thread_modify_sender_commit (I4_cap_idx_t thread, I4_msgtag_t tag) L4_NOTHROW
 Apply (commit) a sender modification sequence.
- I4_msgtag_t I4_thread_arm_set_tpidruro (I4_cap_idx_t thread, I4_addr_t tpidruro) L4_NOTHROW Set the TPIDRURO thread specific register.

9.62.1 Detailed Description

Thread object. #include <14/sys/thread.h>

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 Factory (I4_factory_create_thread()).

An L4 thread encapsulates:

- · CPU state
 - General-purpose registers
 - Program counter
 - Stack pointer
- FPU state
- · Scheduling parameters
 - CPU-set
 - Priority (0-255)

9.62 Thread 257

- Time slice length
- · 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 I4_thread_control_start() for more information.

9.62.2 Enumeration Type Documentation

9.62.2.1 enum L4 thread ops

Operations on thread objects.

Enumerator

- L4_THREAD_CONTROL_OP Control operation.
- **L4_THREAD_EX_REGS_OP** Exchange registers operation.
- **L4_THREAD_SWITCH_OP** Do a thread switch.
- L4_THREAD_STATS_OP Thread statistics.
- **L4_THREAD_VCPU_RESUME_OP** VCPU resume.
- L4_THREAD_REGISTER_DELETE_IRQ_OP Register an IPC-gate deletion IRQ.
- L4_THREAD_MODIFY_SENDER_OP Modify all senders IDs that match the given pattern.
- L4_THREAD_X86_GDT_OP Gdt.
- L4_THREAD_ARM_TPIDRURO_OP Set TPIDRURO register.
- L4_THREAD_AMD64_SET_SEGMENT_BASE_OP Set segment base.
- L4_THREAD_OPCODE_MASK Mask for opcodes.

Definition at line 591 of file thread.h.

9.62.2.2 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 618 of file thread.h.

9.62.2.3 enum L4_thread_control_mr_indices

Indices for the values in the message register for thread control.

Enumerator

See Also

- L4_THREAD_CONTROL_MR_IDX_FLAGS 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 641 of file thread.h.

```
9.62.2.4 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 656 of file thread.h.

9.62.3 Function Documentation

```
9.62.3.1 I4_msgtag_t I4_thread_ex_regs ( I4_cap_idx_t thread, I4_addr_t ip, I4_addr_t sp, I4_umword_t flags )
[inline]
```

Exchange basic thread registers.

Parameters

thread	Thread to manipulate
ip	New instruction pointer, use \sim 0UL to leave the instruction pointer unchanged
sp	New stack pointer, use $\sim\!\!$ 0UL to leave the stack pointer unchanged
flags	Ex-regs flags, see L4_thread_ex_regs_flags

Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see *flags*).

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 796 of file thread.h.

References I4_utcb().

9.62 Thread 259

Here is the call graph for this function:



9.62.3.2 I4_msgtag_t I4_thread_ex_regs_ret (I4_cap_idx_t thread, I4_addr_t * ip, I4_addr_t * sp, I4_umword_t * flags) [inline]

Exchange basic thread registers and return previous values.

Parameters

in	thread	Thread to manipulate
in,out	ip	New instruction pointer, use \sim 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 803 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.62.3.3 I4_msgtag_t I4_thread_yield (void) [inline]

Yield current time slice.

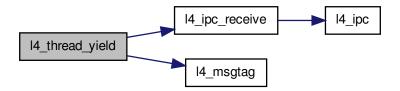
Returns

system call return tag

Definition at line 756 of file thread.h.

References L4_INVALID_CAP, L4_IPC_BOTH_TIMEOUT_0, I4_ipc_receive(), and I4_msgtag().

Here is the call graph for this function:



9.62.3.4 I4_msgtag_t I4_thread_switch(I4_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 856 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.62.3.5 I4_msgtag_t I4_thread_stats_time (I4_cap_idx_t thread) [inline]

Get consumed timed of thread in μ s.

9.62 Thread 261

Parameters

thread Thread to get the consumed time from.

The consumed time is returned as I4_kernel_clock_t at UTCB message register 0.

Definition at line 865 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.62.3.6 I4_msgtag_t I4_thread_vcpu_resume_start(void) [inline]

vCPU return from event handler.

Returns

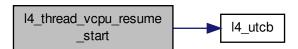
Message tag to be used for I4_sndfpage_add() and I4_thread_vcpu_commit()

The vCPU resume functionality is split in multiple functions to allow the specification of additional send-flex-pages using I4_sndfpage_add().

Definition at line 871 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.62.3.7 I4_msgtag_t I4_thread_vcpu_resume_commit(I4_cap_idx_t thread, I4_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 I4_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 are 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. 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.

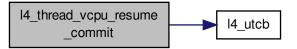
See Also

I4_vcpu_state_t

Definition at line 877 of file thread.h.

References I4 utcb().

Here is the call graph for this function:



9.62.3.8 I4_msgtag_t I4_thread_vcpu_control (I4_cap_idx_t thread, I4_addr_t vcpu_state) [inline]

Enable or disable the vCPU feature for the thread.

Parameters

thread	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.

Returns

Systemcall result message 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.

Definition at line 914 of file thread.h.

9.62 Thread 263

References I4_utcb().

Here is the call graph for this function:



9.62.3.9 I4_msgtag_t I4_thread_vcpu_control_ext(I4_cap_idx_t thread, I4_addr_t ext_vcpu_state) [inline]

Enable or disable the extended vCPU feature for the thread.

Parameters

thread	The thread for which the extended 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.

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 *vcpu_state* is set to a valid kernel-user-memory address, or disables the vCPU feature if *vcpu_state* is 0.

Definition at line 929 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.62.3.10 I4_msgtag_t I4_thread_register_del_irq(I4_cap_idx_t thread, I4_cap_idx_t irq) [inline]

Register an IRQ that will trigger upon deletion events.

Parameters

thread	Thread to register IRQ for.
irq	Irq to register.

Returns

System call result message tag.

Definition at line 897 of file thread.h.

References I4 utcb().

Here is the call graph for this function:



9.62.3.11 I4_msgtag_t I4_thread_modify_sender_start (void) [inline]

Start a thread sender modifiction sequence.

Add modification rules with I4_thread_modify_sender_add() and commit with I4_thread_modify_sender_commit(). Do not touch the UTCB between I4_thread_modify_sender_start() and I4_thread_modify_sender_commit().

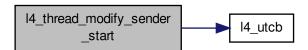
See Also

I4_thread_modify_sender_add
I4_thread_modify_sender_commit

Definition at line 970 of file thread.h.

References I4 utcb().

Here is the call graph for this function:



9.62.3.12 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.

9.62 Thread 265

Parameters

tag	Tag received from I4_thread_modify_sender_start() or previous I4_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 & \sim del_bits) | add_bits; } Only the first match is applied.

See Also

I4_thread_modify_sender_startI4_thread_modify_sender_commit

Definition at line 976 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.62.3.13 I4_msgtag_t I4_thread_modify_sender_commit (I4_cap_idx_t thread, I4_msgtag_t tag) [inline]

Apply (commit) a sender modification sequence.

See Also

I4_thread_modify_sender_startI4_thread_modify_sender_add

Definition at line 987 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.62.3.14 I4_msgtag_t I4_thread_arm_set_tpidruro (I4_cap_idx_t thread, I4_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.

References I4_utcb().

Here is the call graph for this function:



9.63 Thread control 267

9.63 Thread control

API for Thread Control method.

Collaboration diagram for Thread control:



Functions

• void I4_thread_control_start (void) L4_NOTHROW

Start a thread control API sequence.

void I4_thread_control_pager (I4_cap_idx_t pager) L4_NOTHROW

Set the pager.

void I4_thread_control_exc_handler (I4_cap_idx_t exc_handler) L4_NOTHROW

Set the exception handler.

void I4_thread_control_bind (I4_utcb_t *thread_utcb, I4_cap_idx_t task) L4_NOTHROW

Bind the thread to a task.

void I4_thread_control_alien (int on) L4_NOTHROW

Enable alien mode.

• void I4_thread_control_ux_host_syscall (int on) L4_NOTHROW

Enable pass through of native host (Linux) system calls.

• I4_msgtag_t I4_thread_control_commit (I4_cap_idx_t thread) L4_NOTHROW

Commit the thread control parameters.

9.63.1 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 I4_thread_control_commit()).

A thread control operation must always start with I4_thread_control_start() and be committed with I4_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.

```
I4_utcb_t *u = I4_utcb();
I4_thread_control_start(u);
I4_thread_control_pager(u, pager_cap);
I4_thread_control_bind (u, thread_utcb, task);
I4_thread_control_commit(u, thread_cap);
```

9.63.2 Function Documentation

9.63.2.1 void I4_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.

- 14 thread control pager()
- 14 thread control exc handler()
- 14 thread control bind()
- I4_thread_control_alien()
- I4_thread_control_ux_host_syscall() (Fiasco-UX only)

To commit the changes to the thread I4_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 I4_thread_control_start() and I4_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 810 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.63.2.2 void I4_thread_control_pager (I4_cap_idx_t pager) [inline]

Set the pager.

Parameters

pager	Capability selector invoked to send a page-fault IPC.

9.63 Thread control 269

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 816 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.63.2.3 void I4_thread_control_exc_handler (I4_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).

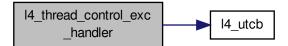
Examples:

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 822 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.63.2.4 void I4_thread_control_bind (I4_utcb_t * thread_utcb, I4_cap_idx_t task) [inline]

Bind the thread to a task.

9.63 Thread control 271

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 829 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.63.2.5 void I4_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 835 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.63.2.6 void I4_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 841 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.63.2.7 I4_msgtag_t I4_thread_control_commit (I4_cap_idx_t thread) [inline]

Commit the thread control parameters.

Parameters

thread	Capability selector of target thread to commit to.

9.63 Thread control 273

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 847 of file thread.h.

References I4_utcb().

Here is the call graph for this function:



9.64 Message Tag

API related to the message tag data type.

Collaboration diagram for Message Tag:



Data Structures

struct I4_msgtag_t

Message tag data structure.

Typedefs

typedef struct I4_msgtag_t I4_msgtag_t
 Message tag data structure.

Enumerations

```
    enum I4_msgtag_protocol {
        L4_PROTO_NONE = 0, L4_PROTO_ALLOW_SYSCALL = 1, L4_PROTO_PF_EXCEPTION = 1, L4_PROTO_O_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_TO_TASK = -11L,
        L4_PROTO_THREAD = -12L, L4_PROTO_LOG = -13L, L4_PROTO_SCHEDULER = -14L, L4_PROTO_F-ACTORY = -15L,
        L4_PROTO_VM = -16L, L4_PROTO_META = -21L }
        Message tag for IPC operations.
    enum I4_msgtag_flags {
        L4_MSGTAG_ERROR, L4_MSGTAG_XCPU, L4_MSGTAG_TRANSFER_FPU, L4_MSGTAG_SCHEDULE,
        L4_MSGTAG_PROPAGATE, L4_MSGTAG_FLAGS }
        Flags for message tags.
```

Functions

- I4_msgtag_t I4_msgtag (long label, unsigned words, unsigned items, unsigned flags) L4_NOTHROW
 Create a message tag from the specified values.
- long I4_msgtag_label (I4_msgtag_t t) L4_NOTHROW

Get the protocol of tag.

unsigned I4 msgtag words (I4 msgtag t t) L4 NOTHROW

Get the number of untyped words.

• unsigned I4_msgtag_items (I4_msgtag_t t) L4_NOTHROW

9.64 Message Tag 275

Get the number of typed items.

unsigned I4_msgtag_flags (I4_msgtag_t t) L4_NOTHROW
 Get the flags.

unsigned I4_msgtag_has_error (I4_msgtag_t t) L4_NOTHROW

Test for error indicator flag.

unsigned I4_msgtag_is_page_fault (I4_msgtag_t t) L4_NOTHROW
 Test for page-fault protocol.

• unsigned I4_msgtag_is_preemption (I4_msgtag_t t) L4_NOTHROW

Test for preemption protocol.

unsigned I4 msgtag is sys exception (I4 msgtag t t) L4 NOTHROW

Test for system-exception protocol.

unsigned I4_msgtag_is_exception (I4_msgtag_t t) L4_NOTHROW
 Test for exception protocol.

unsigned I4_msgtag_is_sigma0 (I4_msgtag_t t) L4_NOTHROW
 Test for sigma0 protocol.

unsigned I4_msgtag_is_io_page_fault (I4_msgtag_t t) L4_NOTHROW
 Test for IO-page-fault protocol.

9.64.1 Detailed Description

API related to the message tag data type. #include <14/sys/types.h>

9.64.2 Typedef Documentation

9.64.2.1 typedef struct I4_msgtag_t I4_msgtag_t

Message tag data structure.

```
#include <14/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.

9.64.3 Enumeration Type Documentation

9.64.3.1 enum I4_msgtag_protocol

Message tag for IPC operations.

All predefined protocols used by the kernel.

Enumerator

L4_PROTO_NONE Default protocol tag to reply to kernel.

L4_PROTO_ALLOW_SYSCALL Allow an alien the system call.

L4_PROTO_PF_EXCEPTION Make an exception out of a page fault.

L4_PROTO_IRQ IRQ message.

L4_PROTO_PAGE_FAULT Page fault message.

L4_PROTO_PREEMPTION Preemption message.

- L4_PROTO_SYS_EXCEPTION System exception.
- L4_PROTO_EXCEPTION Exception.
- L4_PROTO_SIGMA0 Sigma0 protocol.
- L4_PROTO_IO_PAGE_FAULT I/O page fault message.
- L4_PROTO_KOBJECT Protocol for messages to a 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_META** Meta information protocol.

Definition at line 49 of file types.h.

9.64.3.2 enum l4 msgtag flags

Flags for message tags.

Enumerator

- L4_MSGTAG_ERROR Error indicator flag.
- L4_MSGTAG_XCPU Cross-CPU invocation 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 89 of file types.h.

9.64.4 Function Documentation

9.64.4.1 I4_msgtag_t I4_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

9.64 Message Tag 277

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

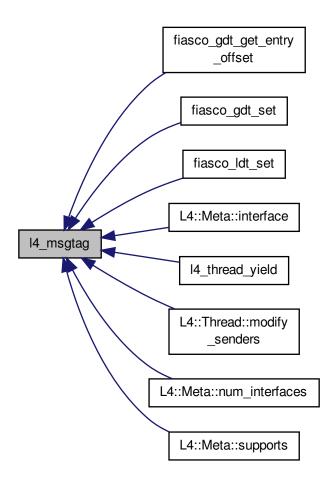
Examples:

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 366 of file types.h.

Referenced by fiasco_gdt_get_entry_offset(), fiasco_gdt_set(), fiasco_ldt_set(), L4::Meta::interface(), l4_thread_yield(), L4::Thread::modify_senders(), L4::Meta::num_interfaces(), and L4::Meta::supports().

Here is the caller graph for this function:



9.64.4.2 long l4_msgtag_label(l4_msgtag_t t) [inline]

Get the protocol of tag.

Parameters

, T	
$t \mid Ine tag$	
i inotag	

Returns

Label

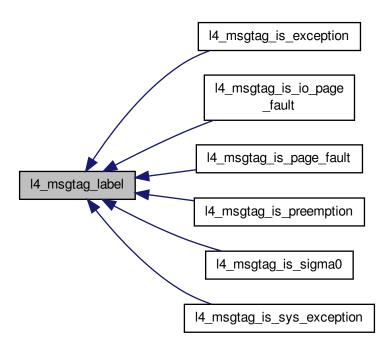
Examples:

examples/sys/singlestep/main.c, and examples/sys/start-with-exc/main.c.

Definition at line 377 of file types.h.

Referenced by I4_msgtag_is_exception(), I4_msgtag_is_io_page_fault(), I4_msgtag_is_page_fault(), I4_ms

Here is the caller graph for this function:



9.64.4.3 unsigned I4_msgtag_words (I4_msgtag_t t) [inline]

Get the number of untyped words.

Parameters

t	The tag

9.64 Message Tag 279

Returns

Number of words

Examples:

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

Definition at line 381 of file types.h.

9.64.4.4 unsigned I4_msgtag_items (I4_msgtag_t t) [inline]

Get the number of typed items.

Parameters

```
t The tag
```

Returns

Number of items.

Definition at line 385 of file types.h.

```
9.64.4.5 unsigned I4_msgtag_flags ( I4_msgtag_t t ) [inline]
```

Get the flags.

The flag are defined by I4_msgtag_flags.

Parameters

```
t \mid the tag
```

Returns

Flags

Definition at line 389 of file types.h.

```
9.64.4.6 unsigned I4_msgtag_has_error(I4_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:

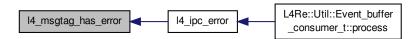
examples/sys/aliens/main.c, examples/sys/singlestep/main.c, and examples/sys/utcb-ipc/main.c.

Definition at line 394 of file types.h.

References L4_MSGTAG_ERROR.

Referenced by I4_ipc_error().

Here is the caller graph for this function:



9.64.4.7 unsigned I4_msgtag_is_page_fault (I4_msgtag_t t) [inline]

Test for page-fault protocol.

Parameters

```
t the tag
```

Returns

Boolean value

Definition at line 399 of file types.h.

References I4_msgtag_label(), and L4_PROTO_PAGE_FAULT.

Here is the call graph for this function:



9.64.4.8 unsigned I4_msgtag_is_preemption (I4_msgtag_t t) [inline]

Test for preemption protocol.

Parameters

t	the tag

9.64 Message Tag 281

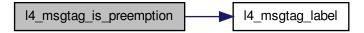
Returns

Boolean value

Definition at line 402 of file types.h.

References I4_msgtag_label(), and L4_PROTO_PREEMPTION.

Here is the call graph for this function:



9.64.4.9 unsigned I4_msgtag_is_sys_exception (I4_msgtag_t t) [inline]

Test for system-exception protocol.

Parameters

```
t the tag
```

Returns

Boolean value

Definition at line 405 of file types.h.

References I4_msgtag_label(), and L4_PROTO_SYS_EXCEPTION.

Here is the call graph for this function:



9.64.4.10 unsigned I4_msgtag_is_exception (I4_msgtag_t t) [inline]

Test for exception protocol.

Parameters

Generated for L4Re by Doxygen

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 408 of file types.h.

References I4_msgtag_label(), and L4_PROTO_EXCEPTION.

Here is the call graph for this function:



9.64.4.11 unsigned I4_msgtag_is_sigma0 (I4_msgtag_t t) [inline]

Test for sigma0 protocol.

Parameters

```
t \mid the tag
```

Returns

Boolean value

Definition at line 411 of file types.h.

References I4_msgtag_label(), and L4_PROTO_SIGMA0.

Here is the call graph for this function:



9.64.4.12 unsigned I4_msgtag_is_io_page_fault(I4_msgtag_t t) [inline]

Test for IO-page-fault protocol.

9.64 Message Tag 283

Parameters

t	the tag

Returns

Boolean value

Definition at line 414 of file types.h.

References I4_msgtag_label(), and L4_PROTO_IO_PAGE_FAULT.

Here is the call graph for this function:



9.65 Capabilities

Functions and definitions related to capabilities.

Collaboration diagram for Capabilities:



Data Structures

· class L4::Cap_base

Base class for all kinds of capabilities.

class L4::Cap< T >

Capability Selector a la C++.

Macros

• #define L4_DISABLE_COPY(_class)

Disable copy of a class.

• #define L4_KOBJECT_DISABLE_COPY(_class)

Disable copy and instantiation of a class.

• #define L4_KOBJECT(_class) L4_KOBJECT_DISABLE_COPY(_class)

Declare a kernel object class.

Typedefs

• typedef unsigned long I4_cap_idx_t

L4 Capability selector Type.

Enumerations

• enum I4_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 }
```

Default capabilities setup for the initial tasks.

Functions

```
    template < typename T, typename F >
        Cap < T > L4::cap_cast (Cap < F > const &c) throw ()
        static_cast for capabilities.
```

9.65 Capabilities 285

```
    template<typename T, typename F >
        Cap< T > L4::cap_reinterpret_cast (Cap< F > const &c) throw ()
        reinterpret_cast for capabilities.
    template<typename T, typename F >
        Cap< T > L4::cap_dynamic_cast (Cap< F > const &c) throw ()
        dynamic_cast for capabilities.
    unsigned I4_is_invalid_cap (I4_cap_idx_t c) L4_NOTHROW
        Test if a capability selector is the invalid capability.
    unsigned I4_is_valid_cap (I4_cap_idx_t c) L4_NOTHROW
        Test if a capability selector is a valid selector.
    unsigned I4_capability_equal (I4_cap_idx_t c1, I4_cap_idx_t c2) L4_NOTHROW
        Test if two capability_selectors are equal.
```

9.65.1 Detailed Description

Functions and definitions related to capabilities. #include <14/sys/consts.h>

C++ interface for capabilities:

```
#include <14/sys/capability>
```

C interface for capabilities:

```
#include <14/sys/types.h>
```

9.65.2 Macro Definition Documentation

```
9.65.2.1 #define L4_DISABLE_COPY( _class )
```

Value:

Disable copy of a class.

Parameters

```
_class | is the 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 397 of file capability.

```
9.65.2.2 #define L4_KOBJECT_DISABLE_COPY( _class )
```

Value:

Disable copy and instantiation of a class.

Parameters

_class is the name of the class to be not copyable and not instantiatable.

The typical use looks like:

```
* class Type
* {
* L4_KOBJECT_DISABLE_COPY(Type)
* };
```

Definition at line 417 of file capability.

```
9.65.2.3 #define L4_KOBJECT( _class ) L4_KOBJECT_DISABLE_COPY(_class)
```

Declare a kernel object class.

Parameters

```
_class is the class name.
```

The use of this macro disables copy and instantiation of the class as needed for kernel object classes derived from L4::Kobject.

The typical use looks like:

```
* class Type : public L4::Kobject_t<Type, L4::Kobject>
* {
* L4_KOBJECT(Type)
* };
```

Definition at line 440 of file capability.

9.65.3 Typedef Documentation

```
9.65.3.1 typedef unsigned long I4_cap_idx_t
```

L4 Capability selector Type.

```
#include <14/sys/types.h>
```

Definition at line 319 of file types.h.

9.65.4 Enumeration Type Documentation

```
9.65.4.1 enum I4_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 I4_cap_idx_t.

L4_INVALID_CAP Invalid capability selector.

Definition at line 134 of file consts.h.

9.65 Capabilities 287

```
9.65.4.2 enum I4_default_caps_t
```

Default capabilities setup for the initial tasks.

```
#include <14/sys/consts.h>
```

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.

Definition at line 248 of file consts.h.

9.65.5 Function Documentation

```
9.65.5.1 template < typename F > Cap < T > L4::cap cast ( Cap < <math>F > const & c ) throw) [inline]
```

static_cast for capabilities.

Parameters

Т	is the target type of the capability
F	is the source type (and is usually implicitly set)
С	is 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 adjustment 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 343 of file capability.

Referenced by L4Re::Cap_alloc::alloc().

Here is the caller graph for this function:



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

reinterpret_cast for capabilities.

Parameters

T	is the target type of the capability
F	is the source type (and is usually implicitly set)
С	is 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:

Definition at line 367 of file capability.

Referenced by L4::cap_dynamic_cast().

Here is the caller graph for this function:



```
9.65.5.3 template < typename T , typename F > Cap < T > L4::cap_dynamic_cast ( Cap < F > const & c ) throw )  \lceil \texttt{inline} \rceil
```

dynamic_cast for capabilities.

9.65 Capabilities 289

Parameters

T	is the target type of the capability
F	is the source type (and is usually implicitly set)
С	is the source capability that shall be casted

Returns

A capability typed to the interface T. If the object does not support the target interface T or does not support the L4::Meta interface the result is the invalid capability selector.

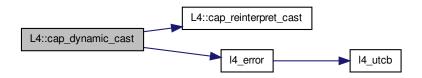
The use of this cast operator is similar to the dynamic_cast<>() for C++ pointers. It also induces overhead, because it uses the meta interface (L4::Meta) to do runtime type checking.

Example code:

Definition at line 550 of file capability.

References L4::cap_reinterpret_cast(), and I4_error().

Here is the call graph for this function:



```
9.65.5.4 unsigned I4_is_invalid_cap ( I4_cap_idx_t c ) [inline]
```

Test if a capability selector is the invalid capability.

Parameters

С	Capability selector

Returns

Boolean value

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 350 of file types.h.

```
9.65.5.5 unsigned I4_is_valid_cap(I4_cap_idx_tc) [inline]
```

Test if a capability selector is a valid selector.

Parameters

С	Capability selector

Returns

Boolean value

Definition at line 354 of file types.h.

```
9.65.5.6 unsigned I4_capability_equal ( I4_cap_idx_t c1, I4_cap_idx_t c2 ) [inline]
```

Test if two capability selectors are equal.

Parameters

c1	Capability
c2	Capability

Returns

1 if equal, 0 if not equal

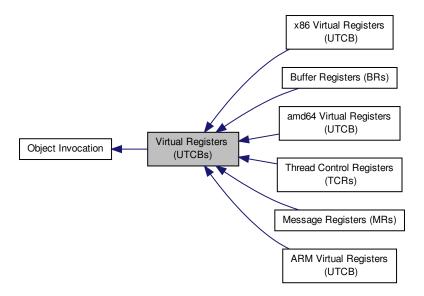
Definition at line 358 of file types.h.

References L4_CAP_SHIFT.

9.66 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 I4_utcb_t I4_utcb_t

Opaque type for the UTCB.

Functions

I4_utcb_t * I4_utcb (void) L4_NOTHROW L4_PURE

Get the UTCB address.

I4_msg_regs_t * I4_utcb_mr (void) L4_NOTHROW L4_PURE

Get the message-register block of a UTCB.

• I4_buf_regs_t * I4_utcb_br (void) L4_NOTHROW L4_PURE

Get the buffer-register block of a UTCB.

I4_thread_regs_t * I4_utcb_tcr (void) L4_NOTHROW L4_PURE

Get the thread-control-register block of a UTCB.

9.66.1 Detailed Description

L4 Virtual Registers (UTCB). Includes:

```
#include <14/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, I4_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 I4_utcb_mr(), I4_utcb_tcr(), and I4_utcb_br() functions must be used.

9.66.2 Typedef Documentation

```
9.66.2.1 typedef struct I4_utcb_t I4_utcb_t
```

Opaque type for the UTCB.

To access the contents of the virtual registers the I4_utcb_mr(), I4_utcb_tcr(), and I4_utcb_br() functions must be used.

Definition at line 68 of file utcb.h.

9.66.3 Function Documentation

```
9.66.3.1 I4_msg_regs_t * I4_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 342 of file utcb.h.

References I4_utcb().

Referenced by L4Re::Util::Vcon_svr< SVR >::dispatch().

Here is the call graph for this function:



Here is the caller graph for this function:



Get the buffer-register block of a UTCB.

Returns

A pointer to the buffer-register block of u.

Definition at line 345 of file utcb.h.

References I4_utcb().

Here is the call graph for this function:



9.66.3.3 I4_thread_regs_t * I4_utcb_tcr(void) [inline]

Get the thread-control-register block of a UTCB.

Returns

A pointer to the thread-control-register block of \boldsymbol{u} .

Definition at line 348 of file utcb.h.

References I4_utcb().

Here is the call graph for this function:



9.67 Message Registers (MRs)

Collaboration diagram for Message Registers (MRs):



Modules

· Exception registers

Overly definition of the MRs for exception messages.

Data Structures

• union I4_msg_regs_t

Encapsulation of the message-register block in the UTCB.

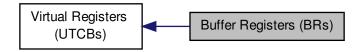
Typedefs

typedef union I4_msg_regs_t I4_msg_regs_t
 Encapsulation of the message-register block in the UTCB.

9.67.1 Detailed Description

9.68 Buffer Registers (BRs)

Collaboration diagram for Buffer Registers (BRs):



Data Structures

• struct I4_buf_regs_t

Encapsulation of the buffer-registers block in the UTCB.

Typedefs

typedef struct I4_buf_regs_t I4_buf_regs_t

Encapsulation of the buffer-registers block in the UTCB.

Enumerations

enum I4_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 I4_utcb_inherit_fpu (int switch_on) L4_NOTHROW

Enable or disable inheritance of FPU state to receiver.

- 9.68.1 Detailed Description
- 9.68.2 Enumeration Type Documentation
- 9.68.2.1 enum I4_buffer_desc_consts_t

Constants for buffer descriptors.

Enumerator

L4_BDR_MEM_SHIFT Bit offset for the memory-buffer index.

L4_BDR_IO_SHIFT Bit offset for the IO-buffer index.

L4_BDR_OBJ_SHIFT Bit offset for the capability-buffer index.

Definition at line 225 of file consts.h.

9.69 Thread Control Registers (TCRs)

Collaboration diagram for Thread Control Registers (TCRs):



Data Structures

• struct I4_thread_regs_t

Encapsulation of the thread-control-register block of the UTCB.

Typedefs

typedef struct I4_thread_regs_t I4_thread_regs_t
 Encapsulation of the thread-control-register block of the UTCB.

9.69.1 Detailed Description

9.70 Exception registers

Overly definition of the MRs for exception messages.

Collaboration diagram for Exception registers:



Functions

- I4_exc_regs_t * I4_utcb_exc (void) L4_NOTHROW L4_PURE
 Get the message-register block of a UTCB (for an exception IPC).
- I4_umword_t I4_utcb_exc_pc (I4_exc_regs_t *u) L4_NOTHROW L4_PURE
 Access function to get the program counter of the exception state.
- void I4_utcb_exc_pc_set (I4_exc_regs_t *u, I4_addr_t pc) L4_NOTHROW
- Set the program counter register in the exception state.
- unsigned long I4_utcb_exc_typeval (I4_exc_regs_t *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 *u) L4_NOTHROW L4_PURE

 Check whether an exception IPC is a page fault.
- I4_addr_t I4_utcb_exc_pfa (I4_exc_regs_t *u) L4_NOTHROW L4_PURE Function to get the L4 style page fault address out of an exception.

9.70.1 Detailed Description

Overly definition of the MRs for exception messages.

9.70.2 Function Documentation

```
\textbf{9.70.2.1} \quad \textbf{I4\_exc\_regs\_t} * \textbf{I4\_utcb\_exc(void)} \quad \texttt{[inline]}
```

Get the message-register block of a UTCB (for an exception IPC).

Returns

A pointer to the exception message in u.

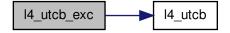
Examples:

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

Definition at line 351 of file utcb.h.

References I4_utcb().

Here is the call graph for this function:



Access function to get the program counter of the exception state.

Parameters

и	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.

9.70.2.3 void
$$14_utcb_exc_pc_set(14_exc_regs_t*u, 14_addr_t pc)$$
 [inline]

Set the program counter register in the exception state.

Parameters

и	UTCB	
рс	The program counter to set.	

Definition at line 95 of file utcb.h.

Check whether an exception IPC is a page fault.

Returns

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.

9.71 Virtual Console

Virtual console for simple character based input and output.

Collaboration diagram for Virtual Console:



Data Structures

class L4::Vcon

C++ L4 Vcon.

struct I4_vcon_attr_t

Vcon attribute structure.

Typedefs

typedef struct I4_vcon_attr_t I4_vcon_attr_t

Vcon attribute structure.

Enumerations

```
    enum L4_vcon_write_consts { L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(I4_umword_t) }
```

Constants for I4_vcon_write.

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 | flags { L4 VCON | ICANON = 000002, L4 VCON | ECHO = 000010 }

Local flags.

 enum L4_vcon_ops { L4_VCON_WRITE_OP = 0UL, L4_VCON_SET_ATTR_OP = 2UL, L4_VCON_GET_A-TTR_OP = 3UL }

Operations on the vcon objects.

Functions

- I4_msgtag_t I4_vcon_send (I4_cap_idx_t vcon, char const *buf, int size) L4_NOTHROW
 Send data to virtual console.
- long I4_vcon_write (I4_cap_idx_t vcon, char const *buf, int size) L4_NOTHROW
 Write data to virtual console.
- int l4_vcon_read (l4_cap_idx_t vcon, char *buf, int size) L4_NOTHROW

9.71 Virtual Console 301

Read data from virtual console.

I4_msgtag_t I4_vcon_set_attr (I4_cap_idx_t vcon, I4_vcon_attr_t const *attr) L4_NOTHROW
 Set attributes of a Vcon.

I4_msgtag_t I4_vcon_get_attr (I4_cap_idx_t vcon, I4_vcon_attr_t *attr) L4_NOTHROW
 Get attributes of a Vcon.

9.71.1 Detailed Description

Virtual console for simple character based input and output. #include < 14/sys/vcon.h >Interrupt for read events are provided by the virtual key interrupt.

9.71.2 Enumeration Type Documentation

```
9.71.2.1 enum L4_vcon_write_consts
```

Constants for I4_vcon_write.

Enumerator

L4_VCON_WRITE_SIZE Maximum size that can be written with one I4_vcon_write call.

Definition at line 83 of file vcon.h.

```
9.71.2.2 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 126 of file vcon.h.

```
9.71.2.3 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 ouput CR.

Definition at line 137 of file vcon.h.

```
9.71.2.4 enum L4_vcon_I_flags
```

Local flags.

Enumerator

L4_VCON_ICANON Cannonical mode.

L4_VCON_ECHO Echo input.

Definition at line 148 of file vcon.h.

9.71.2.5 enum L4_vcon_ops

Operations on the vcon objects.

Enumerator

L4_VCON_WRITE_OP Write.

L4_VCON_SET_ATTR_OP Get console attributes.

L4_VCON_GET_ATTR_OP Set console attributes.

Definition at line 198 of file vcon.h.

9.71.3 Function Documentation

9.71.3.1 I4_msgtag_tI4_vcon_send(I4_cap_idx_t vcon, char const * buf, int 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.

Definition at line 222 of file vcon.h.

References I4_utcb().

Here is the call graph for this function:



9.71.3.2 long l4_vcon_write (l4_cap_idx_t vcon, char const * buf, int size) [inline]

Write data to virtual console.

9.71 Virtual Console 303

Parameters

vcon	Vcon object.	
buf	Pointer to data buffer.	
size	Size of buffer in bytes.	

Returns

Number of bytes written to the virtual console.

Definition at line 243 of file vcon.h.

References I4_utcb().

Here is the call graph for this function:



9.71.3.3 int I4_vcon_read (I4_cap_idx_t vcon, char * buf, int size) [inline]

Read data from virtual console.

Parameters

vcon	Vcon object.
buf	Pointer to data buffer.
size	Size of buffer in bytes.

Returns

Negative error code on error, > size if more to read, size bytes are in the buffer, <= size bytes read

Definition at line 280 of file vcon.h.

References I4_utcb().

Here is the call graph for this function:



9.71.3.4 I4_msgtag_tI4_vcon_set_attr(I4_cap_idx_t vcon, I4_vcon_attr_t const * attr) [inline]

Set attributes of a Vcon.

Parameters

vcon	Vcon object.	
attr		

Returns

Syscall return tag

Definition at line 300 of file vcon.h.

References I4_utcb().

Here is the call graph for this function:



9.71.3.5 I4_msgtag_t I4_vcon_get_attr(I4_cap_idx_t vcon, I4_vcon_attr_t * attr) [inline]

Get attributes of a Vcon.

Parameters

vcon	Vcon object.

Return values

attr	Attribute structure.

Returns

Syscall return tag

Definition at line 324 of file vcon.h.

References I4_utcb().

Here is the call graph for this function:



9.72 vCPU API 305

9.72 vCPU API

vCPU API

Collaboration diagram for vCPU API:



Data Structures

```
    struct I4_vcpu_state_t
        State of a vCPU.
    struct I4_vcpu_regs_t
        vCPU registers.
    struct I4_vcpu_ipc_regs_t
        vCPU message registers.
```

Typedefs

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.
```

9.72.1 Detailed Description

vCPU API

9.72.2 Enumeration Type Documentation

9.72.2.1 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 55 of file vcpu.h.

9.72.2.2 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 69 of file vcpu.h.

9.72.2.3 enum L4_vcpu_state_offset

Offsets for vCPU state layouts.

Enumerator

L4_VCPU_OFFSET_EXT_STATE Offset where extended state begins.

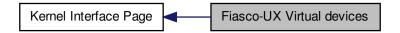
L4_VCPU_OFFSET_EXT_INFOS Offset where extended infos begin.

Definition at line 78 of file vcpu.h.

9.73 Fiasco-UX Virtual devices

Virtual hardware devices, provided by Fiasco-UX.

Collaboration diagram for Fiasco-UX Virtual devices:



Data Structures

struct I4_vhw_entry

Description of a device.

struct I4_vhw_descriptor

Virtual hardware devices description.

Enumerations

 enum l4_vhw_entry_type { L4_TYPE_VHW_NONE, L4_TYPE_VHW_FRAMEBUFFER, L4_TYPE_VHW_I-NPUT, L4_TYPE_VHW_NET }

Type of device.

9.73.1 Detailed Description

Virtual hardware devices, provided by Fiasco-UX. #include <14/sys/vhw.h>

9.73.2 Enumeration Type Documentation

9.73.2.1 enum I4_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.

9.74 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 I4_mem_read (unsigned long virtaddress, unsigned width)

Read memory from kernel privilege level.

• void I4_mem_write (unsigned long virtaddress, unsigned width, unsigned long value)

Write memory from kernel privilege level.

9.74.1 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 <14/sys/mem_op.h>
```

9.74.2 Enumeration Type Documentation

9.74.2.1 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.

9.74.3 Function Documentation

9.74.3.1 unsigned long I4_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.

9.74.3.2 void I4_mem_write (unsigned long virtaddress, unsigned width, unsigned long value) [inline]

Write memory from kernel privilege level.

Parameters

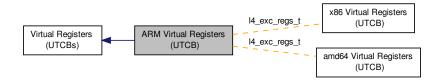
virtaddress	Virtual address in the calling task.
width	Width of access in bytes in log2 (i.e. allowed values: 0, 1, 2)
value	Value to write.

Upon an given invalid address or invalid width value the function does nothing.

Definition at line 147 of file mem_op.h.

9.75 ARM Virtual Registers (UTCB)

Collaboration diagram for ARM Virtual Registers (UTCB):



Data Structures

• struct I4_exc_regs_t

UTCB structure for exceptions.

Typedefs

• typedef struct I4_exc_regs_t I4_exc_regs_t UTCB structure for exceptions.

Enumerations

enum L4_utcb_consts_arm
 UTCB constants for ARM.

9.75.1 Detailed Description

9.76 VM API for TZ

Virtual Machine API for ARM TrustZone.

Collaboration diagram for VM API for TZ:



Data Structures

• struct I4_vm_tz_state

state structure for TrustZone VMs

9.76.1 Detailed Description

Virtual Machine API for ARM TrustZone.

9.77 amd64 Virtual Registers (UTCB)

Collaboration diagram for amd64 Virtual Registers (UTCB):



Data Structures

• struct I4_exc_regs_t

UTCB structure for exceptions.

Typedefs

typedef struct I4_exc_regs_t I4_exc_regs_t
 UTCB structure for exceptions.

Enumerations

enum L4_utcb_consts_amd64
 UTCB constants for AMD64.

9.77.1 Detailed Description

9.78 x86 Virtual Registers (UTCB)

Collaboration diagram for x86 Virtual Registers (UTCB):



Data Structures

struct I4_exc_regs_t
 UTCB structure for exceptions.

Typedefs

typedef struct I4_exc_regs_t I4_exc_regs_t
 UTCB structure for exceptions.

Enumerations

```
    enum L4_utcb_consts_x86 {
    L4_UTCB_EXCEPTION_REGS_SIZE = 16, 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(I4_umword_t), L4_UTCB_THREAD_REGS_OFFSET = 123 * sizeof(I4_umword_t), L4_UTCB_INHERIT_FPU = 1UL << 24, L4_UTCB_OFFSET = 512 }</li>
    UTCB constants for x86.
```

9.78.1 Detailed Description

9.78.2 Enumeration Type Documentation

9.78.2.1 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.

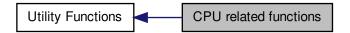
L4_UTCB_OFFSET Offset of two consecutive UTCBs.

Definition at line 41 of file utcb.h.

9.79 CPU related functions 315

9.79 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 *edx)

Generic CPUID access function.

9.79.1 Detailed Description

9.79.2 Function Documentation

9.79.2.1 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 I4util_cpu_capabilities().

Here is the caller graph for this function:



9.79.2.2 unsigned int l4util_cpu_capabilities (void) [inline]

Returns the CPU capabilities if the "cpuid" instruction is available.

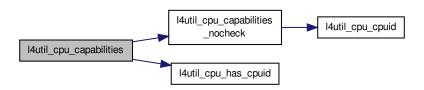
Returns

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 I4util_cpu_capabilities_nocheck(), and I4util_cpu_has_cpuid().

Here is the call graph for this function:



9.79.2.3 unsigned int l4util_cpu_capabilities_nocheck (void) [inline]

Returns the CPU capabilities.

Returns

CPU capabilities.

Definition at line 86 of file cpu.h.

References I4util_cpu_cpuid().

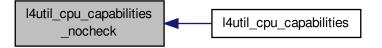
Referenced by I4util_cpu_capabilities().

Here is the call graph for this function:



9.79 CPU related functions 317

Here is the caller graph for this function:



9.80 Functions to manipulate the local IDT

Collaboration diagram for Functions to manipulate the local IDT:



Data Structures

- struct |4util_idt_desc_t | IDT entry.
- struct l4util_idt_header_t

Header of an IDT table.

9.80.1 Detailed Description

9.81 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

• I4_cpu_time_t I4_rdtsc (void)

Read current value of CPU-internal time stamp counter.

• I4_uint32_t I4_rdtsc_32 (void)

Read the lest significant 32 bit of the TSC.

I4_cpu_time_t I4_rdpmc (int nr)

Return current value of CPU-internal performance measurement counter.

• 14 uint32 t 14 rdpmc 32 (int nr)

Return the least significant 32 bit of a performance counter.

I4_uint64_t I4_tsc_to_ns (I4_cpu_time_t tsc)

Convert time stamp to ns value.

I4_uint64_t I4_tsc_to_us (I4_cpu_time_t tsc)

Convert time stamp into micro seconds value.

• void I4_tsc_to_s_and_ns (I4_cpu_time_t tsc, I4_uint32_t *s, I4_uint32_t *ns)

Convert time stamp to s.ns value.

I4_cpu_time_t I4_ns_to_tsc (I4_uint64_t ns)

Convert nano seconds into CPU ticks.

void I4_busy_wait_ns (I4_uint64_t ns)

Wait busy for a small amount of time.

void I4_busy_wait_us (I4_uint64_t us)

Wait busy for a small amount of time.

• I4_uint32_t I4_calibrate_tsc (I4_kernel_info_t *kip)

Calibrate scalers for time stamp calculations.

• I4_uint32_t I4_tsc_init (int constraint, I4_kernel_info_t *kip)

Initialitze scaler for TSC calicaltions.

I4_uint32_t I4_get_hz (void)

Get CPU frequency in Hz.

9.81.1 Detailed Description

9.81.2 Function Documentation

Read current value of CPU-internal time stamp counter.

Returns

64-bit time stamp

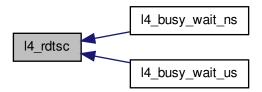
Examples:

examples/sys/aliens/main.c.

Definition at line 185 of file rdtsc.h.

Referenced by I4_busy_wait_ns(), and I4_busy_wait_us().

Here is the caller graph for this function:



Read the lest significant 32 bit of the TSC.

Useful for smaller differences, needs less cycles.

Definition at line 246 of file rdtsc.h.

9.81.2.3 I4_cpu_time_t I4_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.

```
9.81.2.4 I4_uint32_t I4_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.

```
9.81.2.5 I4_uint64_t I4_tsc_to_ns ( I4_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.

```
9.81.2.6 I4_uint64_t I4_tsc_to_us ( I4_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.

```
9.81.2.7 void I4_tsc_to_s_and_ns ( I4_cpu_time_t tsc, I4_uint32_t * s, I4_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.

```
9.81.2.8 I4_cpu_time_t I4_ns_to_tsc( I4_uint64_t ns ) [inline]
```

Convert nano seconds into CPU ticks.

Parameters

ns	nano seconds

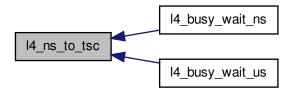
Returns

CPU ticks

Definition at line 303 of file rdtsc.h.

Referenced by I4_busy_wait_ns(), and I4_busy_wait_us().

Here is the caller graph for this function:



9.81.2.9 void I4_busy_wait_ns (I4_uint64_t ns) [inline]

Wait busy for a small amount of time.

Parameters

ns	nano seconds to wait

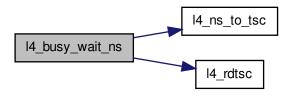
Attention

Not intendet for any use!

Definition at line 317 of file rdtsc.h.

References I4_ns_to_tsc(), and I4_rdtsc().

Here is the call graph for this function:



9.81.2.10 void I4_busy_wait_us (I4_uint64_t *us* **)** [inline]

Wait busy for a small amount of time.

Parameters

us	micro seconds to wait

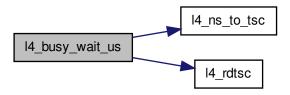
Attention

Not intendet for any use!

Definition at line 327 of file rdtsc.h.

References I4_ns_to_tsc(), and I4_rdtsc().

Here is the call graph for this function:



9.81.2.11 I4_uint32_t I4_calibrate_tsc (I4_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 I4_tsc_init(L4_TSC_INIT_AUTO).

Examples:

examples/sys/aliens/main.c.

Definition at line 179 of file rdtsc.h.

References I4_tsc_init(), and L4_TSC_INIT_AUTO.

Here is the call graph for this function:



9.81.2.12 I4_uint32_t I4_tsc_init (int constraint, I4_kernel_info_t * kip)

Initialitze scaler for TSC calicaltions.

Initialize the scalers needed by I4_tsc_to_ns()/I4_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 I4_calibrate_tsc().

Here is the caller graph for this function:



9.81.2.13 I4_uint32_t l4_get_hz (void)

Get CPU frequency in Hz.

Returns

frequency in Hz

9.82 Atomic Instructions 327

9.82 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)
```

```
    I4_uint32_t I4util_xchg32 (volatile I4_uint32_t *dest, I4_uint32_t val)
```

Atomic exchange (32 bit version)

I4_uint16_t I4util_xchg16 (volatile I4_uint16_t *dest, I4_uint16_t val)

Atomic exchange (16 bit version)

I4_uint8_t I4util_xchg8 (volatile I4_uint8_t *dest, I4_uint8_t val)

Atomic exchange (8 bit version)

• I4_umword_t I4util_xchg (volatile I4_umword_t *dest, I4_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 |4util_add8 (volatile |4_uint8_t *dest, |4_uint8_t val)
```

- void I4util_add16 (volatile I4_uint16_t *dest, I4_uint16_t val)
- void I4util_add32 (volatile I4_uint32_t *dest, I4_uint32_t val)
- void I4util_sub8 (volatile I4_uint8_t *dest, I4_uint8_t val)
- void I4util_sub16 (volatile I4_uint16_t *dest, I4_uint16_t val)

```
void I4util_sub32 (volatile I4_uint32_t *dest, I4_uint32_t val)
void I4util_and8 (volatile I4_uint8_t *dest, I4_uint8_t val)
void I4util_and16 (volatile I4_uint16_t *dest, I4_uint16_t val)
void I4util_and32 (volatile I4_uint32_t *dest, I4_uint32_t val)
void I4util_or8 (volatile I4_uint8_t *dest, I4_uint8_t val)
void I4util_or16 (volatile I4_uint16_t *dest, I4_uint16_t val)
```

void I4util_or32 (volatile I4_uint32_t *dest, I4_uint32_t val)

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

```
I4_uint8_t I4util_add8_res (volatile I4_uint8_t *dest, I4_uint8_t val)
I4_uint16_t I4util_add16_res (volatile I4_uint16_t *dest, I4_uint16_t val)
I4_uint32_t I4util_add32_res (volatile I4_uint32_t *dest, I4_uint32_t val)
I4_uint8_t I4util_sub8_res (volatile I4_uint8_t *dest, I4_uint8_t val)
I4_uint16_t I4util_sub16_res (volatile I4_uint16_t *dest, I4_uint16_t val)
I4_uint32_t I4util_sub32_res (volatile I4_uint32_t *dest, I4_uint32_t val)
I4_uint8_t I4util_and8_res (volatile I4_uint8_t *dest, I4_uint8_t val)
I4_uint16_t I4util_and16_res (volatile I4_uint16_t *dest, I4_uint16_t val)
I4_uint32_t I4util_and32_res (volatile I4_uint32_t *dest, I4_uint82_t val)
I4_uint8_t I4util_or8_res (volatile I4_uint8_t *dest, I4_uint8_t val)
I4_uint16_t I4util_or16_res (volatile I4_uint16_t *dest, I4_uint16_t val)
I4_uint32_t I4util_or26_res (volatile I4_uint16_t *dest, I4_uint32_t val)
I4_uint32_t I4util_or36_res (volatile I4_uint16_t *dest, I4_uint36_t val)
I4_uint36_t I4util_or36_res (volatile I4_uint16_t *dest, I4_uint36_t val)
```

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

```
void I4util_inc8 (volatile I4_uint8_t *dest)
void I4util_inc16 (volatile I4_uint16_t *dest)
void I4util_inc32 (volatile I4_uint32_t *dest)
void I4util_dec8 (volatile I4_uint8_t *dest)
void I4util_dec16 (volatile I4_uint16_t *dest)
void I4util_dec32 (volatile I4_uint32_t *dest)
```

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

```
    I4_uint8_t | 4util_inc8_res (volatile | 4_uint8_t *dest)
    I4_uint16_t | I4util_inc16_res (volatile | I4_uint16_t *dest)
    I4_uint32_t | I4util_inc32_res (volatile | I4_uint32_t *dest)
    I4_uint8_t | I4util_dec8_res (volatile | I4_uint8_t *dest)
    I4_uint16_t | I4util_dec16_res (volatile | I4_uint16_t *dest)
    I4_uint32_t | I4util_dec32_res (volatile | I4_uint32_t *dest)
```

9.82.1 Detailed Description

9.82.2 Function Documentation

```
9.82.2.1 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)

9.82 Atomic Instructions 329

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

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 24 of file atomic_arch.h.

9.82.2.3 int |4util_cmpxchg16 (volatile |4_uint16_t * dest, |4_uint16_t cmp_val, |4_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

9.82.2.4 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

```
9.82.2.5 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

Atomic exchange (32 bit version)

Parameters

dest	destination operand
val	new value for dest

Returns

old value at destination

Atomic exchange (16 bit version)

Parameters

dest	destination operand
val	new value for dest

Returns

old value at destination

```
9.82.2.8 I4_uint8_t I4util_xchg8 ( volatile I4_uint8_t * dest, I4_uint8_t val ) [inline]
```

Atomic exchange (8 bit version)

9.82 Atomic Instructions 331

Parameters

dest	destination operand
val	new value for dest

Returns

old value at destination

9.82.2.9 I4_umword_t I4util_xchg (volatile I4_umword_t * dest, I4_umword_t val) [inline]

Atomic exchange (machine wide fields)

Parameters

dest	destination operand
val	new value for dest

Returns

old value at destination

9.82.2.10 void | 4util_add8 (volatile | 4_uint8_t * dest, | 4_uint8_t val) [inline]

Parameters

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

9.82.2.11 I4_uint8_t I4util_add8_res (volatile I4_uint8_t * dest, I4_uint8_t val) [inline]

Parameters

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

Returns

res

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

Parameters

dest destination operand	
--------------------------	--

9.82.2.13 I4_uint8_t I4util_inc8_res (volatile I4_uint8_t * dest) [inline]

Parameters

dest	destination operand

Returns

res

9.82.2.14 void | | void | | void | vo

Atomic add.

Parameters

dest	destination operand
val	value to add

Definition at line 54 of file atomic_arch.h.

9.82.2.15 void | | void | void

Atomic increment.

Parameters

dest	destination operand
------	---------------------

Definition at line 61 of file atomic_arch.h.

9.83 Internal functions 333

9.83 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

9.83.1 Detailed Description

9.84 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 I4util_find_first_set_bit (const void *dest, I4_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.

9.84.1 Detailed Description

9.84.2 Function Documentation

9.84.2.1 void | | 4util_set_bit (int b, volatile | 4 umword t * dest) [inline]

Set bit in memory.

9.84 Bit Manipulation 335

Parameters

b	bit position
dest	destination operand

9.84.2.2 void l4util_clear_bit (int b, volatile l4_umword_t * dest) [inline]

Clear bit in memory.

Parameters

b	bit position
dest	destination operand

9.84.2.3 void l4util_complement_bit (int b, volatile l4_umword_t * dest) [inline]

Complement bit in memory.

Parameters

b	bit position
dest	destination operand

Test bit (return value of bit)

Parameters

b	bit position
dest	destination operand

Returns

Value of bit b.

9.84.2.5 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.

9.84.2.6 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.

9.84.2.7 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.

9.84.2.8 int l4util_bsr(l4_umword_t word) [inline]

Bit scan reverse.

Parameters

word	value (machine size)

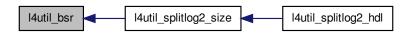
Returns

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))

Referenced by I4util_splitlog2_size().

Here is the caller graph for this function:



9.84.2.9 int | 4util_bsf (| 14_umword_t word) [inline]

Bit scan forward.

9.84 Bit Manipulation 337

Parameters

word	value (machine size)
------	----------------------

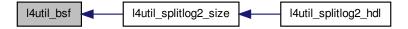
Returns

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.

Referenced by I4util_splitlog2_size().

Here is the caller graph for this function:



9.84.2.10 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

9.84.2.11 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

9.84.2.12 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 408 of file bitops.h.

9.85 ELF binary format 339

9.85 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 El_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.

9.85 ELF binary format 341

```
    #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.

9.85 ELF binary format 343

```
Motorola Coldfire.
• #define EM_68HC12 53
     Motorola M68HC12.

    #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

    #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
```

#define EM_COLDFIRE 52

```
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.

9.85 ELF binary format 345

 #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 Iwpinfo 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.

Generated for L4Re by Doxygen

9.85 ELF binary format 347

#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
• #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

• #define R_386_JMP_SLOT 7
• #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

9.85 ELF binary format 349

```
    #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 I4_uint32_t Elf32_Addr

          size 4 align 4
    • typedef I4_uint32_t Elf32_Off
          size 4 align 4

    typedef I4_uint16_t Elf32_Half

          size 2 align 2
    • typedef I4_uint32_t Elf32_Word
          size 4 align 4
    • typedef I4_int32_t Elf32_Sword
          size 4 align 4

    typedef I4_uint64_t Elf64_Addr

          size 8 align 8

    typedef I4 uint64 t Elf64 Off

          size 8 align 8

    typedef I4_uint16_t Elf64_Half

          size 2 align 2
    • typedef I4_uint32_t Elf64_Word
          size 4 align 4
    • typedef I4_int32_t Elf64_Sword
          size 4 align 4

    typedef I4_uint64_t Elf64_Xword

          size 8 align 8

    typedef I4_int64_t Elf64_Sxword

          size 8 align 8
```

9.85.1 Detailed Description

Functions and types related to ELF binaries.

9.85.2 Macro Definition Documentation

9.85.2.1 #define EI_CLASS 4

ELF class byte index.

file class

Definition at line 254 of file elf.h.

9.85.2.2 #define EI_CLASS 4

ELF class byte index.

file class

Definition at line 254 of file elf.h.

9.85.2.3 #define ELFCLASSNONE 0

Invalid ELF class.

Invalid class.

Definition at line 270 of file elf.h.

9.85.2.4 #define ELFCLASSNONE 0

Invalid ELF class.

Invalid class.

Definition at line 270 of file elf.h.

9.85.2.5 #define EI_DATA 5

Data encoding byte index.

data encoding

Definition at line 255 of file elf.h.

9.85.2.6 #define EI_DATA 5

Data encoding byte index.

data encoding

Definition at line 255 of file elf.h.

9.85.2.7 #define ELFDATANONE 0

Invalid data encoding.

invalid data encoding

Definition at line 276 of file elf.h.

9.85.2.8 #define ELFDATANONE 0

Invalid data encoding.

invalid data encoding

Definition at line 276 of file elf.h.

9.85.2.9 #define ELFDATA2LSB 1

2's complement, little endian

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

Definition at line 277 of file elf.h.

9.85.2.10 #define ELFDATA2LSB 1

2's complement, little endian

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

Definition at line 277 of file elf.h.

9.85.2.11 #define ELFDATA2MSB 2

2's complement, big endian

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

Definition at line 278 of file elf.h.

9.85.2.12 #define ELFDATA2MSB 2

2's complement, big endian

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

Definition at line 278 of file elf.h.

9.85.2.13 #define EI_VERSION 6

File version byte index.

file version

Value must be EV_CURRENT

Definition at line 256 of file elf.h.

9.85.2.14 #define EI_VERSION 6

File version byte index.

file version

Value must be EV_CURRENT

Definition at line 256 of file elf.h.

9.85.2.15 #define EI_OSABI 7

OS ABI identification.

Operating system / ABI identification.

Definition at line 257 of file elf.h.

9.85.2.16 #define EI_OSABI 7

OS ABI identification.

Operating system / ABI identification.

Definition at line 257 of file elf.h.

9.85.2.17 #define ELFOSABI_SYSV 0

Alias.

UNIX System V ABI (this specification)

Definition at line 282 of file elf.h.

9.85.2.18 #define ELFOSABI_SYSV 0

Alias.

UNIX System V ABI (this specification)

Definition at line 282 of file elf.h.

9.85.2.19 #define ELFOSABI_HPUX 1

HP-UX.

HP-UX operating system.

Definition at line 283 of file elf.h.

9.85.2.20 #define ELFOSABI_HPUX 1

HP-UX.

HP-UX operating system.

Definition at line 283 of file elf.h.

9.85.2.21 #define ELFOSABI_NETBSD 2

NetBSD.

Definition at line 174 of file elf.h.

9.85.2.22 #define ELFOSABI_LINUX 3

Linux.

Definition at line 175 of file elf.h.

9.85.2.23 #define ELFOSABI_SOLARIS 6 Sun Solaris. Definition at line 176 of file elf.h. 9.85.2.24 #define ELFOSABI_AIX 7 IBM AIX. Definition at line 177 of file elf.h. 9.85.2.25 #define ELFOSABI_IRIX 8 SGI Irix. Definition at line 178 of file elf.h. 9.85.2.26 #define ELFOSABI_FREEBSD 9 FreeBSD. Definition at line 179 of file elf.h. 9.85.2.27 #define ELFOSABI_TRU64 10 Compaq TRU64 UNIX. Definition at line 180 of file elf.h. 9.85.2.28 #define ELFOSABI_MODESTO 11 Novell Modesto. Definition at line 181 of file elf.h. 9.85.2.29 #define ELFOSABI_OPENBSD 12 OpenBSD. Definition at line 182 of file elf.h. 9.85.2.30 #define EI_PAD 9 Byte index of padding bytes. start of padding bytes Definition at line 259 of file elf.h. 9.85.2.31 #define EI_PAD 9

Generated for L4Re by Doxygen

start of padding bytes

Byte index of padding bytes.

Definition at line 259 of file elf.h.

9.85.2.32 #define EM_ARC 45 Argonaut RISC Core, Argonaut Techn Inc. Definition at line 226 of file elf.h. 9.85.2.33 #define SHT_NUM 19 Number of defined types. Definition at line 348 of file elf.h. 9.85.2.34 #define SHF_GROUP 0x200 Section is member of a group. Definition at line 368 of file elf.h. 9.85.2.35 #define SHF_TLS 0x400 Section hold thread-local data. Definition at line 369 of file elf.h. 9.85.2.36 #define SHF_MASKOS 0x0ff00000 OS-specific. Definition at line 370 of file elf.h. #define PT_LOOS 0x60000000 9.85.2.37 os spec. Definition at line 413 of file elf.h. 9.85.2.38 #define PT_HIOS 0x6fffffff os spec. Definition at line 414 of file elf.h. 9.85.2.39 #define PT_LOPROC 0x70000000 processor spec. Definition at line 415 of file elf.h. 9.85.2.40 #define PT_HIPROC 0x7fffffff processor spec.

Definition at line 416 of file elf.h.

```
9.85.2.41 #define PT_GNU_EH_FRAME (PT_LOOS + 0x474e550)
EH frame information.
Definition at line 418 of file elf.h.
9.85.2.42 #define PT_GNU_STACK (PT_LOOS + 0x474e551)
Flags for stack.
Definition at line 419 of file elf.h.
9.85.2.43 #define PT_GNU_RELRO (PT_LOOS + 0x474e552)
Read only after reloc.
Definition at line 420 of file elf.h.
9.85.2.44 #define PT_L4_STACK (PT_LOOS + 0x12)
Address of the stack.
Definition at line 422 of file elf.h.
9.85.2.45 #define PT_L4_KIP (PT_LOOS + 0x13)
Address of the KIP.
Definition at line 423 of file elf.h.
9.85.2.46 #define PT_L4_AUX (PT_LOOS + 0x14)
Address of the AUX strcutures.
Definition at line 424 of file elf.h.
9.85.2.47 #define NT_VERSION 1
Contains a version string.
Definition at line 455 of file elf.h.
9.85.2.48 #define DT_NULL 0
Dynamic Array Tags, d_tag - figure 2-10, page 2-12.
end of _DYNAMIC array
Definition at line 479 of file elf.h.
9.85.2.49 #define DT_LOPROC 0x70000000
processor spec.
Definition at line 516 of file elf.h.
```

```
9.85.2.50 #define DT_HIPROC 0x7fffffff
processor spec.
Definition at line 517 of file elf.h.
9.85.2.51 #define DF_1_NOW 0x00000001
Set RTLD_NOW for this object.
Definition at line 528 of file elf.h.
9.85.2.52 #define DF_1_GLOBAL 0x00000002
Set RTLD_GLOBAL for this object.
Definition at line 529 of file elf.h.
9.85.2.53 #define DF_1_GROUP 0x00000004
Set RTLD_GROUP for this object.
Definition at line 530 of file elf.h.
9.85.2.54 #define DF_1_NODELETE 0x00000008
Set RTLD_NODELETE for this object.
Definition at line 531 of file elf.h.
9.85.2.55 #define DF_1_LOADFLTR 0x00000010
Trigger filtee loading at runtime.
Definition at line 532 of file elf.h.
9.85.2.56 #define DF_1_NOOPEN 0x00000040
Set RTLD_NOOPEN for this object.
Definition at line 534 of file elf.h.
9.85.2.57 #define DF_1_ORIGIN 0x00000080
$ORIGIN must be handled.
Definition at line 535 of file elf.h.
9.85.2.58 #define DF_1_DIRECT 0x00000100
Direct binding enabled.
```

Definition at line 536 of file elf.h.

9.85 ELF binary format 357

```
9.85.2.59 #define DF_1_INTERPOSE 0x00000400
Object is used to interpose.
Definition at line 538 of file elf.h.
9.85.2.60 #define DF_1_NODEFLIB 0x00000800
Ignore default lib search path.
Definition at line 539 of file elf.h.
9.85.2.61 #define DF_1_NODUMP 0x00001000
Object can't be dldump'ed.
Definition at line 540 of file elf.h.
9.85.2.62 #define DF_1_CONFALT 0x00002000
Configuration alternative created.
Definition at line 541 of file elf.h.
9.85.2.63 #define DF_1_ENDFILTEE 0x00004000
Filtee terminates filters search.
Definition at line 542 of file elf.h.
9.85.2.64 #define DF_1_DISPRELDNE 0x00008000
Disp reloc applied at build time.
Definition at line 543 of file elf.h.
9.85.2.65 #define DF_1_DISPRELPND 0x00010000
Disp reloc applied at run-time.
Definition at line 544 of file elf.h.
9.85.2.66 #define DF_P1_LAZYLOAD 0x00000001
Lazyload following object.
Definition at line 551 of file elf.h.
9.85.2.67 #define DF_P1_GROUPPERM 0x00000002
```

Definition at line 552 of file elf.h.

Symbols from next object are not generally available.

9.86 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.

• I4_addr_t I4util_memdesc_vm_high (I4_kernel_info_t *kinfo)

Return end of virtual memory.

9.86.1 Detailed Description

9.86.2 Macro Definition Documentation

```
9.86.2.1 #define | 4util_kip_for_each_feature(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 I4util_kip_version-string().

Definition at line 74 of file kip.h.

9.86.3 Function Documentation

```
9.86.3.1 int |4util_kip_kernel_is_ux ( |4 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.

```
9.86.3.2 int |4uti|_{kip_{kernel_has_feature}} ( |4_{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.

```
9.86.3.3 unsigned long l4util_kip_kernel_abi_version ( I4_kernel_info_t * )
```

Return kernel ABI version.

Returns

Kernel ABI version.

```
9.86.3.4 I4_addr_t | 4util_memdesc_vm_high ( I4_kernel_info_t * kinfo )
```

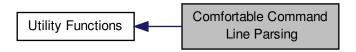
Return end of virtual memory.

Returns

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

9.87 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.

9.87.1 Detailed Description

9.87.2 Function Documentation

```
9.87.2.1 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 preceded 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 preceded 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.

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 <code>getoptlong</code> for details on scanning.

9.88 Priority related functions

Collaboration diagram for Priority related functions:



9.88.1 Detailed Description

9.89 Random number support

Collaboration diagram for Random number support:



Functions

• I4_uint32_t I4util_rand (void)

Deliver next random number.

void l4util_srand (l4_uint32_t seed)

Initialize random number generator.

9.89.1 Detailed Description

9.89.2 Function Documentation

9.89.2.1 I4_uint32_t l4util_rand (void)

Deliver next random number.

Returns

A new random number

9.89.2.2 void I4util_srand (I4_uint32_t seed)

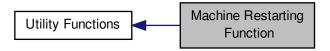
Initialize random number generator.

Parameters

seed Value to initialize

9.90 Machine Restarting Function

Collaboration diagram for Machine Restarting Function:



Functions

void |4util_reboot (void))
 Machine reboot.

9.90.1 Detailed Description

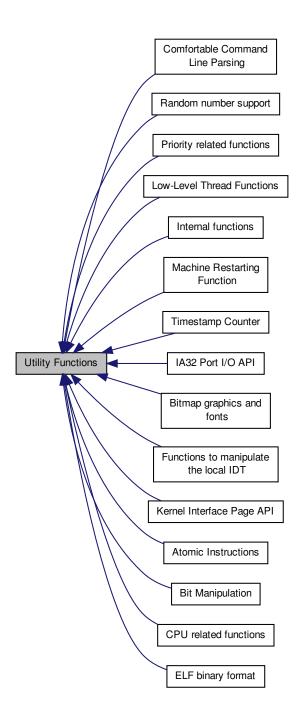
9.91 Low-Level Thread Functions

Collaboration diagram for Low-Level Thread Functions:



9.92 Utility Functions

Collaboration diagram for Utility Functions:



Modules

- · Atomic Instructions
- Bit Manipulation
- · Bitmap graphics and fonts

9.92 Utility Functions 367

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
- · Priority related functions
- · Random number support
- Timestamp Counter

Files

· file rand.h

Simple Pseudo-Random Number Generator.

Functions

void I4_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.
- I4_addr_t I4util_splitlog2_size (I4_addr_t start, I4_addr_t end)

Return log2 base and size aligned length of a range.

• I4_timeout_s I4util_micros2I4to (unsigned int mus) L4_NOTHROW

Calculate I4 timeouts.

9.92.1 Detailed Description

9.92.2 Function Documentation

9.92.2.1 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 I4util_splitlog2_size().

Here is the call graph for this function:



9.92.2.2 I4_addr_t I4util_splitlog2_size(I4_addr_t start, I4_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

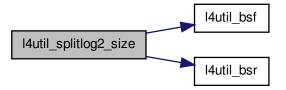
length of elements in log2size (length is 1 << log2size)

Definition at line 72 of file splitlog2.h.

References I4util_bsf(), and I4util_bsr().

Referenced by I4util_splitlog2_hdl().

Here is the call graph for this function:



9.92 Utility Functions 369

Here is the caller graph for this function:



9.92.2.3 I4_timeout_s I4util_micros2I4to (unsigned int mus)

Calculate I4 timeouts.

Parameters

mus	time in microseconds. Special cases:
	• 0 - > timeout 0
	• ~0U -> timeout NEVER

Returns

the corresponding I4_timeout value

9.93 IA32 Port I/O API

Collaboration diagram for IA32 Port I/O API:



Functions

```
• I4_uint8_t I4util_in8 (I4_uint16_t port)
```

Read byte from I/O port.

I4_uint16_t I4util_in16 (I4_uint16_t port)

Read 16-bit-value from I/O port.

• I4_uint32_t I4util_in32 (I4_uint16_t port)

Read 32-bit-value from I/O port.

• void |4util_ins8 (|4_uint16_t port, |4_umword_t addr, |4_umword_t count)

Read a block of 8-bit-values from I/O ports.

• void |4util_ins16 (|4_uint16_t port, |4_umword_t addr, |4_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 I4util_out8 (I4_uint8_t value, I4_uint16_t port)

Write byte to I/O port.

• void I4util_out16 (I4_uint16_t value, I4_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 |4util_outs8 (|4_uint16_t port, |4_umword_t addr, |4_umword_t count)

Write a block of bytes to I/O port.

void |4util outs16 (|4 uint16 t port, |4 umword t addr, |4 umword t count)

Write a block of 16-bit-values to I/O port.

void |4util_outs32 (|4_uint16_t port, |4_umword_t addr, |4_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

9.93.1 Detailed Description

9.93.2 Function Documentation

9.93.2.1 I4_uint8_t I4util_in8 (I4_uint16_t port) [inline]

Read byte from I/O port.

9.93 IA32 Port I/O API 371

Parameters

port	I/O port address

Returns

value

Definition at line 172 of file port_io.h.

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.

9.93.2.3 l4_uint32_t l4util_in32 (l4_uint16_t *port* **)** [inline]

Read 32-bit-value from I/O port.

Parameters

port	I/O port address
port	70 port address

Returns

value

Definition at line 188 of file port_io.h.

9.93.2.4 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.

9.93.2.5 void I4util_ins16 (I4_uint16_t port, I4_umword_t addr, I4_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.

9.93.2.6 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.

9.93.2.7 void | Hutil_out8 (| H_uint8_t value, | H_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.

9.93.2.8 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.

9.93.2.9 void | | 4util_out32 (| 14_uint32_t value, | 14_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.

9.93.2.10 void l4util_outs8 (I4_uint16_t port, I4_umword_t addr, I4_umword_t count) [inline]

Write a block of bytes to I/O port.

9.93 IA32 Port I/O API 373

Parameters

	port	I/O port address
ſ	addr	address of buffer
ĺ	count	number of I/O operations

Definition at line 241 of file port_io.h.

9.93.2.11 void |4util_outs16 (|4_uint16_t port, |4_umword_t addr, |4_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.

9.93.2.12 void | | void | | 4_uint16_t port, | 14_umword_t addr, | 14_umword_t count | [inline]

Write block of 32-bit-values to I/O port.

Parameters

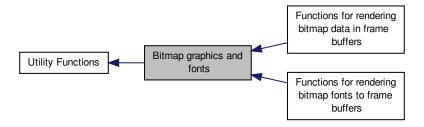
port	I/O port address
addr	address of buffer
count	number of I/O operations

Definition at line 259 of file port_io.h.

9.94 Bitmap graphics and fonts

This library provides some functions for bitmap handling in frame buffers.

Collaboration diagram for Bitmap graphics and fonts:



Modules

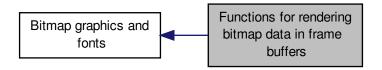
- · Functions for rendering bitmap data in frame buffers
- Functions for rendering bitmap fonts to frame buffers

9.94.1 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.

9.95 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 (I4_uint8_t *vfb, I4re_video_view_info_t *vi, I4_int16_t x, I4_int16_t y, I4_uint32_t w, I4_uint32_t h, I4_uint8_t *bmap, gfxbitmap_color_pix_t fgc, gfxbitmap_color_pix_t bgc, struct gfxbitmap_offset *offset, I4_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 (I4_uint8_t *dest, I4_uint8_t *src, I4re_video_view_info_t *vi, int x, int y, int w, int h, int dx, int dy)

Copy a rectangular area.

9.95.1 Detailed Description

9.95.2 Typedef Documentation

9.95.2.1 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.

9.95.2.2 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.

9.95.3 Function Documentation

9.95.3.1 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.

Fill a rectangular area with a color.

Parameters

vfb	Frame buffer.
fbi	Frame buffer information structure.
X	X position of area.
У	Y position of area.
W	Width of area.
h	Height of area.
color	Color of area.

9.95.3.3 void gfxbitmap_bmap (I4_uint8_t * vfb, I4re_video_view_info_t * vi, I4_int16_t x, I4_int16_t y, I4_uint32_t w, I4_uint32_t h, I4_uint8_t * bmap, gfxbitmap_color_pix_t fgc, gfxbitmap_color_pix_t bgc, struct gfxbitmap_offset * offset, I4_uint8_t mode)

Fill a rectangular area with a bicolor bitmap pattern.

Parameters

vfb	Frame buffer.
fbi	Frame buffer information structure.
X	X position of area.
У	Y position of area.
W	Width of area.

h	Height of area.
bmap	Bitmap pattern.
fgc	Foreground color.
bgc	Background color.
offset	Offsets.
mode	Mode (

See Also

 $\#pSLIM_BMAP_START_MSB$ and $*\#pSLIM_BMAP_START_LSB$).

9.95.3.4 void gfxbitmap_set ($14_uint8_t * vfb$, $14re_video_view_info_t * vi$, $14_uint16_t x$, $14_uint16_t y$, $14_uint32_t w$, $14_uint32_t h$, $14_uint32_t xoffs$, $14_uint32_t yoffs$, $14_uint8_t * pmap$, struct gfxbitmap_offset * offset, $14_uint32_t pwidth$)

Set area from source area.

Parameters

vfb	Frame buffer.
fbi	Frame buffer information structure.
X	X position of area.
У	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.

9.95.3.5 void gfxbitmap_copy ($14_uint8_t * dest$, $14_uint8_t * src$, $14re_video_view_info_t * vi$, int x, int y, int y

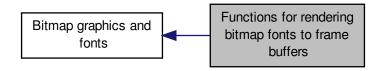
Copy a rectangular area.

Parameters

dest	Destination frame buffer.
src	Source frame buffer.
fbi	Frame buffer information structure.
X	Source X position of area.
У	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.

9.96 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.

9.96.1 Detailed Description

9.96.2 Enumeration Type Documentation

9.96.2.1 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.

9.96.3 Function Documentation

9.96.3.1 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

9.96.3.2 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.

9.96.3.3 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.

9.96.3.4 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.

9.96.3.5 void* gfxbitmap_font_data (gfxbitmap_font_t font, unsigned c)

Get bitmap font data for a specific character.

Parameters

font	Font.
С	Character.

Returns

Pointer to bmap data, NULL on error.

9.96.3.6 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.
fbi	Frame buffer info structure.
font	Font.
text	Text string.
len	Length of the text string.
X	Horizontal position in the frame buffer.
У	Vertical position in the frame buffer.
fg	Foreground color.
bg	Background color.

9.96.3.7 void gfxbitmap_font_text_scale (void * fb, I4re_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.
fbi	Frame buffer info structure.
font	Font.
text	Text string.
len	Length of the text string.

X	Horizontal position in the frame buffer.
У	Vertical position in the frame buffer.
fg	Foreground color.
bg	Background color.
scale_x	Horizonal scale factor.
scale_y	Vertical scale factor.

9.97 IO interface

Typedefs

```
    typedef l4vbus_resource_t l4io_resource_t
        Resource descriptor.
    typedef l4vbus_device_t l4io_device_t
        Device descriptor.
```

Enumerations

```
enum I4io_iomem_flags_t {
    L4IO_MEM_NONCACHED = 0, L4IO_MEM_CACHED = 1, L4IO_MEM_USE_MTRR = 2, L4IO_MEM_USE_RESERVED_AREA = 0x40 << 8,
    L4IO_MEM_EAGER_MAP = 0x80 << 8 }
    Flags for IO memory.</li>
enum I4io_device_types_t {
    L4IO_DEVICE_INVALID = 0, L4IO_DEVICE_PCI, L4IO_DEVICE_USB, L4IO_DEVICE_OTHER,
    L4IO_DEVICE_ANY = ~0 }
    Device types.
enum I4io_resource_types_t {
    L4IO_RESOURCE_INVALID = L4VBUS_RESOURCE_INVALID, L4IO_RESOURCE_IRQ = L4VBUS_RESOURCE_IRQ, L4IO_RESOURCE_PORT = L4VBUS_RESOURCE_MEM, L4IO_RESOURCE_PORT = L4VBUS_RESOURCE_PORT,
    L4IO_RESOURCE_ANY = ~0 }
    Resource types.
```

Functions

- long L4_EXPORT l4io_request_iomem (l4_addr_t phys, unsigned long size, int flags, l4_addr_t *virt)
 Request an IO memory region.
- long L4_EXPORT l4io_request_iomem_region (l4_addr_t phys, l4_addr_t virt, unsigned long size, int flags)

 Request an IO memory region and map to a specified region.
- long L4_EXPORT l4io_release_iomem (l4_addr_t virt, unsigned long size)

Release an IO memory region.

long L4_EXPORT 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 L4_EXPORT l4io_request_ioport (unsigned portnum, unsigned len)

Request an IO port region.

long L4_EXPORT l4io_release_ioport (unsigned portnum, unsigned len)

Release an IO port region.

• 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)

Find a device by name.

• 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)

Request a specific resource from a device description.

I4_addr_t L4_EXPORT I4io_request_resource_iomem (I4io_device_handle_t devhandle, I4io_resource_handle_t *reshandle)

Request IO memory.

• int L4_EXPORT l4io_has_resource (enum l4io_resource_types_t type, l4vbus_paddr_t start, l4vbus_paddr_t end)

Check if a resource is available.

9.97 IO interface 383

```
9.97.1
        Detailed Description
9.97.2
       Typedef Documentation
9.97.2.1 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.
9.97.3 Enumeration Type Documentation
9.97.3.1 enum l4io_iomem_flags_t
Flags for IO memory.
Enumerator
    L4IO_MEM_NONCACHED Non-cache memory.
    L4IO_MEM_CACHED Cache memory.
    L4IO_MEM_USE_MTRR Use MTRR.
    L4IO_MEM_USE_RESERVED_AREA Use reserved area for mapping I/O memory. Flag only valid for I4io_-
         request_iomem_region()
    L4IO_MEM_EAGER_MAP Eagerly map the I/O memory. Passthrough to the I4re-rm.
Definition at line 16 of file types.h.
9.97.3.2 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.
9.97.3.3 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
```

Generated for L4Re by Doxygen

Definition at line 50 of file types.h.

9.97.4 Function Documentation

9.97.4.1 long L4_EXPORT 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

Return values

virt	Virtual address the region is available at.

Returns

0 on success, <0 on error

Note

This function uses L4Re functionality to reserve a part of the virtual address space of the caller.

9.97.4.2 long L4_EXPORT l4io_request_iomem_region (I4_addr_t phys, I4_addr_t virt, unsigned long size, int flags)

Request an IO memory region and map 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

Returns

0 on success, <0 on error

Note

This function uses L4Re functionality to reserve a part of the virtual address space of the caller.

9.97.4.3 long L4 EXPORT l4io_release_iomem (I4 addr_t virt, unsigned long size)

Release an IO memory region.

Parameters

virt	Virtual address of region to free, see I4io_request_iomem
size	Size of the region to release.

Returns

0 on success, <0 on error

9.97 IO interface 385

9.97.4.4 long L4_EXPORT l4io_search_iomem_region (l4_addr_t phys, l4_addr_t size, l4_addr_t * rstart, l4_addr_t * rstze)

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

9.97.4.5 long L4_EXPORT 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

9.97.4.6 long L4_EXPORT 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

9.97.4.7 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)

Find a device by name.

9.97 IO interface 387

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

9.97.4.8 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)

Request a specific resource from a device description.

Parameters

devhan	dle	Device handle.
ty	уре	Type of resource to request (see #l4io_resource_types_t)
reshan	dle	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

9.97.4.9 I4_addr_t L4_EXPORT l4io_request_resource_iomem (l4io_device_handle_t devhandle, l4io_resource_handle_t * reshandle)

Request IO memory.

Parameters

devhandle	Device handle.

Return values

reshandle	Resource handle, input and ouput, return next resource handle

Returns

0 on error, virtual address otherwise

9.97.4.10 int L4_EXPORT l4io_has_resource (enum l4io_resource_types_t type, l4vbus_paddr_t start, l4vbus_paddr_t end)

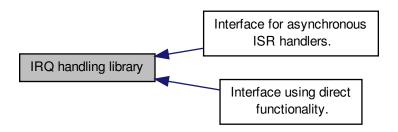
Check if a resource is available.

Parameters

type	Type of resource
start	Minimal value.
end	Maximum value.

9.98 IRQ handling library

Collaboration diagram for IRQ handling library:



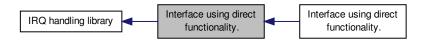
Modules

- Interface for asynchronous ISR handlers.
 This interface has just two (main) functions.
- · Interface using direct functionality.

9.98.1 Detailed Description

9.99 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.

• I4irq_t * I4irq_attach_ft (int irqnum, unsigned mode)

Attach/connect to IRQ using given type.

I4irq_t * I4irq_attach_thread (int irqnum, I4_cap_idx_t to_thread)

Attach/connect to IRQ.

• I4irq_t * I4irq_attach_thread_ft (int irqnum, I4_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.

9.99.1 Detailed Description

9.99.2 Function Documentation

9.99.2.1 | I4irq_t* | I4irq_attach (int irqnum)

Attach/connect to IRQ.

Parameters

irqnum	IRQ number to request

Returns

Pointer to l4irq_t structure, 0 on error

This I4irq_attach has to be called in the same thread as I4irq_wait and caller has to be a pthread thread.

Examples:

examples/libs/libirq/loop.c.

9.99.2.2 | 14irq_t* | 14irq_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 l4irg t structure, 0 on error

This I4irq_attach has to be called in the same thread as I4irq_wait and caller has to be a pthread 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.

9.99.2.4 | I4irq_t* | I4irq_attach_thread_ft (int irqnum, I4_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.

```
9.99.2.5 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.

```
9.99.2.6 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

```
9.99.2.7 long l4irq_wait_any ( l4irq_t ** irq )
```

Wait for any attached IRQ.

Return values

irq	Received interrupt.

Returns

0 on success, != 0 on error

9.99.2.8 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 I4_ipc_wait.

9.99.2.9 long l4irq_detach (l4irq_t *irq)

Detach from IRQ.

Parameters

irq	IRQ data structure

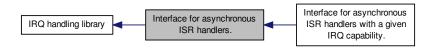
Returns

0 on success, != 0 on error

9.100 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 I4irq_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.

9.100.1 Detailed Description

This interface has just two (main) functions. I4irq_request to install a handler for an interrupt and I4irq_release to uninstall the handler again and release all resources associated with it.

9.100.2 Function Documentation

9.100.2.1 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.

9.100.2.2 long l4irq_release (l4irq_t * irq)

Release asynchronous ISR handler and free resources.

Parameters

irq IRQ data structure

Returns

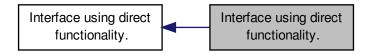
0 sucess, != 0 failure

Examples:

examples/libs/libirq/async_isr.c.

9.101 Interface using direct functionality.

Collaboration diagram for Interface using direct functionality.:



Functions

- I4irq_t * I4irq_attach_cap (I4_cap_idx_t irqcap)
 - Attach/connect to IRQ.
- I4irq_t * I4irq_attach_cap_ft (I4_cap_idx_t irqcap, unsigned mode)
 - Attach/connect to IRQ using given type.
- I4irq_t * I4irq_attach_thread_cap (I4_cap_idx_t irqcap, I4_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.
- 9.101.1 Detailed Description
- 9.101.2 Function Documentation
- 9.101.2.1 | I4irq_t* | I4irq_attach_cap (| I4_cap_idx_t irqcap)

Attach/connect to IRQ.

Parameters

irqcap	IRQ capability

Returns

Pointer to l4irq_t structure, 0 on error

This I4irq_attach has to be called in the same thread as I4irq_wait and caller has to be a pthread thread.

9.101.2.2 | I4irq_t* | I4irq_attach_cap_ft (| I4_cap_idx_t irqcap, unsigned mode)

Attach/connect to IRQ using given type.

Parameters

irqcap	IRQ capability

,	mode	Interrupt type.
	IIIUUE	ilitellupt type.

See Also

L4_irq_mode

Returns

Pointer to l4irq_t structure, 0 on error

This I4irq_attach has to be called in the same thread as I4irq_wait and caller has to be a pthread thread.

9.101.2.3 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.

9.101.2.4 l4irq_t* l4irq_attach_thread_cap_ft (I4_cap_idx_t irqcap, I4_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

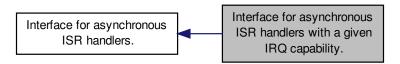
Returns

Pointer to l4irq_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

9.102 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.

9.102.1 Detailed Description

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

9.102.2 Function Documentation

9.102.2.1 | I4irq_t* | I4irq_request_cap (| I4_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

9.103 Sigma0 API 399

9.103 Sigma0 API

Sigma0 API bindings.

Collaboration diagram for Sigma0 API:



Modules

· Internal constants

Internal sigma0 definitions.

Files

• file sigma0.h

Sigma0 interface.

Enumerations

enum l4sigma0_return_flags_t {
 L4SIGMA0_OK, L4SIGMA0_NOTALIGNED, L4SIGMA0_IPCERROR, L4SIGMA0_NOFPAGE ,
 L4SIGMA0_SMALLERFPAGE }

Return flags of libsigma0 functions.

Functions

- I4_kernel_info_t * I4sigma0_map_kip (I4_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.

9.103.1 Detailed Description

Sigma0 API bindings. Convenience bindings for the Sigma0 protocol.

9.103.2 Enumeration Type Documentation

9.103.2.1 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.

 $\textbf{\textit{L4SIGMA0_NOFPAGE}} \quad \text{No fpage received}.$

L4SIGMA0_SMALLERFPAGE Superpage requested but smaller flexpage received.

Definition at line 81 of file sigma0.h.

9.103.3 Function Documentation

9.103.3.1 I4_kernel_info_t* I4sigma0_map_kip (I4_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.

9.103.3.2 int l4sigma0_map_mem (I4_cap_idx_t sigma0, I4_addr_t phys, I4_addr_t virt, I4_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()).

9.103 Sigma0 API 401

9.103.3.3 int l4sigma0_map_iomem (I4_cap_idx_t sigma0, I4_addr_t phys, I4_addr_t virt, I4_addr_t size, int cached)

Request IO memory from sigma0.

This function is similar to I4sigma0_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.

Parameters 4 8 1

sigma0	usually the thread id of sigma0.
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.

Returns

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

9.103.3.4 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 I4sigma0_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()).

9.103.3.5 int l4sigma0_map_tbuf (I4_cap_idx_t sigma0, I4_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()).

9.103.3.6 void l4sigma0_debug_dump (I4_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.
--------	-----------------------

9.103.3.7 int l4sigma0_new_client (I4_cap_idx_t sigma0, I4_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.

9.103.3.8 char const * I4sigma0_map_errstr(int err) [inline]

Get a user readable error messages for the return codes.

Parameters

<i>err</i> the error code reported by the <i>map</i> functions.	
---	--

Returns

a string containing the error message.

Definition at line 208 of file sigma0.h.

9.104 Internal constants 403

9.104 Internal constants

Internal sigma0 definitions.

Collaboration diagram for Internal constants:



Macros

#define SIGMA0_REQ_MAGIC ~0xFFUL

Request magic.

• #define SIGMA0_REQ_MASK \sim 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

KIF

#define SIGMA0_REQ_ID_TBUF 0xB0

TRUE

#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.

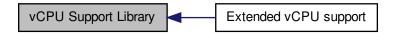
9.104.1 Detailed Description

Internal sigma0 definitions.

9.105 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.

Typedefs

typedef enum l4vcpu_irq_state_t l4vcpu_irq_state_t
 IRQ/Event enable and disable flags.

Enumerations

 enum l4vcpu_irq_state_t { L4VCPU_IRQ_STATE_DISABLED = 0, L4VCPU_IRQ_STATE_ENABLED = L4_-VCPU_F_IRQ }

IRQ/Event enable and disable flags.

Functions

I4vcpu_state_t I4vcpu_state (I4_vcpu_state_t const *vcpu) L4_NOTHROW

Return the state flags of a vCPU.

void l4vcpu_irq_disable (l4_vcpu_state_t *vcpu) L4_NOTHROW

Disable a vCPU for event delivery.

I4vcpu_irq_state_t I4vcpu_irq_disable_save (I4_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, l4vcpu_irq_state_t 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 (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 *vcpu) L4_NOTHROW

Return whether the entry reason was an IRQ/IPC message.

int l4vcpu_is_page_fault_entry (l4_vcpu_state_t *vcpu) L4_NOTHROW

Return whether the entry reason was a page fault.

9.105.1 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.

9.105.2 Enumeration Type Documentation

9.105.2.1 enum l4vcpu irq state t

IRQ/Event enable and disable flags.

Enumerator

L4VCPU_IRQ_STATE_DISABLED IRQ/Event delivery disabled. **L4VCPU_IRQ_STATE_ENABLED** IRQ/Event delivery enabled.

Definition at line 44 of file vcpu.h.

9.105.3 Function Documentation

9.105.3.1 | I4vcpu_state_t I4vcpu_state(I4_vcpu_state_t const * vcpu) [inline]

Return the state flags of a vCPU.

Parameters

```
vcpu Pointer to vCPU area.
```

Definition at line 229 of file vcpu.h.

Referenced by I4vcpu_irq_disable_save().

Here is the caller graph for this function:



9.105.3.2 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 236 of file vcpu.h.

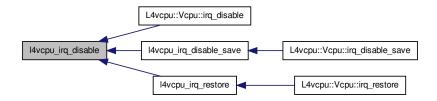
References I4_barrier(), and L4_VCPU_F_IRQ.

Referenced by L4vcpu::Vcpu::irq_disable(), l4vcpu_irq_disable_save(), and l4vcpu_irq_restore().

Here is the call graph for this function:



Here is the caller graph for this function:



Disable a vCPU for event delivery and return previous state.

Parameters

vcpu Pointer to vCPU area.

Returns

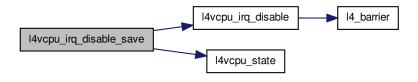
IRQ state before disabling IRQs.

Definition at line 244 of file vcpu.h.

References I4vcpu irg disable(), and I4vcpu state().

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

Here is the call graph for this function:



Here is the caller graph for this function:



9.105.3.4 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-	Call-back function that is called in case an event (such as an interrupt) is pending.
_cb	
setup_ipc	Function call-back that is called right before any IPC operation, and before event delivery is
	enabled.

Definition at line 267 of file vcpu.h.

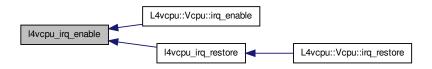
References EXPECT_TRUE, I4_barrier(), L4_IPC_BOTH_TIMEOUT_0, L4_VCPU_F_IRQ, and L4_VCPU_SF_IR-Q_PENDING.

Referenced by L4vcpu::Vcpu::irq_enable(), and I4vcpu_irq_restore().

Here is the call graph for this function:



Here is the caller graph for this function:



9.105.3.5 void l4vcpu_irq_restore (l4_vcpu_state_t * vcpu, l4vcpu_irq_state_t 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

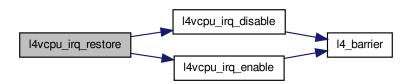
vcpu	Pointer to vCPU area.
S	IRQ state to be restored.
utcb	Utcb pointer of the calling vCPU.
do_event_work-	Call-back function that is called in case an event (such as an interrupt) is pending after en-
_cb	abling.
setup_ipc	Function call-back that is called right before any IPC operation, and before event delivery is
	enabled.

Definition at line 292 of file vcpu.h.

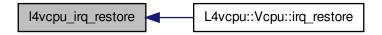
References L4_VCPU_F_IRQ, I4vcpu_irq_disable(), and I4vcpu_irq_enable().

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

Here is the call graph for this function:



Here is the caller graph for this function:



9.105.3.6 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-	Call-back function that is called when the vCPU awakes and needs to handle an event/IRQ.
_cb	
setup_ipc	Function call-back that is called right before any IPC operation.

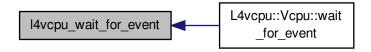
Note that event delivery remains disabled after this function returns.

Definition at line 305 of file vcpu.h.

References L4_IPC_NEVER.

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

Here is the caller graph for this function:



9.105.3.7 void l4vcpu_print_state (l4_vcpu_state_t * vcpu, const char * prefix)

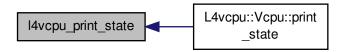
Print the state of a vCPU.

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:



9.105.3.8 int l4vcpu_is_irq_entry (l4_vcpu_state_t * vcpu) [inline]

Return whether the entry reason was an IRQ/IPC message.

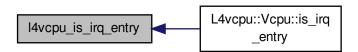
Parameters

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:



9.105.3.9 int l4vcpu_is_page_fault_entry (l4_vcpu_state_t * vcpu) [inline]

Return whether the entry reason was a page fault.

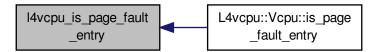
Parameters

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:



9.106 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.

9.106.1 Detailed Description

extended vCPU handling functionality.

9.106.2 Function Documentation

9.106.2.1 int l4vcpu_ext_alloc (l4_vcpu_state_t ** vcpu, l4_addr_t * ext_state, l4_cap_idx_t task, l4_cap_idx_t regmgr)

Allocate state area for an extended vCPU.

Return values

vcpu	Allocated vcpu-state area.
ext_state	Allocated extended vcpu-state area.

Parameters

task	Task to use for allocation.
regmgr	Region manager to use for allocation.

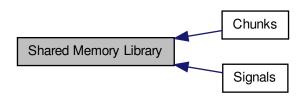
Returns

0 for success, error code otherwise

9.107 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.

9.107.1 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, reponsible 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).

9.107.2 Function Documentation

9.107.2.1 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.

Returns

0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.107.2.2 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.

Return values

shmarea	Pointer to shared memory area descriptor to be filled with information for the
	shared memory area.

Returns

0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.107.2.3 long l4shmc_attach_to (const char * shmc_name, I4_umword_t timeout_ms, I4shmc_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.

Return values

shmarea	Pointer to shared memory area descriptor to be filled with information for the
	shared memory area.

Returns

0 on success, <0 on error

9.107.2.4 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.

9.107.2.5 long l4shmc_area_size (l4shmc_area_t * shmarea) [inline]

Get size of shared memory area.

Parameters

shmarea Shared memory area.	
-----------------------------	--

Returns

<0 on error, otherwise: size of the shared memory area

9.107.2.6 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.
---------	---------------------

Returns

<0 on error, otherwise: free capacity in the area.

```
9.107.2.7 long l4shmc_area_overhead (void)
```

Get memory overhead per area that is not available for chunks.

Returns

size of the overhead in bytes

9.107.2.8 long l4shmc_chunk_overhead (void)

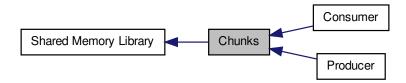
Get memory overhead required in addition to the chunk capacity for adding one chunk.

Returns

size of the overhead in bytes

9.108 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 * I4shmc_chunk_ptr (I4shmc_chunk_t *chunk)

Get data pointer to chunk.

• long l4shmc chunk capacity (l4shmc chunk t *chunk)

Get capacity of a chunk.

• I4shmc_signal_t * I4shmc_chunk_signal (I4shmc_chunk_t *chunk)

Get the signal of a chunk.

9.108.1 Detailed Description

9.108.2 Function Documentation

9.108.2.1 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.

9.108 Chunks 419

Parameters

sh	marea	The shared memory area to put the chunk in.
chunk	_name	Name of the chunk.
chunk_ca	apacity	Capacity for payload of the chunk in bytes.

Return values

chunk	Chunk structure to fill in.

Returns

0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.108.2.2 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.

Return values

chunk Chunk data structure to fill.	
---------------------------------------	--

Returns

0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.108.2.3 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.

Return values

chunk	chunk data structure to fill.

Returns

0 on success, <0 on error

9.108.2.4 long l4shmc_iterate_chunk (l4shmc_area_t * shmarea, const char ** chunk_name, long offs) Iterate over names of all existing chunks.

9.108 Chunks 421

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

Returns

<0 on error, 0 if no more chunks, >0 iterator value for next call

9.108.2.5 $void*I4shmc_chunk_ptr(I4shmc_chunk_t*\mathit{chunk})$ [inline]

Get data pointer to chunk.

Parameters

chunk	Chunk.

Returns

0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.108.2.6 long l4shmc_chunk_capacity (l4shmc_chunk_t * chunk) [inline]

Get capacity of a chunk.

Parameters

chunk	Chunk.

Returns

0 on success, <0 on error

 $\textbf{9.108.2.7} \quad \textbf{I4shmc_signal_t* I4shmc_chunk_signal (I4shmc_chunk_t* \textit{chunk})} \quad \texttt{[inline]}$

Get the signal of a chunk.

Parameters

chunk	Chunk.

Returns

0 if no signal has been register with this chunk, signal otherwise

9.109 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.

9.109.1 Detailed Description

9.109.2 Function Documentation

9.109.2.1 long l4shmc_chunk_try_to_take (l4shmc_chunk_t * chunk) [inline]

Try to mark chunk busy.

Parameters

```
chunk chunk to mark.
```

Returns

0 if chunk could be taken, <0 if not (try again then)

Examples:

examples/libs/shmc/prodcons.c.

9.109.2.2 long l4shmc_chunk_ready (l4shmc_chunk_t * chunk, l4_umword_t size) [inline]

Mark chunk as filled (ready).

Parameters

9.109 Producer 423

chunk	chunk.
size	Size of data in the chunk, in bytes.

Returns

0 on success, <0 on error

9.109.2.3 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.

Returns

0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.109.2.4 long l4shmc_is_chunk_clear (l4shmc_chunk_t * chunk) [inline]

Check whether chunk is free.

Parameters

chunk	Chunk to check.

Returns

0 on success, <0 on error

9.110 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.

9.110.1 Detailed Description

9.110.2 Function Documentation

9.110.2.1 long l4shmc_enable_chunk (l4shmc_chunk_t * chunk)

Enable a signal connected with a chunk.

Parameters

chunk | Chunk to enable.

Returns

0 on success, <0 on 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.

9.110.2.2 long l4shmc_wait_chunk(l4shmc_chunk_t * chunk) [inline]

Wait on a specific chunk.

9.110 Consumer 425

Parameters

chunk	Chunk to wait for.
-------	--------------------

Returns

0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.110.2.3 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.

Returns

0 on success, <0 on 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.

9.110.2.4 long l4shmc_wait_chunk_try (l4shmc_chunk_t * chunk) [inline]

Check whether a specific chunk has an event pending.

Parameters

chunk	Chunk to check.

Returns

0 on success, <0 on 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.

9.110.2.5 long l4shmc_chunk_consumed (l4shmc_chunk_t * chunk) [inline]

Mark a chunk as free.

Parameters

chunk	Chunk to mark as free.

Returns

0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.110.2.6 long l4shmc_is_chunk_ready (l4shmc_chunk_t * chunk) [inline]

Check whether data is available.

9.110 Consumer 427

Parameters

chunk	Chunk to check.
-------	-----------------

Returns

0 on success, <0 on error

9.110.2.7 long l4shmc_chunk_size (l4shmc_chunk_t * chunk) [inline]

Get current size of a chunk.

Parameters

chunk	Chunk.

Returns

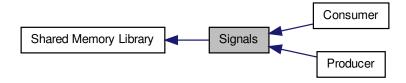
0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.111 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.

• I4_cap_idx_t I4shmc_signal_cap (I4shmc_signal_t *signal)

Get the signal capability of a signal.

• long l4shmc_check_magic (l4shmc_chunk_t *chunk)

Check magic value of a chunk.

9.111.1 Detailed Description

9.111.2 Function Documentation

9.111.2.1 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

9.111 Signals 429

shmarea	The shared memory area to put the chunk in.
signal_name	Name of the signal.

Return values

signal	Signal structure to fill in.

Returns

0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.111.2.2 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.

Return values

signal	Signal data structure to fill.
--------	--------------------------------

Returns

0 on success, <0 on error

9.111.2.3 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.

Return values

signal	Signal data structure to fill.

Returns

0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.111.2.4 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.

9.111 Signals 431

Parameters

9.111.2.5 I4_cap_idx_t I4shmc_signal_cap(I4shmc_signal_t * signal) [inline]

Get the signal capability of a signal.

Parameters

signal Signal.

Returns

Capability of the signal object.

9.111.2.6 long l4shmc_check_magic (l4shmc_chunk_t * chunk) [inline]

Check magic value of a chunk.

Parameters

chunk Chunk.

Returns

True if chunk is ok (magic value valid), false if not.

9.112 Producer

Collaboration diagram for Producer:



Functions

• long l4shmc_trigger (l4shmc_signal_t *signal)

Trigger a signal.

9.112.1 Detailed Description

9.112.2 Function Documentation

9.112.2.1 long l4shmc_trigger (l4shmc_signal_t * signal) [inline]

Trigger a signal.

Parameters

signal Signal to trigger.	
-----------------------------	--

Returns

0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.113 Consumer 433

9.113 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.

9.113.1 Detailed Description

9.113.2 Function Documentation

9.113.2.1 long l4shmc_enable_signal (l4shmc_signal_t * signal)

Enable a signal.

Parameters

```
signal | Signal to enable.
```

Returns

0 on success, <0 on 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.

9.113.2.2 long l4shmc_wait_any (l4shmc_signal_t ** retsignal) [inline]

Wait on any signal.

Return values

retsignal	Signal received.

Returns

0 on success, <0 on error

9.113.2.3 long l4shmc_wait_any_try (l4shmc_signal_t ** retsignal) [inline]

Check whether any waited signal has an event pending.

Return values

retsignal	Signal that has the event pending if any.

Returns

0 on success, <0 on 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.

9.113.2.4 long l4shmc_wait_any_to (I4_timeout_t timeout, l4shmc_signal_t ** retsignal)

Wait for any signal with timeout.

Parameters

timeout	Timeout.

Return values

retsignal	Signal that has the event pending if any.

Returns

0 on success, <0 on 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.

9.113.2.5 long l4shmc_wait_signal (l4shmc_signal_t * signal) [inline]

Wait on a specific signal.

Parameters

signal	Signal to wait for.

Returns

0 on success, <0 on error

Examples:

examples/libs/shmc/prodcons.c.

9.113 Consumer 435

9.113.2.6 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.

Returns

0 on success, <0 on error

9.113.2.7 long l4shmc_wait_signal_try (l4shmc_signal_t * signal) [inline]

Check whether a specific signal has an event pending.

Parameters

signal	Signal to check.

Returns

0 on success, <0 on 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.

9.114 Integer Types 437

9.114 Integer Types

#include<14/sys/14int.h>

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 I4_int8_t

Signed 8bit value.

• typedef unsigned char I4_uint8_t

Unsigned 8bit value.

• typedef signed short int I4_int16_t

Signed 16bit value.

• typedef unsigned short int I4_uint16_t

Unsigned 16bit value.

• typedef signed int I4_int32_t

Signed 32bit value.

• typedef unsigned int I4_uint32_t

Unsigned 32bit value.

typedef signed long long l4_int64_t

```
• typedef unsigned long long l4_uint64_t
          Unsigned 64bit value.
    • typedef unsigned long l4_addr_t
          Address type.
    • typedef signed long I4_mword_t
          Signed machine word.
    • typedef unsigned long I4_umword_t
          Unsigned machine word.
    • typedef I4_uint64_t I4_cpu_time_t
          CPU clock type.

    typedef l4_uint64_t l4_kernel_clock_t

          Kernel clock type.
    • typedef unsigned int I4_size_t
          Signed size type.
    • typedef signed int I4_ssize_t
          Unsigned size type.
    • typedef unsigned long I4_size_t
          Signed size type.
    • typedef signed long I4_ssize_t
          Unsigned size type.

    typedef unsigned int I4_size_t

          Signed size type.
    • typedef signed int I4_ssize_t
          Unsigned size type.
9.114.1 Detailed Description
#include<14/sys/14int.h>
9.114.2 Typedef Documentation
9.114.2.1 typedef signed char I4_int8_t
Signed 8bit value.
Definition at line 35 of file l4int.h.
9.114.2.2 typedef unsigned char I4_uint8_t
Unsigned 8bit value.
Definition at line 36 of file l4int.h.
9.114.2.3 typedef signed short int I4_int16_t
Signed 16bit value.
```

Definition at line 37 of file l4int.h.

Signed 64bit value.

9.114 Integer Types 439

9.114.2.4 typedef unsigned short int I4_uint16_t

Unsigned 16bit value.

Definition at line 38 of file l4int.h.

9.114.2.5 typedef signed int I4_int32_t

Signed 32bit value.

Definition at line 39 of file l4int.h.

9.114.2.6 typedef unsigned int I4_uint32_t

Unsigned 32bit value.

Definition at line 40 of file l4int.h.

9.114.2.7 typedef signed long long I4_int64_t

Signed 64bit value.

Definition at line 41 of file l4int.h.

9.114.2.8 typedef unsigned long long I4_uint64_t

Unsigned 64bit value.

Definition at line 42 of file l4int.h.

Chapter 10

Namespace Documentation

10.1 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 Tree for simple comapreable items.

class Avl_tree_node

Node of an AVL tree.

· class Avl_tree

A generic AVL tree.

class Bitfield

Definition for a member (part) of a bit field.

• class Bitmap_base

Basic bitmap abstraction.

class Bitmap

A static bit map.

· class List_item

Basic list item.

· class List

Doubly linked list, with internal allocation.

class List_alloc

Standard list-based allocator.

struct Pair

Pair of two values.

class Pair_first_compare

Comparison functor for Pair.

· class Base_slab

Basic slab allocator.

class Slab

Slab allocator for object of type Type.

• class Base_slab_static

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

· class Slab_static

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

class Nothrow

Helper type to distinguish the opprator new version that does not throw exceptions.

class New_allocator

Standard allocator based on operator new () .

struct Lt_functor

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

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)
```

10.1.1 Detailed Description

Our C++ library. Various kinds of C++ utilities.

Get the maximum of a and b.

10.2 cxx::Bits Namespace Reference

Internal helpers for the cxx package.

Data Structures

· class Bst

Basic binary search tree (BST).

• struct Direction

The direction to go in a binary search tree.

• class Bst_node

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

10.2.1 Detailed Description

Internal helpers for the cxx package.

10.3 L4 Namespace Reference

L4 low-level kernel interface.

Namespaces

lpc_svr

Helper classes for L4::Server instantiation.

Data Structures

· class Alloc list

A simple list-based allocator.

class IOModifier

Modifier class for the IO stream.

class Exception_tracer

Back-trace support for exceptions.

· class Base_exception

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

· class Runtime error

Exception for an abstract runtime error.

class Out_of_memory

Exception signalling insufficient memory.

class Element_already_exists

Exception for duplicate element insertions.

• class Unknown_error

Exception for an unknown condition.

· class Element not found

Exception for a failed lookup (element not found).

· class Invalid_capability

Indicates that an invalid object was invoked.

· class Com_error

Error conditions during IPC.

class Bounds_error

Access out of bounds.

· class Server

Basic server loop for handling client requests.

· class Server object

Abstract server object to be used with L4::Server and L4::Basic_registry.

class Basic_registry

This registry returns the corresponding server object based on the label of an <code>lpc_gate</code>.

class String

A null-terminated string container class.

struct Type_info

Dynamic Type Information for L4Re Interfaces.

· class Kobject t

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

class Kobject_2t

Helper class to create an L4Re interface class that is derived from two base classes.

class Vm

Virtual machine.

class Cap

Capability Selector a la C++.

class Cap_base

Base class for all kinds of capabilities.

· class Kobject

Base class for all kinds of kernel objects, referred to by capabilities.

· class Debugger

Debugger interface.

· class Factory

C++ L4 Factory, to create all kinds of kernel objects.

class lpc_gate

L4 IPC gate.

class Irq

C++ version of an L4 IRQ.

• class Icu

C++ version of an interrupt controller.

· class Meta

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

· class Scheduler

Scheduler object.

class Smart_cap

Smart capability class.

class Task

An L4 Task.

· class Thread

L4 kernel thread.

class Vcon

C++ L4 Vcon.

reinterpret_cast for capabilities.

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_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 >
  Cap< T > cap_dynamic_cast (Cap< F > const &c) throw ()
     dynamic_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 capabilities.
• template<typename T , typename F , typename SMART >
  Smart_cap < T, SMART > cap_reinterpret_cast (Smart_cap < F, SMART > const &c) throw ()
```

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.

10.3.1 Detailed Description

L4 low-level kernel interface.

10.3.2 Function Documentation

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

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

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 87 of file __typeinfo.h.

10.4 L4::lpc_svr Namespace Reference

Helper classes for L4::Server instantiation.

Data Structures

struct Ignore_errors

Mix in for LOOP_HOOKS to ignore IPC errors.

struct Default_timeout

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

struct Compound_reply

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

struct Default_setup_wait

Mix in for LOOP_HOOKS for setup_wait no op.

struct Default_loop_hooks

Default LOOP_HOOKS.

Enumerations

• enum Reply_mode { Reply_compound, Reply_separate }

Reply mode for server loop.

10.4.1 Detailed Description

Helper classes for L4::Server instantiation.

10.4.2 Enumeration Type Documentation

10.4.2.1 enum L4::lpc_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_compund), 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 52 of file ipc_server.

10.5 L4Re Namespace Reference

L4 Runtime Environment.

Namespaces

Vfs

Virtual file system for interfaces POSIX libc.

Data Structures

class Cap_alloc

Capability allocator interface.

· class Smart_cap_auto

Helper for Auto_cap and Auto_del_cap.

· class Console

Console class.

class Dataspace

This class represents a data space.

· class Debug_obj

Debug interface.

class Env

Initial Environment (C++ version).

class Event

Event class.

class Event_buffer_t

Event buffer class.

class Log

Log interface class.

· class Mem_alloc

Memory allocator.

class Namespace

Name-space interface.

· class Parent

Parent interface.

class Rm

Region map.

10.5.1 Detailed Description

L4 Runtime Environment.

10.6 L4Re::Vfs Namespace Reference

Virtual file system for interfaces POSIX libc.

Data Structures

· class Be_file

Boiler plate class for implementing an open file for L4Re::Vfs.

class Be_file_system

Boilerplate class for implementing a L4Re::Vfs::File_system.

· class Generic_file

The common interface for an open POSIX file.

class Directory

Interface for a POSIX file that is a directory.

class Regular_file

Interface for a POSIX file that provides regular file semantics.

class Special_file

Interface for a POSIX file that provides special file semantics.

class File

The basic interface for an open POSIX file.

• class Mman

Interface for the POSIX memory management.

class File_system

Basic interface for an L4Re::Vfs file system.

class Fs

POSIX File-system related functionality.

class Ops

Interface for the POSIX backends for an application.

Functions

L4Re::Vfs::Ops *vfs_ops asm ("l4re_env_posix_vfs_ops")
 Reference to the applications L4Re::Vfs::Ops singleton.

10.6.1 Detailed Description

Virtual file system for interfaces POSIX libc.

Chapter 11

Data Structure Documentation

11.1 L4::Alloc_list Class Reference

A simple list-based allocator.

#include <alloc.h>

Collaboration diagram for L4::Alloc_list:



11.1.1 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:

• I4/cxx/alloc.h

11.2 L4::Thread::Attr Class Reference

Thread attributes used for control_commit().

Collaboration diagram for L4::Thread::Attr:

+ Attr() + pager() + pager() + exc_handler() + bind() + alien() + ux_host_syscall()

Public Member Functions

```
• Attr (I4_utcb_t *utcb=I4_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 (I4_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

11.2.1 Detailed Description

Thread attributes used for control_commit().

This class is responsible for initalizing various attributes of a thread in a UTCB for the control_commit() method.

See Also

Thread control for some more details.

Definition at line 70 of file thread.

11.2.2 Constructor & Destructor Documentation

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 82 of file thread.

11.2.3 Member Function Documentation

11.2.3.1 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 91 of file thread.

References pager().

Here is the call graph for this function:



11.2.3.2 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 99 of file thread.

References I4_msg_regs_t::mr.

Referenced by pager().

Here is the caller graph for this function:



11.2.3.3 void L4::Thread::Attr::exc_handler (Cap < void > const & exc_handler) throw) [inline]

Set the exception-handler capability selector.

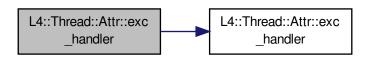
Parameters

pager	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 108 of file thread.

References exc_handler().

Here is the call graph for this function:



11.2.3.4 Cap<void>L4::Thread::Attr::exc_handler()throw) [inline]

Get the capability selector used for exception messages.

Returns

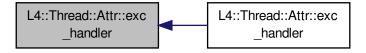
the capability selector used to send exception messages. The selector is valid in the task the thread is bound to.

Definition at line 116 of file thread.

References I4_msg_regs_t::mr.

Referenced by exc_handler().

Here is the caller graph for this function:



11.2.3.5 void L4::Thread::Attr::bind (I4_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 130 of file thread.

11.2.3.6 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 143 of file thread.

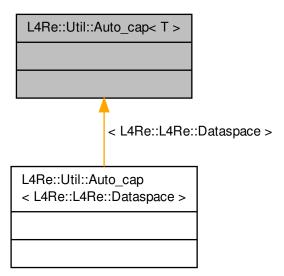
The documentation for this class was generated from the following file:

· I4/sys/thread

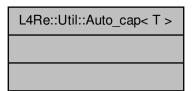
11.3 L4Re::Util::Auto_cap< T > Struct Template Reference

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

Inheritance diagram for L4Re::Util::Auto_cap< T >:



Collaboration diagram for L4Re::Util::Auto_cap< T >:



11.3.1 Detailed Description

template<typename T>struct L4Re::Util::Auto_cap< T>

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

Parameters

```
T \mid the type of the object that is referred by the capability.
```

This kind of automatic capability is useful for capabilities with 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::Datasapce>));

// 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.

// At the end of the scope ds_cap is unmapped and the capability selector
```

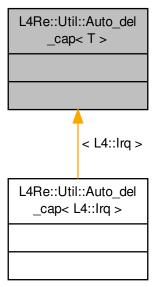
Definition at line 159 of file cap_alloc.

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

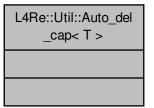
· I4/re/util/cap_alloc

11.4 L4Re::Util::Auto_del_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector. Inheritance diagram for L4Re::Util::Auto_del_cap< T >:



Collaboration diagram for L4Re::Util::Auto_del_cap< T >:



11.4.1 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.

Parameters

```
T the type of the object that is referred by the capability.
```

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::Datasapce>));

* // 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 193 of file cap_alloc.

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

I4/re/util/cap_alloc

11.5 cxx::Auto_ptr< T > Class Template Reference

Smart pointer with automatic deletion.

Collaboration diagram for cxx::Auto_ptr< T >:

cxx::Auto_ptr< T > + Auto_ptr() + Auto_ptr() + operator=() + ~Auto_ptr() + operator*() + operator->() + get() + release() + reset() + operator Priv_type *()

Public Types

typedef T Ref_type

Public Member Functions

```
• Auto_ptr (T *p=0) throw ()
```

The referenced type.

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) ...< >.

11.5.1 Detailed Description

template<typename T>class cxx::Auto_ptr< T>

Smart pointer with automatic deletion.

Parameters

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.

11.5.2 Member Typedef Documentation

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

The referenced type.

Definition at line 41 of file auto_ptr.

11.5.3 Constructor & Destructor Documentation

```
11.5.3.1 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.

```
11.5.3.2 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.

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

Destruction, shall delete the object.

Definition at line 76 of file auto_ptr.

11.5.4 Member Function Documentation

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

Assignment from another smart pointer.

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:



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

Dereference the pointer.

Definition at line 80 of file auto_ptr.

 $\textbf{11.5.4.3} \quad \textbf{template} < \textbf{typename} \ \textbf{T} > \textbf{T} * \textbf{cxx::Auto_ptr} < \textbf{T} > \textbf{::operator} > \textbf{() const throw)} \quad \texttt{[inline]}$

Member access for the object.

Definition at line 83 of file auto_ptr.

11.5.4.4 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.

11.5.4.5 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:



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

Operator for if (!ptr) $\ldots < >$.

Definition at line 110 of file auto_ptr.

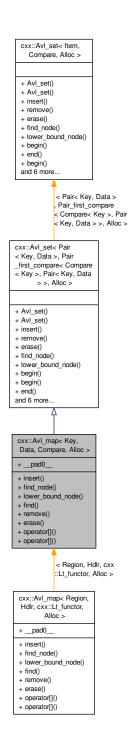
The documentation for this class was generated from the following file:

• I4/cxx/auto_ptr

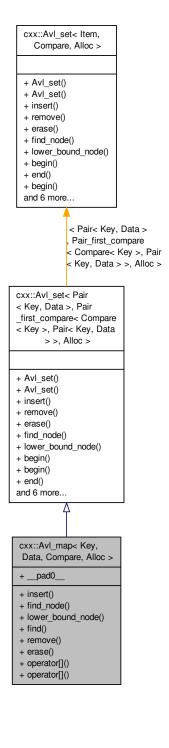
11.6 cxx::Avl_map< Key, Data, Compare, Alloc > Class Template Reference

AVL tree based associative container.

Inheritance diagram for cxx::Avl_map< Key, Data, Compare, Alloc >:



Collaboration diagram for cxx::Avl_map< Key, Data, Compare, Alloc >:



Public Types

- typedef Compare < Key > Key_compare
 Type of the comparison functor.
- typedef Key Key_type

Type of the key values.

typedef Data 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

• Node find_node (Key_type const &key) const

Find a < key, data> pair for a given key.

Node lower_bound_node (Key_type const &key) const

Find the first node greater or equal to key.

Iterator find (Key_type const &key) const

Find a < key, data> pair for a given key.

• int remove (Key_type const &key)

Remove the <key, data> pair for the given key.

int erase (Key_type const &key)

Removed the element key.

Data_type const & operator[] (Key_type const &key) const

Get the data for the given key.

Data_type & operator[] (Key_type const &key)

Get the data for the given key.

Data Fields

• __pad0__: Base_type(alloc) {} cxx::Pair<Iterator

Create an empty AVL-tree based map.

11.6.1 Detailed Description

 $template < typename \ A > class \ Compare = Lt_functor, template < typename \ B > class \ Alloc = New_allocator > class \ cxx::Avl_map < Key, Data, Compare, Alloc >$

AVL tree based associative container.

Parameters

Key	Type of the key values.
Data	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 44 of file avl_map.

11.6.2 Member Function Documentation

11.6.2.1 template < typename Key, typename Data, template < typename A > class Compare = Lt_functor, template < typename B > class Alloc = New_allocator > Node cxx::Avl_map < Key, Data, Compare, Alloc >::find_node (Key_type const & key) const [inline]

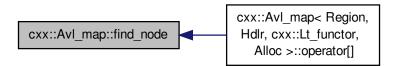
Find a <key, data> pair for a given key.

key	The key value to use for the lookup.

Definition at line 95 of file avl_map.

Referenced by cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc >::operator[]().

Here is the caller graph for this function:



Find the first node greater or equal to key.

Parameters

key	the key to look for.

Returns

The first node greater or equal to key.

Definition at line 103 of file avl_map.

11.6.2.3 template < typename Key, typename Data, template < typename A > class Compare = Lt_functor, template < typename B > class Alloc = New_allocator > Iterator cxx::AvI_map < Key, Data, Compare, Alloc >::find (Key_type const & key) const [inline]

Find a <key, data> pair for a given key.

Parameters

key	The key value to use for the lookup.

Definition at line 110 of file avl map.

11.6.2.4 template < typename Key, typename Data, template < typename A > class Compare = Lt_functor, template < typename B > class Alloc = New_allocator > int cxx::AvI_map < Key, Data, Compare, Alloc >::remove (Key_type const & key) [inline]

Remove the <key, data> pair for the given key.

key The key value of the pair that shall be removed.

Definition at line 117 of file avl_map.

11.6.2.5 template < typename Key, typename Data, template < typename A > class Compare = Lt_functor, template < typename B > class Alloc = New_allocator > int cxx::Avl_map < Key, Data, Compare, Alloc >::erase (Key_type const & key) [inline]

Removed the element key.

See Also

remove()

Definition at line 124 of file avl map.

11.6.2.6 template < typename Key, typename Data, template < typename A > class Compare = Lt_functor, template < typename B > class Alloc = New_allocator > Data_type const& cxx::Avl_map < Key, Data, 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 132 of file avl_map.

11.6.2.7 template < typename Key, typename Data, template < typename A > class Compare = Lt_functor, template < typename B > class Alloc = New_allocator > Data_type& cxx::Avl_map < Key, Data, Compare, Alloc >::operator[] (
 Key type const & key) [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 140 of file avl_map.

11.6.3 Field Documentation

11.6.3.1 template < typename Key, typename Data, template < typename A > class Compare = Lt_functor, template < typename B > class Alloc = New_allocator > cxx::AvI map < Key, Data, Compare, Alloc >::_pad0_

Create an empty AVL-tree based map.

comp	The comparison functor.
alloc	The node allocator.

Definition at line 88 of file avl_map.

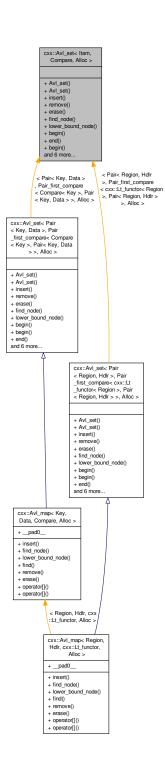
The documentation for this class was generated from the following file:

• I4/cxx/avl_map

11.7 cxx::Avl_set < Item, Compare, Alloc > Class Template Reference

AVL Tree for simple comapreable items.

Inheritance diagram for cxx::Avl_set< Item, Compare, Alloc >:



Collaboration diagram for cxx::Avl_set< Item, Compare, Alloc >:

cxx::Avl_set< Item,
 Compare, Alloc >

+ Avl_set()
+ Avl_set()
+ insert()
+ remove()
+ erase()
+ find_node()
+ lower_bound_node()
+ begin()
+ begin()

and 6 more...

Data Structures

· class Node

A smart pointer to a tree item.

Public Types

typedef Item Item_type

Type for the items contained in the tree.

• typedef Compare Item_compare

Type for the comparison functor.

typedef Alloc
 Node > Node allocator

Type for the node allocator.

 typedef __Avl_set_iter< _Node, ltem_type, Fwd > lterator

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

Avl_set (Node_allocator const &alloc=Node_allocator())

Create a AVL-tree based set.

Avl_set (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 (Item_type const &item)

Remove an item from the set.

Node find node (Item type const &item) const

Lookup a node equal to item.

Node lower_bound_node (Item_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.

11.7.1 Detailed Description

 $template < typename \ A > class \ Alloc = New_allocator > class \\ cxx::Avl_set < Item, Compare, Alloc >$

AVL Tree for simple comapreable items.

The AVL tree can store any kind of items where a partial order is defined. The default relation is defined by the '<' operator.

Parameters

Item	The type of the items to be stored in the tree.
Compare	The relation to define the partial order, default is to use operator '<'.
Alloc	The allocator to use for the nodes of the AVL tree.

Definition at line 106 of file avl_set.

11.7.2 Constructor & Destructor Documentation

11.7.2.1 template<typename ltem, class Compare = Lt_functor<ltem>, template< typename A > class Alloc = New_allocator> cxx::Avl_set< ltem, Compare, Alloc >::Avl_set(Node_allocator const & alloc = Node allocator()) [inline], [explicit]

Create a AVL-tree based set.

comp	Comparison functor.
alloc	Node allocator.

Create an empty set (AVL-tree based).

Definition at line 215 of file avl_set.

11.7.2.2 template < typename ltem , class Compare , template < typename A > class Alloc > cxx::AvI_set < ltem, Compare, Alloc > :::AvI_set (AvI_set < ltem, Compare, Alloc > const & o) [inline]

Create a copy of an AVL-tree based set.

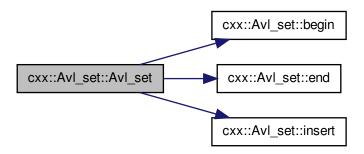
Parameters

0	The set to copy.

Definition at line 335 of file avl_set.

References cxx::Avl_set< Item, Compare, Alloc >::begin(), cxx::Avl_set< Item, Compare, Alloc >::end(), and cxx::Avl_set< Item, Compare, Alloc >::insert().

Here is the call graph for this function:



11.7.3 Member Function Documentation

11.7.3.1 template < typename Item , class Compare , template < typename A > class Alloc > Pair < typename AvI_set < Item, Compare, Alloc >::Iterator, int > cxx::AvI_set < Item, Compare, Alloc >::insert (Item_type const & item)

Insert an item into the set.

Parameters

item	The item to insert.

Returns

0 on success, -1 on out of memory, and -2 if the element already exists in the set.

Insert a new item into the set, each item can only be once in the set.

Definition at line 345 of file avl_set.

References cxx::Pair< First, Second >::first, and cxx::Pair< First, Second >::second.

Referenced by cxx::Avl_set< Item, Compare, Alloc >::Avl_set().

Here is the caller graph for this function:



11.7.3.2 template<typename Item, class Compare = Lt_functor<Item>, template< typename A > class Alloc =

New_allocator> int cxx::AvI_set< Item, Compare, Alloc >::remove (Item_type const & item) [inline]

Remove an item from the set.

Parameters

ſ	item	The item to remove.

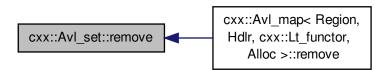
Returns

0 on success, -3 if the item does not exist, and -4 on internal error.

Definition at line 242 of file avl_set.

Referenced by cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc >::remove().

Here is the caller graph for this function:



11.7.3.3 template<typename ltem, class Compare = Lt_functor<ltem>, template< typename A > class Alloc = New_allocator> Node cxx::AvI_set< ltem, Compare, Alloc >::find_node (Item_type const & item) const [inline]

Lookup a node equal to item.

Parameters

item	The value to search for.

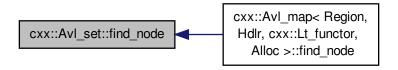
Returns

A smart pointer to the element found, if no element found the pointer is NULL.

Definition at line 267 of file avl_set.

Referenced by cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc >::find_node().

Here is the caller graph for this function:



11.7.3.4 template<typename ltem, class Compare = Lt_functor<ltem>, template< typename A > class Alloc = New_allocator> Node cxx::AvI_set< ltem, Compare, Alloc >::lower_bound_node (ltem_type const & key) const [inline]

Find the first node greater or equal to key.

Parameters

key the key to look for.	
--------------------------	--

Returns

The first node greater or equal to key.

Definition at line 275 of file avl set.

Referenced by cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc >::lower_bound_node().

Here is the caller graph for this function:



11.7.3.5 template<typename Item, class Compare = Lt_functor<Item>, template< typename A > class Alloc =

New_allocator> Const_iterator cxx::Avl_set< Item, Compare, Alloc >::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 283 of file avl set.

Referenced by cxx::Avl_set< Item, Compare, Alloc >::Avl_set().

Here is the caller graph for this function:



11.7.3.6 template<typename Item, class Compare = Lt_functor<Item>, template< typename A > class Alloc = New_allocator> Const_iterator cxx::Avl_set< Item, Compare, Alloc >::end () const [inline]

Get the end marker for the constant forward iterator.

Returns

The end marker for the constant forward iterator.

Definition at line 288 of file avl_set.

Referenced by cxx::Avl set< Item, Compare, Alloc >::Avl set().

Here is the caller graph for this function:



11.7.3.7 template<typename Item, class Compare = Lt_functor<Item>, template< typename A > class Alloc = New_allocator> Iterator cxx::AvI set< Item, Compare, Alloc >::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 294 of file avl_set.

```
11.7.3.8 template<typename Item, class Compare = Lt_functor<Item>, template< typename A > class Alloc = New_allocator> Iterator cxx::AvI_set< Item, Compare, Alloc >::end ( ) [inline]
```

Get the end marker for the mutable forward iterator.

Returns

The end marker for mutable forward iterator.

Definition at line 299 of file avl set.

```
11.7.3.9 template<typename Item, class Compare = Lt_functor<Item>, template< typename A > class Alloc =

New_allocator> Const_rev_iterator cxx::Avl_set< Item, Compare, Alloc >::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 305 of file avl set.

```
11.7.3.10 template<typename Item, class Compare = Lt_functor<Item>, template< typename A > class Alloc = New_allocator> Const rev iterator cxx::AvI set< Item, Compare, Alloc >::rend() const [inline]
```

Get the end marker for the constant backward iterator.

Returns

The end marker for the constant backward iterator.

Definition at line 310 of file avl_set.

```
11.7.3.11 template<typename Item, class Compare = Lt_functor<Item>, template< typename A > class Alloc = New_allocator> Rev_iterator cxx::Avl_set< Item, Compare, Alloc >::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 316 of file avl set.

```
11.7.3.12 template<typename Item, class Compare = Lt_functor<Item>, template< typename A > class Alloc = New_allocator> Rev_iterator cxx::Avl_set< Item, Compare, Alloc >::rend ( ) [inline]
```

Get the end marker for the mutable backward iterator.

Returns

The end marker for mutable backward iterator.

Definition at line 321 of file avl set.

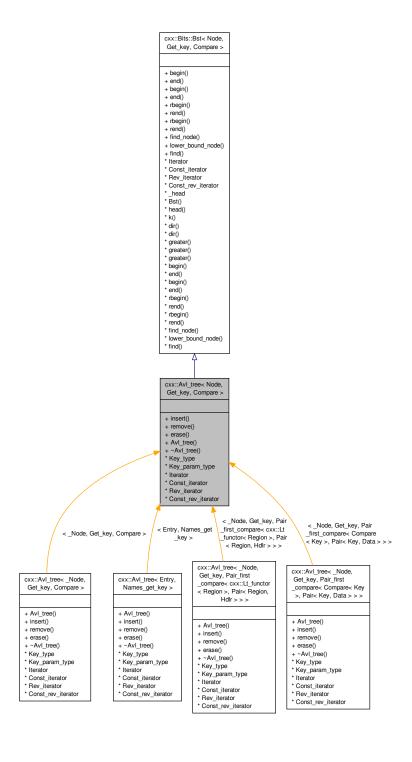
The documentation for this class was generated from the following file:

· I4/cxx/avl_set

11.8 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

Grab iterator types from Bst.

• typedef Bst::Const_iterator Const_iterator

Constant forward iterator for the set.

• typedef Bst::Rev_iterator Rev_iterator

Backward iterator for the set.

typedef Bst::Const_rev_iterator Const_rev_iterator

Constant backward iterator for the set.

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.

∼AvI tree ()

Destroy, and free the set.

11.8.1 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.

Parameters

Node	the data type of the nodes (must inherit from Avl_tree_noed).
Get_key	the meta fcuntion to get the key value from a node. The implementation uses Get_key::key-
	_of(ptr_to_node).
Compare	binary relation to establish a total order for the nodes of the tree. Compare()(I, r) must return
	true if the key / is smaller than the key r.

Examples:

tmpfs/lib/src/fs.cc.

Definition at line 102 of file avl_tree.

11.8.2 Member Typedef Documentation

11.8.2.1 template < typename Node, typename Get_key, typename Compare = Lt_functor < typename Get_key::Key_type >> typedef Bst::Iterator cxx::Avl_tree < Node, Get_key, Compare > ::Iterator

Grab iterator types from Bst.

Forward iterator for the set.

Definition at line 133 of file avl_tree.

11.8.3 Member Function Documentation

11.8.3.1 template < typename Node, typename Get_key , class Compare > Pair < Node *, bool > cxx::AvI_tree < Node, Get_key, Compare >::insert (Node * new_node)

Insert a new node into this AVL tree.

new_node	a pointer to the new node. This node must not already b in an AVL tree.	
----------	---	--

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 that first point to this node and second is 'false'.

Definition at line 225 of file avl_tree.

11.8.3.2 template<typename Node , typename Get_key , class Compare > Node * cxx::AvI_tree < Node, Get_key, Compare >::remove (Key_param_type key) [inline]

Remove the node with key from the tree.

Parameters

key The node to remove.		
	kev	The node to remove.

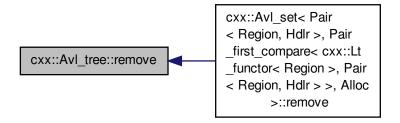
Returns

The pointer to the removed node on success, or NULL -3 if no node with the key exists.

Definition at line 287 of file avl_tree.

Referenced by cxx::Avl_set< Pair< Region, Hdlr >, Pair_first_compare< cxx::Lt_functor< Region >, Pair< Region, Hdlr > >, Alloc >::remove().

Here is the caller graph for this function:



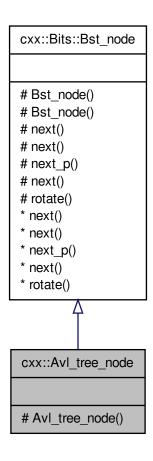
The documentation for this class was generated from the following file:

· I4/cxx/avl_tree

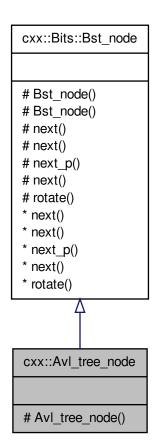
11.9 cxx::Avl_tree_node Class Reference

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 shoulkd do.

Additional Inherited Members

11.9.1 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:

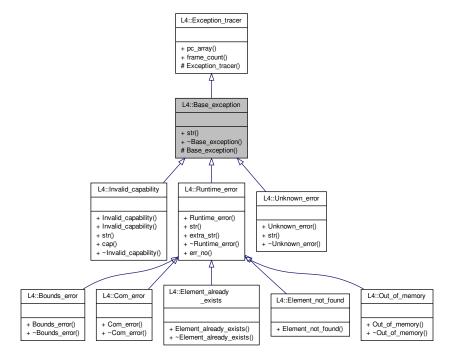
• I4/cxx/avl_tree

11.10 L4::Base_exception Class Reference

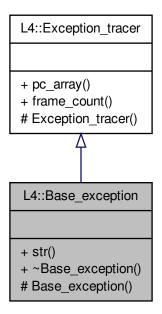
Base class for all exceptions, thrown by the L4Re framework.

#include <14/cxx/exceptions>

Inheritance diagram for L4::Base_exception:



Collaboration diagram for L4::Base_exception:



Public Member Functions

- virtual char const * str () const =0 throw ()
 - Should return a human readable string for the exception.
- virtual ∼Base_exception () throw ()

Destruction.

Protected Member Functions

• Base_exception () throw ()

Create a base exception.

11.10.1 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 117 of file exceptions.

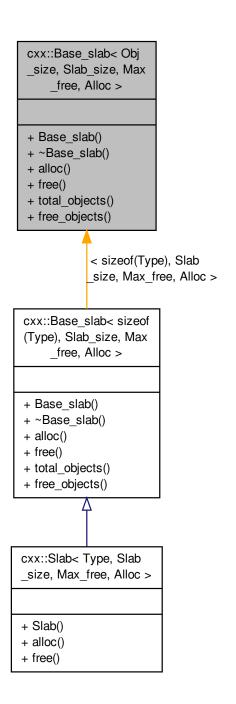
The documentation for this class was generated from the following file:

· I4/cxx/exceptions

11.11 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 >:

cxx::Base_slab< Obj _size, Slab_size, Max _free, Alloc > + Base_slab() + ~Base_slab() + alloc() + free() + total_objects() + free_objects()

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.

11.11.1 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.

11.11.2 Member Enumeration Documentation

11.11.2.1 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.

11.11.3 Member Function Documentation

11.11.3.1 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.

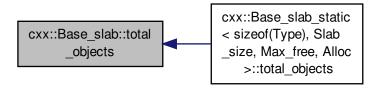
Returns

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:



11.11.3.2 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.

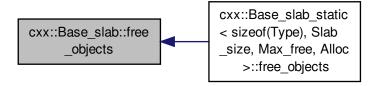
Returns

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:



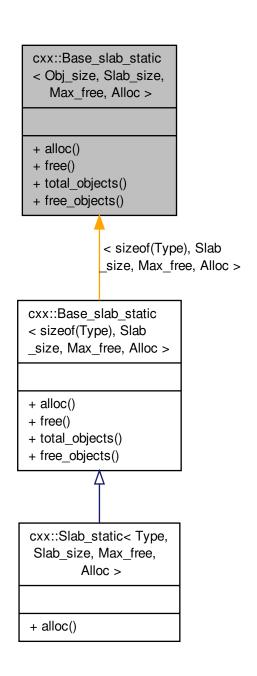
The documentation for this class was generated from the following file:

• I4/cxx/slab_alloc

11.12 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 >:

```
cxx::Base_slab_static
< Obj_size, Slab_size,
    Max_free, Alloc >

+ alloc()
+ free()
+ total_objects()
+ free_objects()
```

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.

11.12.1 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.

11.12.2 Member Enumeration Documentation

11.12.2.1 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.

11.12.3 Member Function Documentation

11.12.3.1 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.

11.12.3.2 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.

```
11.12.3.3 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.

```
11.12.3.4 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.

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

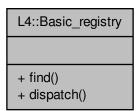
• I4/cxx/slab_alloc

11.13 L4::Basic_registry Class Reference

This registry returns the corresponding server object based on the label of an lpc gate.

Inherited by L4Re::Util::Object_registry.

Collaboration diagram for L4::Basic_registry:



Public Types

• typedef Server_object Value

Get the server object for a lpc_gate label.

Static Public Member Functions

• static int dispatch (I4_umword_t label, L4::Ipc::lostream &ios)

The dispatch function called by the server loop.

11.13.1 Detailed Description

This registry returns the corresponding server object based on the label of an lpc_gate.

Definition at line 319 of file ipc_server.

11.13.2 Member Typedef Documentation

11.13.2.1 typedef Server_object L4::Basic_registry::Value

Get the server object for a lpc_gate label.

Parameters

label	The label usually stored in an lpc_gate.

Returns

A pointer to the Server_object identified the given label.

Definition at line 327 of file ipc_server.

11.13.3 Member Function Documentation

11.13.3.1 static int L4::Basic_registry::dispatch (I4_umword_t label, L4::lpc::lostream & ios) [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

label	The label used to find the object including the rights bits of the invoked capability.
ios	The lpc::lostream for the request and the reply.

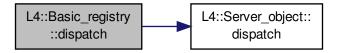
Returns

The return code from the object's dispatch function or -L4_ENOENT if the object does not exist.

Definition at line 343 of file ipc_server.

References L4::Server_object::dispatch(), and L4_ENOENT.

Here is the call graph for this function:



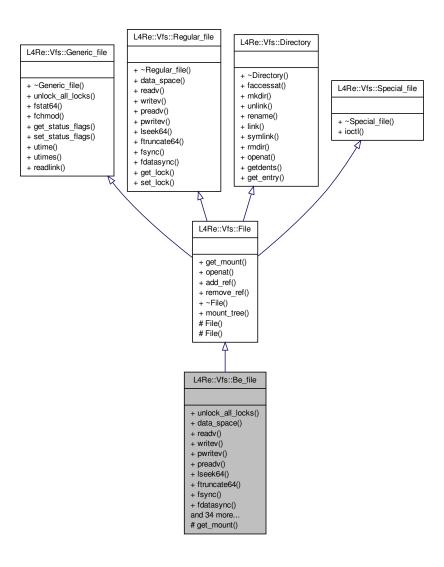
The documentation for this class was generated from the following file:

· I4/cxx/ipc_server

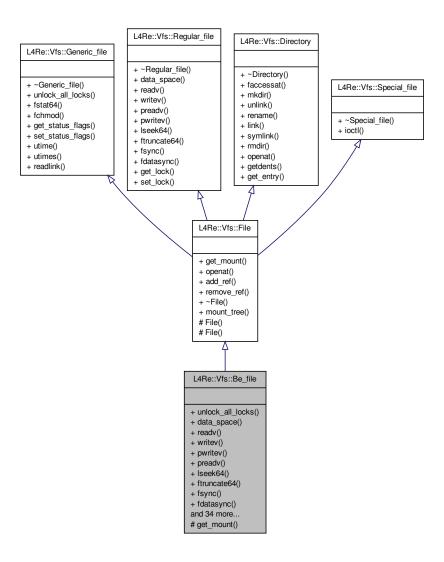
11.14 L4Re::Vfs::Be_file Class Reference

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 pready functions.

• off64_t lseek64 (off64_t, int) throw ()

Default backend for POSIX seek and Iseek 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.

11.14.1 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 39 of file backend.

11.14.2 Member Function Documentation

```
11.14.2.1 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 43 of file backend.

```
11.14.2.2 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 functions returns an invalid capability.

Returns

A capability to an L4Re::Dataspace, that represents the files contents in an L4Re way.

Implements L4Re::Vfs::Regular_file.

Definition at line 47 of file backend.

```
11.14.2.3 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.

Return values

```
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 86 of file backend.

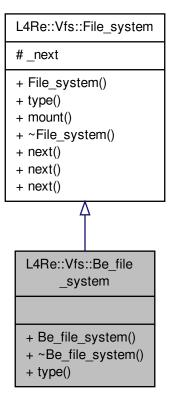
The documentation for this class was generated from the following file:

· I4/I4re_vfs/backend

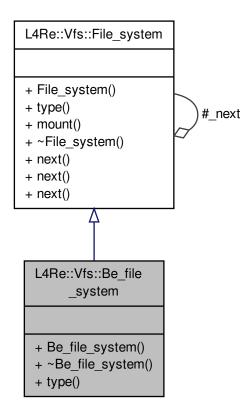
11.15 L4Re::Vfs::Be_file_system Class Reference

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.
- \sim Be_file_system () throw ()

Destroy a file-system object.

• char const * type () const throw ()

Return the file-system type.

11.15.1 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 290 of file backend.

11.15.2 Constructor & Destructor Documentation

```
11.15.2.1 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 304 of file backend.

```
11.15.2.2 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 316 of file backend.

11.15.3 Member Function Documentation

```
11.15.3.1 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 326 of file backend.

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

• I4/I4re_vfs/backend

11.16 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 >:

+ val unshifted()

Data Structures

· class Value

Internal helper type.

class Value_base

Internal helper type.

· class Value_unshifted

Internal helper type.

Public Types

- 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 .

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.

11.16.1 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

Т	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.

11.16.2 Member Typedef Documentation

11.16.2.1 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.

11.16.2.2 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.

11.16.2.3 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.

11.16.2.4 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.

 $11.16.2.5 \quad template < typename \ T\ , unsigned \ LSB, unsigned \ MSB > typedef \ Value_unshifted < T\&> cxx::Bitfield < T, LSB, \\ MSB > ::Ref_unshifted$

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

Definition at line 223 of file bitfield.

11.16.2.6 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.

11.16.3 Member Enumeration Documentation

11.16.3.1 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.

11.16.3.2 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.

11.16.4 Member Function Documentation

11.16.4.1 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.

 $References\ cxx::Bitfield < T,\ LSB,\ MSB > ::Lsb,\ and\ cxx::Bitfield < T,\ LSB,\ and\ cxx::Bitfield$

Here is the call graph for this function:



Get the bits in place out of .

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.

11.16.4.3 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

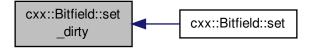
This function does not mask val to the right number of bits.

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:



11.16.4.4 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.

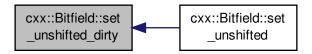
References cxx::Bitfield< T, LSB, MSB >::Mask, and cxx::Bitfield< T, LSB, MSB >::val().

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:



11.16.4.5 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, and cxx::Bitfield < T, LSB, MSB >::set_dirty().

Here is the call graph for this function:



11.16.4.6 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.

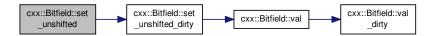
Returns

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:



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

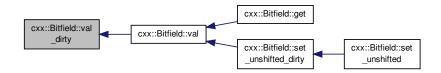
This function does not mask val to the right number of bits.

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:



11.16.4.8 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

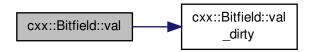
the raw bit field value containing.

Definition at line 170 of file bitfield.

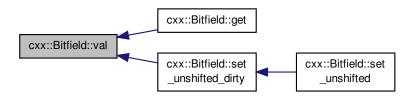
References cxx::Bitfield < T, LSB, MSB >::Low_mask, and cxx::Bitfield < T, LSB, MSB >::val_dirty().

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



11.16.4.9 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.

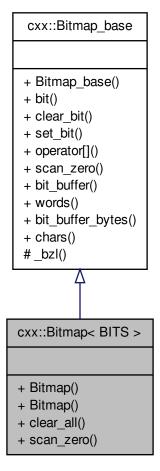
The documentation for this class was generated from the following file:

• I4/cxx/bitfield

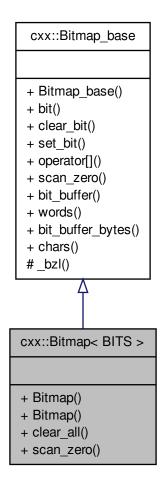
11.17 cxx::Bitmap < BITS > Class Template Reference

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.

• void clear_all ()

Scan for the first zero bit.

Additional Inherited Members

11.17.1 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 120 of file bitmap.

11.17.2 Constructor & Destructor Documentation

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

Create a bitmap with BITS bits.

Definition at line 127 of file bitmap.

11.17.3 Member Function Documentation

```
11.17.3.1 template<int BITS> void cxx::Bitmap< BITS>::clear_all( ) [inline]
```

Scan for the first zero bit.

Parameters

start_bit the bit where the scanning shall begin.

Compared to Bitmap_base::scan_zero(), the upper bound is set to BITS.

Definition at line 137 of file bitmap.

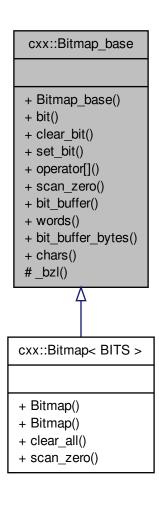
The documentation for this class was generated from the following file:

• I4/cxx/bitmap

11.18 cxx::Bitmap_base Class Reference

Basic bitmap abstraction.

Inheritance diagram for cxx::Bitmap_base:



Collaboration diagram for cxx::Bitmap_base:

cxx::Bitmap_base

- + Bitmap_base()
- + bit()
- + clear_bit()
- + set_bit()
- + operator[]()
- + scan_zero()
- + bit_buffer()
- + words()
- + bit_buffer_bytes()
- + chars()
- # _bzl()

Data Structures

· 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.

• unsigned long operator[] (long bit) const throw ()

Get the truth value of a bit.

• long scan_zero (long max_bit, long start_bit=0) const throw ()

Scans 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.

11.18.1 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.

11.18.2 Member Function Documentation

```
11.18.2.1 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 44 of file bitmap.

```
11.18.2.2 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 61 of file bitmap.

```
11.18.2.3 void cxx::Bitmap_base::bit(long bit, bool on) throw) [inline]
```

Set the value of bit bit to on.

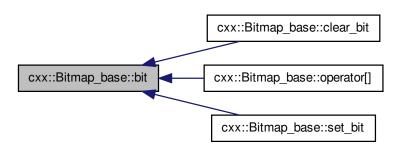
Parameters

bit	the number of the bit
on	the boolean value that shall be assigned to the bit.

Definition at line 146 of file bitmap.

Referenced by clear_bit(), operator[](), and set_bit().

Here is the caller graph for this function:



11.18.2.4 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 155 of file bitmap.

References bit().

Here is the call graph for this function:



11.18.2.5 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 164 of file bitmap.

References bit().

Here is the call graph for this function:



11.18.2.6 unsigned long cxx::Bitmap_base::operator[](long bit) const throw) [inline]

Get the truth value of a bit.

Parameters

bit the number of the bit to read.

Definition at line 173 of file bitmap.

References bit().

Here is the call graph for this function:



11.18.2.7 long cxx::Bitmap_base::scan_zero (long max_bit, long start_bit = 0) const throw) [inline]

Scans for the first zero bit.

Parameters

max_bit	the upper bound for the scanning operation.
start_bit	the number of the first bit to look at.

Definition at line 194 of file bitmap.

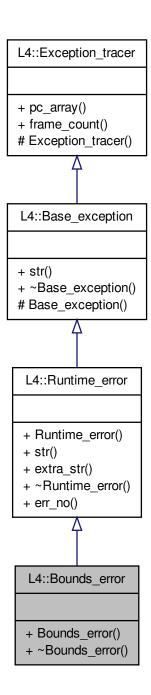
The documentation for this class was generated from the following file:

• I4/cxx/bitmap

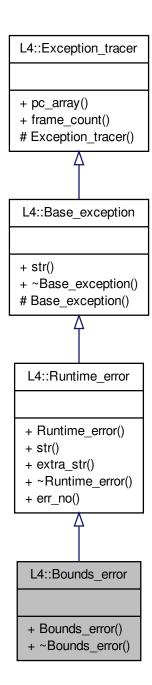
11.19 L4::Bounds_error Class Reference

Access out of bounds.

Inheritance diagram for L4::Bounds_error:



Collaboration diagram for L4::Bounds_error:



Additional Inherited Members

11.19.1 Detailed Description

Access out of bounds.

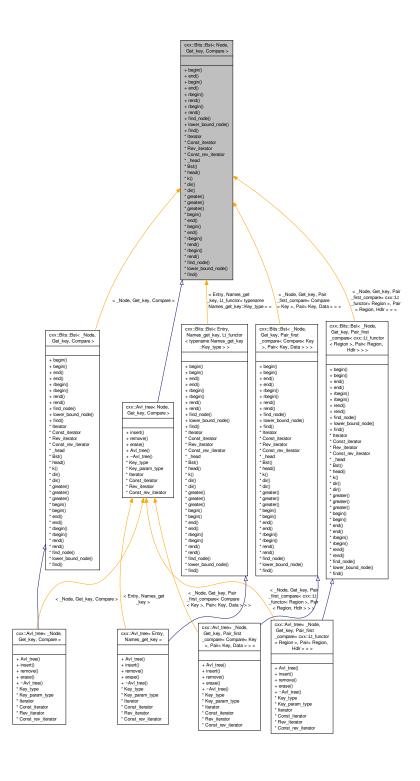
Definition at line 273 of file exceptions.

Basic binary search tree (BST).

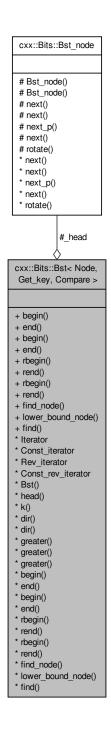
#include <bst.h>

The documentation for this class was generated from the following file:		
• 14/cxx	/exceptions	
11.20 c	xx::Bits::Bst< Node, Get_key, Compare > Class Template Reference	

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_iter_ops

Helper for building reverse iterators for different wrapper classes.

Iterators

 typedef <u>__Bst_iter</u>< Node, Node, Fwd > <u>Iterator</u>

Forward iterator.

 typedef <u>Bst_iter</u>< 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.

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 I, Key_param_type r)

Get the direction to go from I to search for r.

static Dir dir (Key_param_type I, Bst_node const *r)

Get the direction to go from I to search for r.

static bool greater (Key_param_type I, Key_param_type r)

Is I greater than r.

static bool greater (Key_param_type I, Bst_node const *r)

Is I greater than r.

• static bool greater (Bst_node const *I, Bst_node const *r)

Is I greater than r.

11.20.1 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.

11.20.2 Member Function Documentation

Get the direction to go from I to search for r.

Parameters

1	is the key to look for.
r	is the key at the current position.

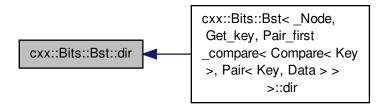
Returns

#Direction::L for left, #Direction::R for right, and #Direction::N if / is equal to r.

Definition at line 117 of file bst.h.

Referenced by cxx::Bits::Bst< _Node, Get_key, Pair_first_compare< Compare< Key >, Pair< Key, Data > >::dir().

Here is the caller graph for this function:



11.20.2.2 template<typename Node, typename Get_key, typename Compare> static Dir cxx::Bits::Bst< Node, Get_key, Compare>::dir(Key_param_type I, Bst_node const * r) [inline], [static], [protected]

Get the direction to go from I to search for r.

Parameters

1	is the key to look for.
r	is the node at the current position.

Returns

#Direction::L for left, #Direction::R for right, and #Direction::N if / is equal to r.

Definition at line 133 of file bst.h.

11.20.2.3 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.

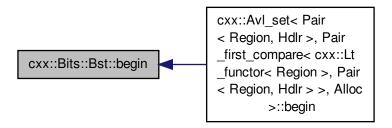
Returns

Constant forward iterator for the first element in the set.

Definition at line 159 of file bst.h.

Referenced by cxx::Avl_set< Pair< Region, Hdlr >, Pair_first_compare< cxx::Lt_functor< Region >, Pair< Region, Hdlr > >, Alloc >::begin().

Here is the caller graph for this function:



11.20.2.4 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.

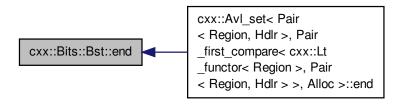
Returns

The end marker for the constant forward iterator.

Definition at line 164 of file bst.h.

Referenced by cxx::Avl_set< Pair< Region, Hdlr >, Pair_first_compare< cxx::Lt_functor< Region >, Pair< Region, Hdlr > >, Alloc >::end().

Here is the caller graph for this function:



11.20.2.5 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 170 of file bst.h.

11.20.2.6 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 175 of file bst.h.

11.20.2.7 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.

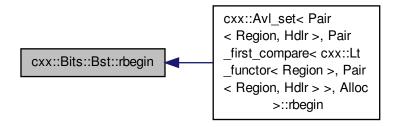
Returns

The constant backward iterator for the last element in the set.

Definition at line 181 of file bst.h.

Referenced by cxx::Avl_set< Pair< Region, Hdlr >, Pair_first_compare< cxx::Lt_functor< Region >, Pair< Region, Hdlr >, Alloc >::rbegin().

Here is the caller graph for this function:



11.20.2.8 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.

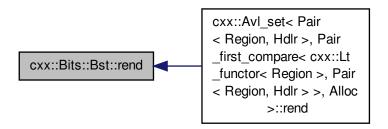
Returns

The end marker for the constant backward iterator.

Definition at line 186 of file bst.h.

Referenced by cxx::Avl_set< Pair< Region, Hdlr >, Pair_first_compare< cxx::Lt_functor< Region >, Pair< Region, Hdlr > >, Alloc >::rend().

Here is the caller graph for this function:



11.20.2.9 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 192 of file bst.h.

11.20.2.10 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 197 of file bst.h.

11.20.2.11 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

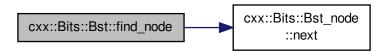
A pointer to the node with the given key, or NULL if key was not found.

Definition at line 236 of file bst.h.

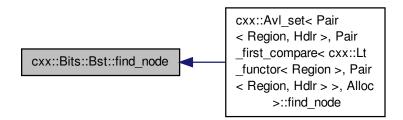
References cxx::Bits::Bst_node::next().

Referenced by cxx::Avl_set< Pair< Region, Hdlr >, Pair_first_compare< cxx::Lt_functor< Region >, Pair< Region, Hdlr >, Alloc >::find_node().

Here is the call graph for this function:



Here is the caller graph for this function:



11.20.2.12 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

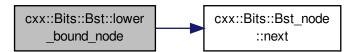
A pointer to the node with the given key, or NULL if key was not found.

Definition at line 252 of file bst.h.

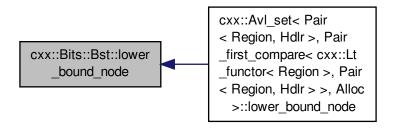
References cxx::Bits::Bst_node::next().

Referenced by cxx::Avl_set< Pair< Region, Hdlr >, Pair_first_compare< cxx::Lt_functor< Region >, Pair< Region, Hdlr >, Alloc >::lower_bound_node().

Here is the call graph for this function:



Here is the caller graph for this function:



11.20.2.13 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.

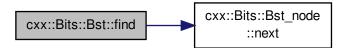
Returns

A valid iterator for the node with the given key, or an invalid iterator if key was not found.

Definition at line 272 of file bst.h.

References cxx::Bits::Bst_node::next().

Here is the call graph for this function:



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

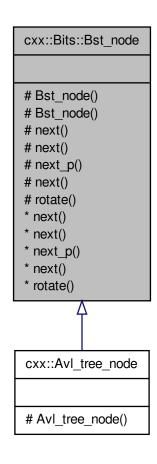
• I4/cxx/bits/bst.h

11.21 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:

```
# Bst_node()
# Bst_node()
# next()
# next()
# next_p()
# next()
# rotate()
* next()
* next()
* next()
* next()
* retate()
* rotate()
* rotate()
* rotate()
```

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.

 $\bullet \ \ \mathsf{template}{<} \mathsf{typename} \ \mathsf{Node} >$

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.

11.21.1 Detailed Description

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

Definition at line 77 of file bst base.h.

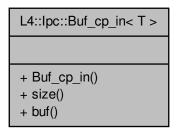
The documentation for this class was generated from the following file:

• I4/cxx/bits/bst_base.h

11.22 L4::lpc::Buf_cp_in < T > Class Template Reference

Abstraction for extracting array from an Ipc::Istream.

Collaboration diagram for L4::lpc::Buf cp in< T>:



Public Member Functions

Buf_cp_in (T *v, unsigned long &size)
 Create a buffer for extracting an array from an lpc::lstream.

11.22.1 Detailed Description

template<typename T>class L4::lpc::Buf_cp_in< T>

Abstraction for extracting array from an Ipc::Istream.

An instance of Buf_cp_in can be used to extract an array from an lpc::Istream. This is the counterpart to the Buf_cp_out abstraction. 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 cp in(), Buf in, buf in(), Buf cp out, and buf cp out().
```

Definition at line 118 of file ipc_stream.

11.22.2 Constructor & Destructor Documentation

11.22.2.1 template < typename T > L4::lpc::Buf_cp_in < T >::Buf_cp_in (T * v, unsigned long & size) [inline] Create a buffer for extracting an array from an lpc::lstream.

Parameters

V	The buffer for array (copy in).
size	Input: the number of elements the array can take at most
	Output: the number of elements found in the stream.

Definition at line 127 of file ipc_stream.

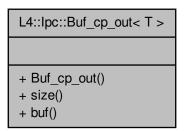
The documentation for this class was generated from the following file:

· I4/cxx/ipc_stream

11.23 L4::lpc::Buf_cp_out < T > Class Template Reference

Abstraction for inserting an array into an Ipc::Ostream.

Collaboration diagram for L4::lpc::Buf_cp_out< T >:



Public Member Functions

• Buf_cp_out (T const *v, unsigned long size)

Create a buffer object for the given array.

• unsigned long size () const

Get the number of elements in the array.

• T const * buf () const

Get the pointer to the array.

11.23.1 Detailed Description

template<typename T>class L4::lpc::Buf_cp_out< T>

Abstraction for inserting an array into an Ipc::Ostream.

An object of Buf_cp_out can be used to insert an array of arbitrary values, that can be inserted into an lpc::Ostream individually. The array is therefore copied to the message buffer, in contrast to data handled with Msg_out_buffer or Msg_io_buffer.

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.

You should use buf_cp_out() to create instances of Buf_cp_out.

The counterpart is either Buf_cp_in (buf_cp_in()) or Buf_in (buf_in()).

Definition at line 62 of file ipc_stream.

11.23.2 Constructor & Destructor Documentation

11.23.2.1 template < typename T> L4::Ipc::Buf_cp_out < T>::Buf_cp_out (T const * v, unsigned long size) [inline]

Create a buffer object for the given array.

Parameters

V	The pointer to the array with size elements of type T.
size	the number of elements in the array.

Definition at line 70 of file ipc_stream.

11.23.3 Member Function Documentation

11.23.3.1 template < typename T > unsigned long L4::lpc::Buf_cp_out < T >::size() const [inline]

Get the number of elements in the array.

Note

This function is usually used by the lpc::Ostream itself.

Definition at line 76 of file ipc_stream.

Referenced by operator<<().

Here is the caller graph for this function:



11.23.3.2 template < typename T > T const* L4::lpc::Buf_cp_out < T >::buf() const [inline]

Get the pointer to the array.

Note

This function is usually used by the Ipc::Ostream itself.

Definition at line 82 of file ipc_stream.

Referenced by operator<<().

Here is the caller graph for this function:



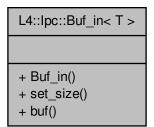
The documentation for this class was generated from the following file:

• I4/cxx/ipc_stream

11.24 L4::lpc::Buf_in < T > Class Template Reference

Abstraction to extract an array from an lpc::lstream.

Collaboration diagram for L4::lpc::Buf_in< T >:



Public Member Functions

Buf_in (T *&v, unsigned long &size)
 Create an Buf_in to adjust a pointer to the array and the size of the array.

11.24.1 Detailed Description

template < typename T > class L4::lpc::Buf_in < T >

Abstraction to extract an array from an lpc::lstream.

This wrapper provides a possibility to extract an array from an lpc::lstream, 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 mechanism is comparable to that of Msg_ptr, however it handles arrays inserted with Buf_cp_out.

See buf_in(), Buf_cp_out, buf_cp_out(), Buf_cp_in, and buf_cp_in().

Definition at line 212 of file ipc_stream.

11.24.2 Constructor & Destructor Documentation

11.24.2.1 template<typename T> L4::lpc::Buf_in< T>::Buf_in (T *& v, unsigned long & size) [inline]

Create an Buf_in to adjust a pointer to the array and the size of the array.

Parameters

V	The pointer to adjust to the first element of the array.
size	The number of elements found in the stream.

Definition at line 221 of file ipc_stream.

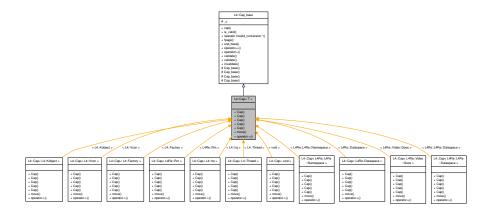
The documentation for this class was generated from the following file:

· I4/cxx/ipc_stream

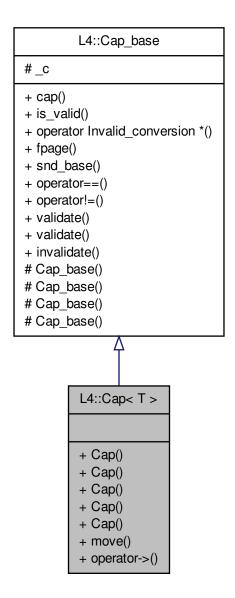
11.25 L4::Cap < T > Class Template Reference

Capability Selector a la C++.

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 (I4_default_caps_t cap) throw ()

Initialize capability with one of the default capability selectors.

• Cap (I4_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.

• T * operator-> () const throw ()

Member access of a T.

Friends

· class L4::Kobject

Additional Inherited Members

11.25.1 Detailed Description

template<typename T>class L4::Cap< T>

Capability Selector a la C++.

Template Parameters

The type of the object the capability points to

The C++ version of a capability looks just as a pointer, in fact it is a kind of a smart pointer for our kernel objects and the objects derived from the kernel objects (L4::Kobject).

Examples:

examples/clntsrv/client.cc, examples/libs/l4re/c++/shared_ds/ds_clnt.cc, and examples/libs/l4re/streammap/client.-cc.

Definition at line 47 of file capability.

11.25.2 Constructor & Destructor Documentation

```
11.25.2.1 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 is the source selector that shall be copied (and casted).

Definition at line 224 of file capability.

11.25.2.2 template < typename T > L4::Cap < T >::Cap (I4 default caps t cap) throw) [inline]

Initialize capability with one of the default capability selectors.

Parameters

cap Capability selector.

Definition at line 236 of file capability.

11.25.2.3 template < typename T > L4::Cap < T >::Cap (I4_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 242 of file capability.

11.25.3 Member Function Documentation

11.25.3.1 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 255 of file capability.

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

· I4/sys/capability

11.26 L4Re::Cap_alloc Class Reference

Capability allocator interface.

Collaboration diagram for L4Re::Cap_alloc:

L4Re::Cap_alloc

- + alloc()
- + alloc()
- + free()
- + ~Cap_alloc()
- + get_cap_alloc()
- # Cap_alloc()
- # Cap_alloc()

Public Member Functions

virtual L4::Cap< void > alloc ()=0 throw ()

Allocate a capability.

 $\bullet \ \ template {<} typename \ T >$

L4::Cap< T> alloc () throw ()

Allocate a capability.

virtual void free (L4::Cap< void > cap)=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.

11.26.1 Detailed Description

Capability allocator interface.

Definition at line 40 of file cap_alloc.

11.26.2 Member Function Documentation

11.26.2.1 virtual L4::Cap<void> L4Re::Cap_alloc::alloc()throw) [pure virtual]

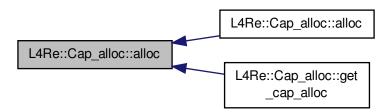
Allocate a capability.

Returns

Capability of type void

Referenced by alloc(), and get_cap_alloc().

Here is the caller graph for this function:



11.26.2.2 template<typename T > L4::Cap<T> L4Re::Cap_alloc::alloc() throw) [inline]

Allocate a capability.

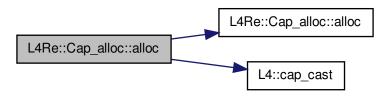
Returns

Capability of type T

Definition at line 62 of file cap_alloc.

References alloc(), and L4::cap_cast().

Here is the call graph for this function:



11.26.2.3 virtual void L4Re::Cap_alloc::free (L4::Cap < void > cap) throw) [pure virtual]

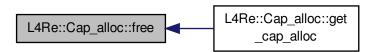
Free a capability.

Parameters

```
cap Capability to free.
```

Referenced by get_cap_alloc().

Here is the caller graph for this function:



11.26.2.4 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

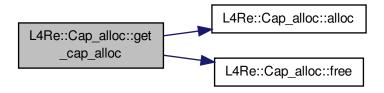
Returns

Instance of a capability allocator.

Definition at line 85 of file cap_alloc.

References alloc(), and free().

Here is the call graph for this function:



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

• I4/re/cap_alloc

11.27 L4Re::Util::Cap_alloc_base Class Reference

Capability allocator.

Inherited by L4Re::Util::Cap_alloc< Size >.

Collaboration diagram for L4Re::Util::Cap_alloc_base:

```
+ throw()
+ alloc()
+ hint()
+ alloc()
+ is_allocated()
+ free()
+ take()
+ release()
+ last()
```

Public Member Functions

Allocate a capability slot.

template<typename T >
 void free (L4::Cap< T > const &cap, I4_cap_idx_t task=-1UL, I4_umword_t unmap_flags=L4_FP_ALL_SP-ACES) throw ()

Free a capability slot.

11.27.1 Detailed Description

Capability allocator.

Definition at line 38 of file bitmap_cap_alloc.

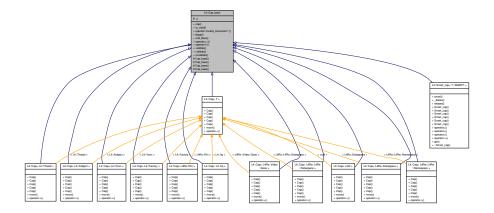
The documentation for this class was generated from the following file:

• I4/re/util/bitmap_cap_alloc

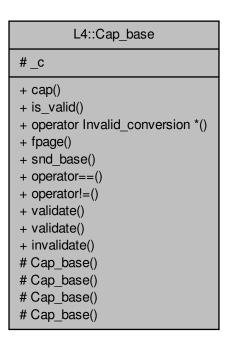
11.28 L4::Cap_base Class Reference

Base class for all kinds of capabilities.

Inheritance diagram for L4::Cap_base:



Collaboration diagram for L4::Cap_base:



Public Types

- enum No_init_type { No_init }
- enum Cap_type { Invalid = L4_INVALID_CAP }
 Invalid capability type.

Public Member Functions

• I4_cap_idx_t cap () const throw ()

Return capability selector.

bool is_valid () const throw ()

Test whether capability selector is not the invalid capability selector.

I4_fpage_t fpage (unsigned rights=L4_FPAGE_RWX) const throw ()

Returns flex-page of the capability selector.

• I4_umword_t snd_base (unsigned grant=0, I4_cap_idx_t base=L4_INVALID_CAP) const throw ()

Returns send base.

• bool operator== (Cap_base const &o) const throw ()

Test if two capability selectors are equal.

bool operator!= (Cap_base const &o) const throw ()

Test if two capability selectors are not equal.

I4_msgtag_t validate (I4_utcb_t *u=I4_utcb()) const throw ()

Check whether a capability selector points to a valid capability.

I4_msgtag_t validate (Cap< Task > task, I4_utcb_t *u=I4_utcb()) const throw ()

Check whether a capability selector points to a valid capability.

• void invalidate () throw ()

Set this selector to the invalid capability (L4_INVALID_CAP).

Protected Member Functions

• Cap_base (I4_cap_idx_t c) throw ()

Generate a capability from its C representation.

Cap_base (Cap_type cap) throw ()

Constructor to create an invalid capability selector.

Cap_base (I4_default_caps_t cap) throw ()

Initialize capability with one of the default capability selectors.

Cap_base () throw ()

Create an uninitialized instance.

Protected Attributes

• 14 cap idx t c

The C representation of a capability selector.

11.28.1 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 65 of file capability.

```
11.28.2 Member Enumeration Documentation
```

11.28.2.1 enum L4::Cap_base::No_init_type

Enumerator

No_init Special value for constructing uninitialized Cap objects.

Definition at line 71 of file capability.

11.28.2.2 enum L4::Cap_base::Cap_type

Invalid capability type.

Enumerator

Invalid Invalid capability selector.

Definition at line 82 of file capability.

11.28.3 Constructor & Destructor Documentation

```
11.28.3.1 L4::Cap_base::Cap_base(I4_cap_idx_t c) throw) [inline], [explicit], [protected]
```

Generate a capability from its C representation.

Parameters

```
c the C capability selector
```

Definition at line 167 of file capability.

```
11.28.3.2 L4::Cap_base::Cap_base( I4_default_caps_t cap ) throw) [inline], [explicit], [protected]
```

Initialize capability with one of the default capability selectors.

Parameters

```
cap Capability selector.
```

Definition at line 177 of file capability.

11.28.4 Member Function Documentation

```
11.28.4.1 I4_cap_idx_t L4::Cap_base::cap() const throw) [inline]
```

Return capability selector.

Returns

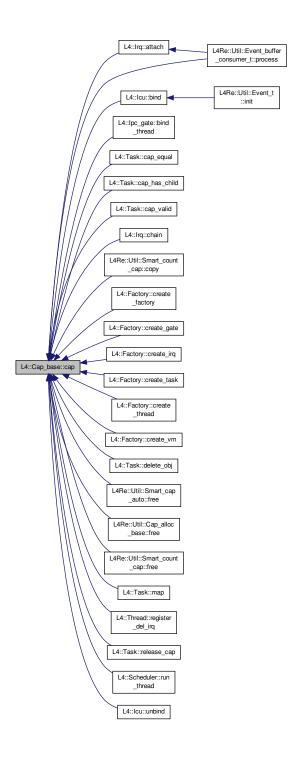
Capability selector.

Definition at line 91 of file capability.

References _c.

Referenced by L4::Irq::attach(), L4::Icu::bind(), L4::Ipc_gate::bind_thread(), L4::Task::cap_equal(), L4::Task::cap_has_child(), L4::Task::cap_valid(), L4::Irq::chain(), L4Re::Util::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(), L4::Task::delete_obj(), L4Re::Util::Smart_cap_auto< Unmap_flags >::free(), L4Re::Util::Cap_alloc_base::free(), L4Re::Util::Smart_count_cap< Unmap_flags >::free(), L4::Task::map(), L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process(), L4::Thread::register_del_irq(), L4::Task::release_cap(), L4::Scheduler::run_thread(), and L4::Icu::unbind().

Here is the caller graph for this function:



11.28.4.2 bool L4::Cap_base::is_valid() const throw) [inline]

Test whether capability selector is not the invalid capability selector.

Returns

True if capability is not invalid, false if invalid

Examples:

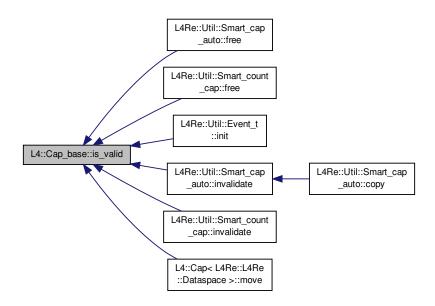
examples/libs/l4re/c++/mem_alloc/ma+rm.cc, and examples/libs/l4re/c++/shared_ds/ds_clnt.cc.

Definition at line 99 of file capability.

References c.

Referenced by L4Re::Util::Smart_cap_auto< Unmap_flags >::free(), L4Re::Util::Smart_count_cap< Unmap_flags >::free(), L4Re::Util::Smart_cap_auto< Unmap_flags >::invalidate(), L4-Re::Util::Smart_count_cap< Unmap_flags >::invalidate(), and L4::Cap< L4Re::L4Re::Dataspace >::move().

Here is the caller graph for this function:



11.28.4.3 I4_fpage_t L4::Cap_base::fpage(unsigned rights = L4_FPAGE_RWX) const throw) [inline]

Returns flex-page of the capability selector.

Parameters

rights	Rights, defaults to 'rwx'
	, <u> </u>

Returns

flex-page

Definition at line 109 of file capability.

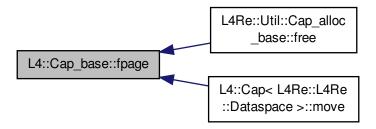
References _c, and I4_obj_fpage().

Referenced by L4Re::Util::Cap_alloc_base::free(), and L4::Cap< L4Re::L4Re::Dataspace >::move().

Here is the call graph for this function:



Here is the caller graph for this function:



11.28.4.4 I4_umword_t L4::Cap_base::snd_base (unsigned grant = 0, I4_cap_idx_t base = L4_INVALID_CAP) const throw) [inline]

Returns send base.

Parameters

grant	True object should be granted.
base	Base capability selector

Returns

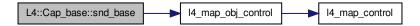
Map object.

Definition at line 118 of file capability.

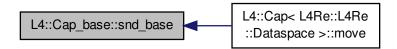
References c, L4 INVALID CAP, and I4 map obj control().

Referenced by L4::Cap< L4Re::L4Re::Dataspace >::move().

Here is the call graph for this function:



Here is the caller graph for this function:



```
11.28.4.5 I4_msgtag_t L4::Cap_base::validate(I4_utcb_t * u = I4_utcb()) const throw) [inline]
```

Check whether a capability selector points to a valid capability.

Parameters

```
u UTCB of the caller
```

Returns

label > 0 valid, label == 0 invalid

Definition at line 516 of file capability.

References L4_BASE_TASK_CAP.

11.28.4.6 I4_msgtag_t L4::Cap_base::validate (Cap < Task >
$$task$$
, I4_utcb_t * u = I4_utcb ()) const throw) [inline]

Check whether a capability selector points to a valid capability.

Parameters

и	UTCB of the caller
task	Task to check the capability in

Returns

label > 0 valid, label == 0 invalid

Definition at line 512 of file capability.

11.28.5 Field Documentation

11.28.5.1 I4_cap_idx_t L4::Cap_base::_c [protected]

The C representation of a capability selector.

Definition at line 186 of file capability.

Referenced by cap(), fpage(), invalidate(), is_valid(), operator!=(), L4::Smart_cap< T, SMART >::operator->(), L4::Cap< L4Re::L4Re::Dataspace >::operator->(), operator==(), and snd_base().

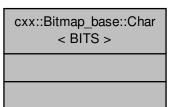
The documentation for this class was generated from the following file:

· I4/sys/capability

11.29 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 >:



11.29.1 Detailed Description

template < long BITS > class cxx::Bitmap_base::Char < BITS >

Helper abstraction for a byte contained in the bitmap.

Definition at line 66 of file bitmap.

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

· I4/cxx/bitmap

11.30 L4Re::Video::Color_component Class Reference

A color component.

Collaboration diagram for L4Re::Video::Color_component:

L4Re::Video::Color _component

- + Color_component()
- + Color_component()
- + size()
- + shift()
- + operator==()
- + get()
- + set()
- + dump()

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 STREAM >

STREAM & dump (STREAM &s) const

Dump information on the view information to a stream.

11.30.1 Detailed Description

A color component.

Definition at line 32 of file colors.

11.30.2 Constructor & Destructor Documentation

11.30.2.1 L4Re::Video::Color_component::Color_component (unsigned char bits, unsigned char shift) [inline]

Constructor.

Parameters

bits	Number of bits used by the component
shift	Position in bits of the component in the pixel

Definition at line 47 of file colors.

11.30.3 Member Function Documentation

11.30.3.1 unsigned char L4Re::Video::Color_component::size () const [inline]

Return the number of bits used by the component.

Returns

Number of bits used by the component

Definition at line 54 of file colors.

11.30.3.2 unsigned char L4Re::Video::Color_component::shift() const [inline]

Return the position of the component in the pixel.

Returns

Position in bits of the component in the pixel

Definition at line 60 of file colors.

11.30.3.3 bool L4Re::Video::Color_component::operator== (Color_component const & o) const [inline]

Compare for equality.

Returns

True if the same components are described, false if not.

Definition at line 66 of file colors.

11.30.3.4 int L4Re::Video::Color_component::get (unsigned long v) const [inline]

Get component from value (normalized to 16bits).

Parameters

V	Value

Returns

Converted value

Definition at line 74 of file colors.

11.30.3.5 long unsigned L4Re::Video::Color_component::set(int v) const [inline]

Transform 16bit normalized value to the component in the color space.

D -			- 4	L	
Pa	ra	m	ല	P	rs

V	Value return Converted value.

Definition at line 85 of file colors.

11.30.3.6 template < typename STREAM > STREAM & L4Re::Video::Color_component::dump (STREAM & s) const [inline]

Dump information on the view information to a stream.

Parameters

S	Stream

Returns

The stream

Definition at line 94 of file colors.

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

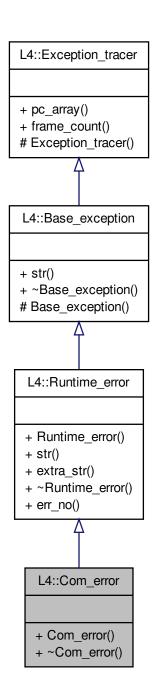
• I4/re/video/colors

11.31 L4::Com_error Class Reference

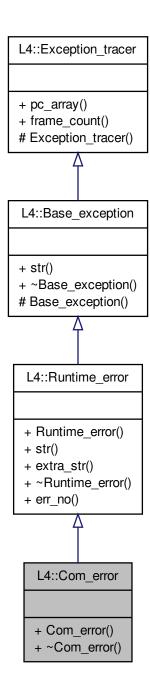
Error conditions during IPC.

#include <14/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

11.31.1 Detailed Description

Error conditions during IPC.

This exception encapsulates all IPC error conditions of L4 IPC.

Definition at line 258 of file exceptions.

11.31.2 Constructor & Destructor Documentation

11.31.2.1 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 (I4_ipc... return value).
```

Definition at line 265 of file exceptions.

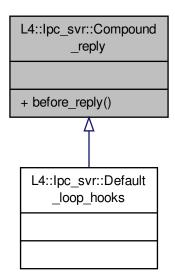
The documentation for this class was generated from the following file:

• I4/cxx/exceptions

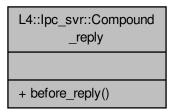
11.32 L4::lpc_svr::Compound_reply Struct Reference

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

Inheritance diagram for L4::lpc_svr::Compound_reply:



Collaboration diagram for L4::lpc_svr::Compound_reply:



11.32.1 Detailed Description

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

Definition at line 73 of file ipc_server.

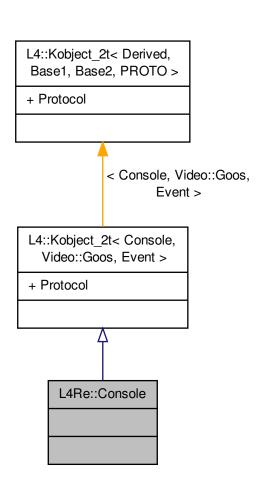
The documentation for this struct was generated from the following file:

• I4/cxx/ipc_server

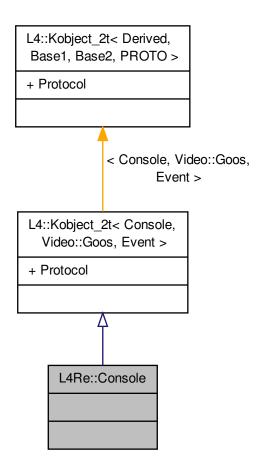
11.33 L4Re::Console Class Reference

Console class.

Inheritance diagram for L4Re::Console:



Collaboration diagram for L4Re::Console:



11.33.1 Detailed Description

Console class.

Definition at line 39 of file console.

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

• I4/re/console

11.34 L4Re::Util::Counting_cap_alloc< COUNTERTYPE > Class Template Reference

Reference-counting cap allocator.

 $Collaboration\ diagram\ for\ L4Re::Util::Counting_cap_alloc < COUNTERTYPE >:$

L4Re::Util::Counting
_cap_alloc< COUNTERTYPE >

+ alloc()
+ alloc()
+ take()
+ free()
+ release()
+ last()
Counting_cap_alloc()
setup()

11.34.1 Detailed Description

 $template < typename \ COUNTERTYPE = L4Re::Util::Counter < unsigned \ char >> class \ L4Re::Util::Counting_cap_alloc < COUNTERTYPE >$

Reference-counting cap allocator.

Definition at line 52 of file counting_cap_alloc.

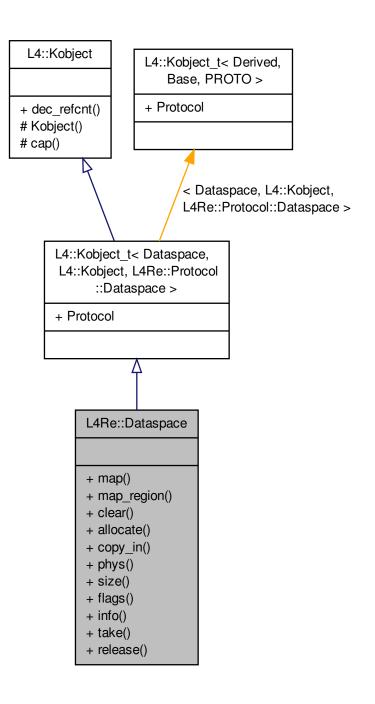
The documentation for this class was generated from the following file:

• I4/re/util/counting_cap_alloc

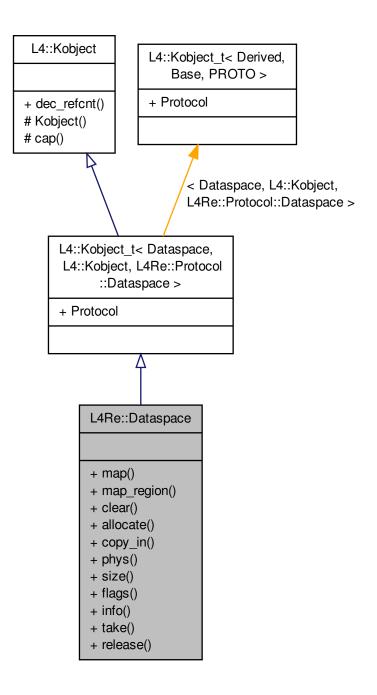
11.35 L4Re::Dataspace Class Reference

This class represents a data space.

Inheritance diagram for L4Re::Dataspace:



Collaboration diagram for L4Re::Dataspace:



Data Structures

• struct Stats

Information about the data space.

Flags for map operations.

Public Types

• enum Map_flags { Map_ro = 0, Map_rw = 1 }

Public Member Functions

long map (I4_addr_t offset, unsigned long flags, I4_addr_t local_addr, I4_addr_t min_addr, I4_addr_t max_addr) const throw ()

Request a flex-page mapping from the data space.

long map_region (I4_addr_t offset, unsigned long flags, I4_addr_t min_addr, I4_addr_t max_addr) const throw
 ()

Map a part of a data space completely.

• long clear (I4 addr t offset, unsigned long size) const throw ()

Clear parts of a data space.

• long allocate (I4_addr_t offset, I4_size_t size) throw ()

Allocate a range in the dataspace.

 long copy_in (l4_addr_t dst_offs, L4::Cap< Dataspace > src, l4_addr_t src_offs, unsigned long size) const throw ()

Copy data space contents.

• long phys (I4_addr_t offset, I4_addr_t &phys_addr, I4_size_t &phys_size) const throw ()

Get the physical addresses of a data space.

• long size () const throw ()

Get size of a data space.

• long flags () const throw ()

Get flags of the data space.

• int info (Stats *stats) const throw ()

Get information on the data space.

Additional Inherited Members

11.35.1 Detailed Description

This class represents a data space.

For more details, see Data-Space API.

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 67 of file dataspace.

11.35.2 Member Enumeration Documentation

11.35.2.1 enum L4Re::Dataspace::Map_flags

Flags for map operations.

Enumerator

Map_ro Request read-only mapping.

Map_rw Request writable mapping.

Definition at line 77 of file dataspace.

11.35.3 Member Function Documentation

11.35.3.1 long L4Re::Dataspace::map (I4_addr_t offset, unsigned long flags, I4_addr_t local_addr, I4_addr_t min_addr, I4_addr_t max_addr) const throw)

Request a flex-page mapping from the data space.

Parameters

offset	Offset to start within data space
flags	map flags, see Map_flags.
local_addr	Local address to map to.
min_addr	Defines start of receive window.
max_addr	Defines end of receive window.

Returns

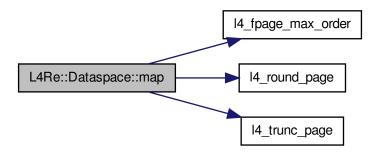
0 on success, <0 on error

- -L4_ERANGE
- -L4 EPERM
- · IPC errors

Definition at line 94 of file dataspace_impl.h.

References I4_fpage_max_order(), L4_LOG2_PAGESIZE, I4_round_page(), and I4_trunc_page().

Here is the call graph for this function:



11.35.3.2 long L4Re::Dataspace::map_region (I4_addr_t offset, unsigned long flags, I4_addr_t min_addr, I4_addr_t max_addr) const throw)

Map a part of a data space completely.

Parameters

offset	Offset to start within data space

flags	map flags, see Map_flags.
min_addr	Defines start of receive window.
max_addr	Defines end of receive window.

Returns

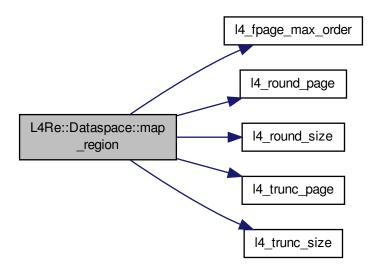
0 on success, <0 on error

- -L4_ERANGE
- -L4_EPERM
- · IPC errors

Definition at line 57 of file dataspace_impl.h.

References EXPECT_FALSE, I4_fpage_max_order(), L4_LOG2_PAGESIZE, I4_round_page(), I4_round_size(), I4_trunc_page(), and I4_trunc_size().

Here is the call graph for this function:



11.35.3.3 long L4Re::Dataspace::clear (I4_addr_t offset, unsigned long size) const throw)

Clear parts of a data space.

Parameters

offset	Offset within data space.
size	Size to clear (in bytes).

Returns

>0 on sucess, <0 on error.

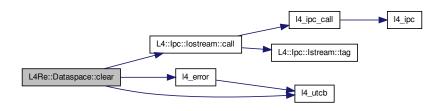
- -L4 EACCESS
- · IPC errors

Clears memory. Depending on the type of memory the memory could also be deallocated and replaced by shared zero-page.

Definition at line 108 of file dataspace_impl.h.

References L4::lpc::lostream::call(), L4Re::Protocol::Dataspace, EXPECT_FALSE, I4_error(), and I4_utcb().

Here is the call graph for this function:



11.35.3.4 long L4Re::Dataspace::allocate (I4_addr_t offset, I4_size_t size) throw)

Allocate a range in the dataspace.

Parameters

offset	Offset in the dataspace, in bytes.
size	Size of the range, in bytes.

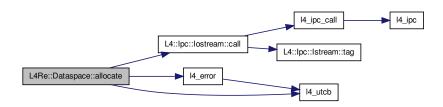
Returns

0 on success, <0 on error

Definition at line 177 of file dataspace impl.h.

References L4::lpc::lostream::call(), L4Re::Protocol::Dataspace, I4_error(), and I4_utcb().

Here is the call graph for this function:



11.35.3.5 long L4Re::Dataspace::copy_in (I4_addr_t *dst_offs*, L4::Cap< Dataspace > *src*, I4_addr_t *src_offs*, unsigned long *size*) const throw)

Copy data space contents.

Parameters

dst_offs	Offset in destination data space.
src	Source data space.
src_offs	Offset in the source data space.
size	Size to copy (in bytes).

Returns

0 on success, <0 on error

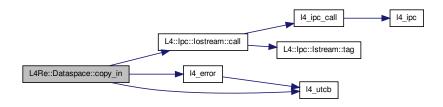
- -L4_EACCESS
- -L4 EINVAL
- · IPC errors

The copy operation may use copy-on-write mechanisms. The operation may also fail if both data spaces are not from the same data space manager or the data space managers do not cooperate.

Definition at line 155 of file dataspace_impl.h.

References L4::lpc::lostream::call(), L4Re::Protocol::Dataspace, I4_error(), and I4_utcb().

Here is the call graph for this function:



11.35.3.6 long L4Re::Dataspace::phys (I4_addr_t offset, I4_addr_t & phys_addr, I4_size_t & phys_size) const throw)

Get the physical addresses of a data space.

Parameters

offset	Offset in data space

Return values

phys_addr	Physical address.
phys_size	Size of largest physically contiguous region in the data space (in bytes).

Returns

0 on success, <0 on error

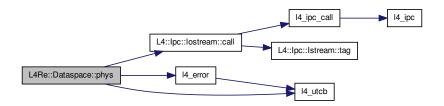
- -L4_EINVAL
- · IPC errors

Get the physical address(es) of a data space. This call will only succeed on pinned memory data spaces.

Definition at line 164 of file dataspace_impl.h.

References L4::lpc::lostream::call(), L4Re::Protocol::Dataspace, EXPECT_FALSE, I4_error(), and I4_utcb().

Here is the call graph for this function:



11.35.3.7 long L4Re::Dataspace::size () const throw)

Get size of a data space.

Returns

Size of the data space (in bytes), <0 on errors

• IPC errors

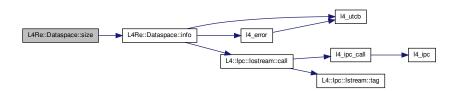
Examples:

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

Definition at line 135 of file dataspace_impl.h.

References info(), and L4Re::Dataspace::Stats::size.

Here is the call graph for this function:



11.35.3.8 long L4Re::Dataspace::flags () const throw)

Get flags of the data space.

Returns

Flags of the data space, <0 on errors

IPC errors

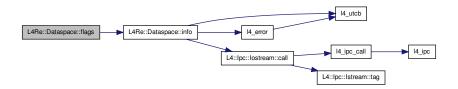
See Also

L4Re::Dataspace::Map_flags

Definition at line 145 of file dataspace_impl.h.

References L4Re::Dataspace::Stats::flags, and info().

Here is the call graph for this function:



11.35.3.9 int L4Re::Dataspace::info (Stats * stats) const throw)

Get information on the data space.

Return values

info	Data space information,

See Also

L4Re::Dataspace::Stats

Returns

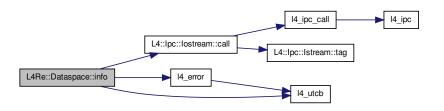
0 on success, < 0 on errors

Definition at line 122 of file dataspace_impl.h.

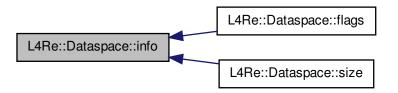
References L4::lpc::lostream::call(), L4Re::Protocol::Dataspace, EXPECT_FALSE, I4_error(), and I4_utcb().

Referenced by flags(), and size().

Here is the call graph for this function:



Here is the caller graph for this function:



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

- I4/re/dataspace
- I4/re/impl/dataspace_impl.h

11.36 L4Re::Util::Dataspace_svr Class Reference

Dataspace server class.

Collaboration diagram for L4Re::Util::Dataspace_svr:

```
L4Re::Util::Dataspace_svr
# _ds_start
# ds size
# map flags
#_cache_flags
# _rw_flags
+ Dataspace_svr()
+ ~Dataspace_svr()
+ map()
+ map_hook()
+ phys()
+ take()
+ release()
+ copy()
+ clear()
+ allocate()
+ page_shift()
+ is_static()
+ dispatch()
# size()
# map_flags()
# rw_flags()
# is_writable()
# page_size()
# round_size()
# check_limit()
# size()
```

Public Member Functions

• int map (I4_addr_t offset, I4_addr_t local_addr, unsigned long flags, I4_addr_t min_addr, I4_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 (I4_addr_t offset, I4_addr_t &phys_addr, I4_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 unsigned long copy (I4_addr_t dst_offs, I4_umword_t src_id, I4_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 (I4_addr_t offset, I4_size_t size, unsigned rights) 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.

• int dispatch (I4_umword_t obj, L4::lpc::lostream &ios)

Dispatch function.

11.36.1 Detailed Description

Dataspace server class.

The default implementation of the interface provides a continiously mapped dataspace.

Definition at line 48 of file dataspace_svr.

11.36.2 Member Function Documentation

11.36.2.1 int L4Re::Util::Dataspace_svr::map (I4_addr_t offset, I4_addr_t local_addr, unsigned long flags, I4_addr_t min_addr, I4_addr_t max_addr, L4::lpc::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 #Map_flags.
min_addr	Defines start of receive window.
max_addr	Defines end of receive window.

Return values

memory	Send fpage to map

Returns

0 on success, <0 on error

11.36.2.2 virtual int L4Re::Util::Dataspace_svr::map_hook (I4_addr_t offs, unsigned long flags, I4_addr_t min, I4_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

Returns

< 0 on error and the map request will be aborted with that error >= 0: ok

See Also

map

Definition at line 97 of file dataspace_svr.

```
11.36.2.3 virtual int L4Re::Util::Dataspace_svr::phys ( I4_addr_t offset, I4_addr_t & phys_addr, I4_size_t & phys_size ) throw ) [virtual]
```

Return physical address for a virtual address.

Parameters

offset	Offset into the dataspace

Return values

phys_addr	Physical address
phys_size	Size of continious physical region

Returns

Zero on success, else failure

```
11.36.2.4 virtual void L4Re::Util::Dataspace_svr::take() throw) [inline], [virtual]
```

Take a reference to this dataspace.

Default does nothing.

Definition at line 120 of file dataspace svr.

```
11.36.2.5 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 130 of file dataspace_svr.

```
11.36.2.6 virtual unsigned long L4Re::Util::Dataspace_svr::copy( I4_addr_t dst_offs, I4_umword_t src_id, I4_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

Returns

Number of bytes copied

Definition at line 143 of file dataspace_svr.

References L4_ENODEV.

11.36.2.7 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

11.36.2.8 virtual long L4Re::Util::Dataspace_svr::allocate (I4_addr_t offset, I4_size_t size, unsigned rights) throw)
[inline], [virtual]

Allocate a region within a dataspace.

Parameters

offset	Offset in the dataspace, in bytes.
size	Size of the range, in bytes.
size	Size of the range, in bytes.

Returns

0 on success, <0 on error

Definition at line 164 of file dataspace_svr.

References L4_ENODEV.

11.36.2.9 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 172 of file dataspace_svr.

References L4_LOG2_PAGESIZE.

11.36.2.10 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 180 of file dataspace_svr.

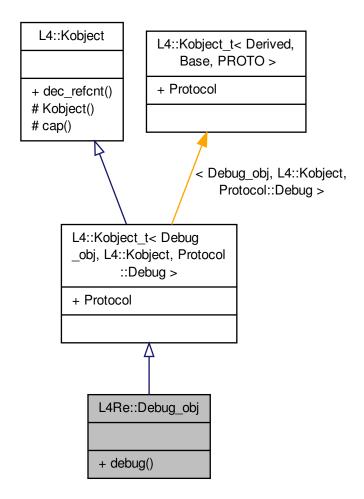
The documentation for this class was generated from the following file:

• I4/re/util/dataspace_svr

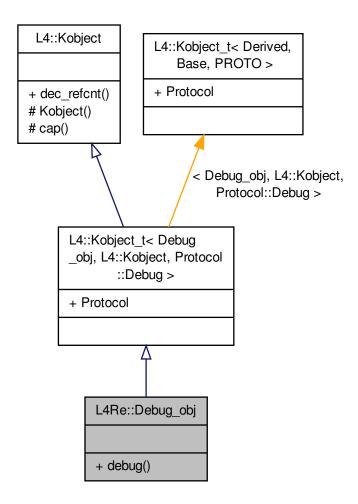
11.37 L4Re::Debug_obj Class Reference

Debug interface.

Inheritance diagram for L4Re::Debug_obj:



Collaboration diagram for L4Re::Debug_obj:



Public Member Functions

int debug (unsigned long function) const throw ()
 Debug call.

Additional Inherited Members

11.37.1 Detailed Description

Debug interface.

See Also

Debugging API.

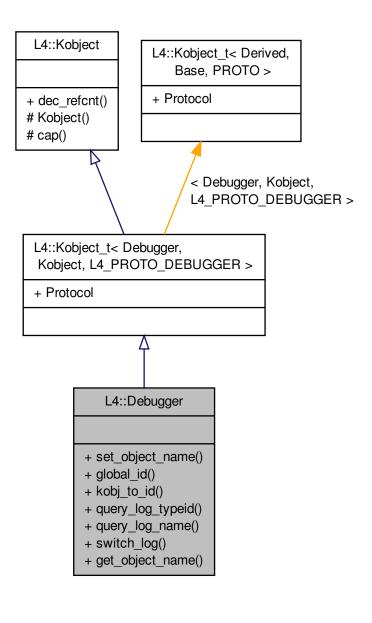
Definition at line 50 of file debug.

·	
11.37.2	Member Function Documentation
11.37.2.1	int L4Re::Debug_obj::debug (unsigned long function) const throw)
Debug ca	
Parameter	
	function Function to call.
Returns	
- L4	EOK
	• IPC errors
The docu	mentation for this class was generated from the following file:
14/	
• 14/1	re/debug
11 20	L Au Dobugger Class Poternos
11.38	L4::Debugger Class Reference

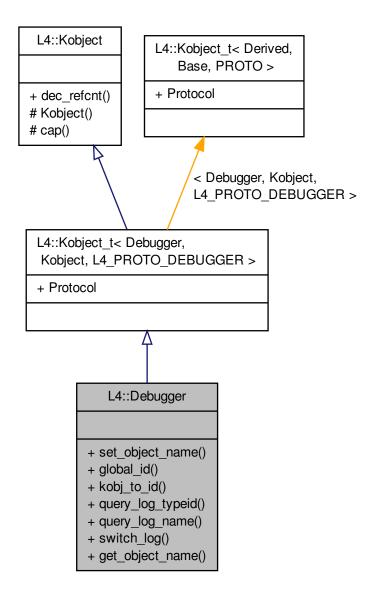
Generated for L4Re by Doxygen

Debugger interface.

Inheritance diagram for L4::Debugger:



Collaboration diagram for L4::Debugger:



Public Member Functions

- I4_msgtag_t set_object_name (const char *name, I4_utcb_t *utcb=I4_utcb()) throw ()
 The string name of kernel object.
- unsigned long global_id (l4_utcb_t *utcb=l4_utcb()) throw ()

Get the globally unique ID of the object behind a capability.

 $\bullet \ \ unsigned \ long \ kobj_to_id \ (l4_addr_t \ kobjp, \ l4_utcb_t \ *utcb=l4_utcb()) \ throw \ ()$

Get the globally unique ID of the object behind the kobject pointer.

- int query_log_typeid (const char *name, unsigned idx, I4_utcb_t *utcb=I4_utcb()) throw ()
- int query_log_name (unsigned idx, char *name, unsigned namelen, char *shortname, unsigned shortnamelen, l4_utcb_t *utcb=l4_utcb()) throw ()
- I4_msgtag_t switch_log (const char *name, unsigned on_off, I4_utcb_t *utcb=I4_utcb()) throw ()

• I4_msgtag_t get_object_name (unsigned id, char *name, unsigned size, I4_utcb_t *utcb=I4_utcb()) throw ()

Additional Inherited Members

11.38.1 Detailed Description

Debugger interface.

#include <14/sys/debugger>

Definition at line 40 of file debugger.

11.38.2 Member Function Documentation

The string name of kernel object.

Parameters

Cá	p Capability
nan	e Name

This is a debugging factility, the call might be invalid.

Note

the cap argument is the implicit this pointer.

Definition at line 55 of file debugger.

References L4::Kobject::cap().

Here is the call graph for this function:



11.38.2.2 unsigned long L4::Debugger::global_id (I4_utcb_t * utcb = I4_utcb ()) throw) [inline]

Get the globally unique ID of the object behind a capability.

Parameters

сар	Capability

Returns

 \sim 0UL on non-valid capability, ID otherwise

This is a debugging factility, the call might be invalid.

Note

the cap argument is the implicit this pointer.

Definition at line 63 of file debugger.

References L4::Kobject::cap().

Here is the call graph for this function:



11.38.2.3 unsigned long L4::Debugger::kobj_to_id (I4_addr_t kobjp, I4_utcb_t * utcb = I4_utcb ()) throw) [inline]

Get the globally unique ID of the object behind the kobject pointer.

Parameters

cap	Capability
kobjp	Kobject pointer

Returns

 \sim 0UL on non-valid capability or invalid kobject pointer, ID otherwise

This is a debugging factility, the call might be invalid.

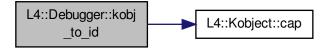
Note

the cap argument is the implicit this pointer.

Definition at line 70 of file debugger.

References L4::Kobject::cap().

Here is the call graph for this function:



11.38.2.4 int L4::Debugger::query_log_typeid (const char * name, unsigned idx, I4_utcb_t * utcb = I4_utcb ()) throw)
[inline]

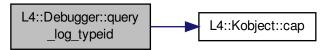
Note

the cap argument is the implicit this pointer.

Definition at line 78 of file debugger.

References L4::Kobject::cap().

Here is the call graph for this function:



11.38.2.5 int L4::Debugger::query_log_name (unsigned idx, char * name, unsigned namelen, char * shortname, unsigned shortnamelen, I4_utcb_t * utcb = I4_utcb ()) throw) [inline]

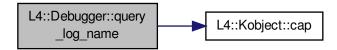
Note

the cap argument is the implicit this pointer.

Definition at line 86 of file debugger.

References L4::Kobject::cap().

Here is the call graph for this function:



11.38.2.6 I4_msgtag_t L4::Debugger::switch_log (const char * name, unsigned on_off, I4_utcb_t * utcb = I4_utcb ()) throw) [inline]

Note

the cap argument is the implicit this pointer.

Definition at line 99 of file debugger.

References L4::Kobject::cap().

Here is the call graph for this function:



11.38.2.7 I4_msgtag_t L4::Debugger::get_object_name (unsigned id, char * name, unsigned size, I4_utcb_t * utcb = I4_utcb()) throw) [inline]

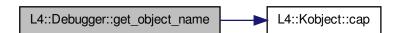
Note

the cap argument is the implicit this pointer.

Definition at line 107 of file debugger.

References L4::Kobject::cap().

Here is the call graph for this function:



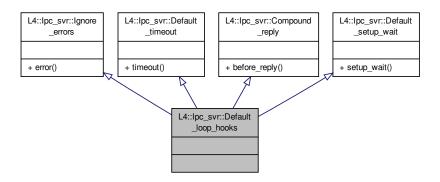
The documentation for this class was generated from the following file:

• I4/sys/debugger

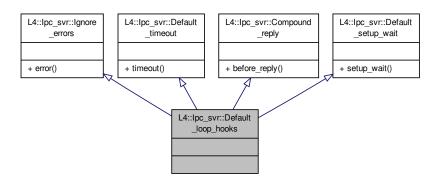
11.39 L4::lpc_svr::Default_loop_hooks Struct Reference

Default LOOP_HOOKS.

Inheritance diagram for L4::lpc_svr::Default_loop_hooks:



Collaboration diagram for L4::lpc_svr::Default_loop_hooks:



11.39.1 Detailed Description

Default LOOP_HOOKS.

Combination of Ignore_errors, Default_timeout, Compound_reply, and Default_setup_wait.

Definition at line 91 of file ipc_server.

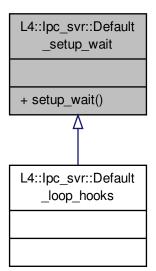
The documentation for this struct was generated from the following file:

• I4/cxx/ipc_server

11.40 L4::lpc_svr::Default_setup_wait Struct Reference

Mix in for LOOP_HOOKS for setup_wait no op.

Inheritance diagram for L4::lpc_svr::Default_setup_wait:



Collaboration diagram for L4::lpc_svr::Default_setup_wait:

L4::Ipc_svr::Default _setup_wait + setup_wait()

11.40.1 Detailed Description

Mix in for LOOP_HOOKS for setup_wait no op.

Definition at line 82 of file ipc_server.

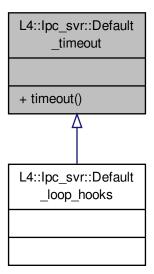
The documentation for this struct was generated from the following file:

• I4/cxx/ipc_server

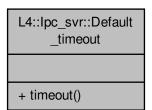
11.41 L4::lpc_svr::Default_timeout Struct Reference

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

Inheritance diagram for L4::lpc_svr::Default_timeout:



Collaboration diagram for L4::lpc_svr::Default_timeout:



11.41.1 Detailed Description

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

Definition at line 67 of file ipc_server.

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

• I4/cxx/ipc_server

11.42 cxx::Bits::Direction Struct Reference

The direction to go in a binary search tree.

```
#include <bst_base.h>
```

Collaboration diagram for cxx::Bits::Direction:

cxx::Bits::Direction + d + Direction() + Direction() + operator!() + operator==() + operator!=() + operator!=() * operator==() * operator!=() * operator!=() * operator!=() * operator!=() * operator!=()

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

11.42.1 Detailed Description

The direction to go in a binary search tree.

Definition at line 39 of file bst_base.h.

11.42.2 Member Enumeration Documentation

11.42.2.1 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.

11.42.3 Member Function Documentation

11.42.3.1 Direction cxx::Bits::Direction::operator! () const [inline]

Negate the direction.

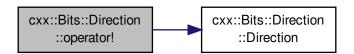
Note

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:

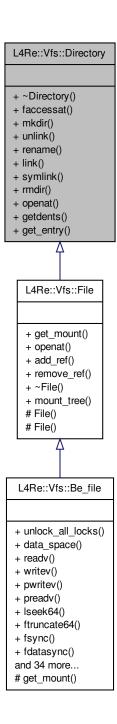
• I4/cxx/bits/bst_base.h

11.43 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:

L4Re::Vfs::Directory

- + ~Directory()
- + faccessat()
- + mkdir()
- + unlink()
- + rename()
- + link()
- + symlink()
- + rmdir()
- + openat()
- + getdents()
- + get_entry()

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 *)=0 throw ()
 - Delete an empty directory.

11.43.1 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 141 of file vfs.h.

11.43.2 Member Function Documentation

11.43.2.1 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: <i>path</i> 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.

11.43.2.2 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.

11.43.2.3 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 subdi-
	rectories.

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

11.43.2.4 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: <i>src_path</i> 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 subdi-
	rectories.

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

11.43.2.5 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 subdi-
	rectories.
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.

11.43.2.6 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 subdirec-
	tories.

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

11.43.2.7 virtual int L4Re::Vfs::Directory::rmdir(const char *) 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.

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

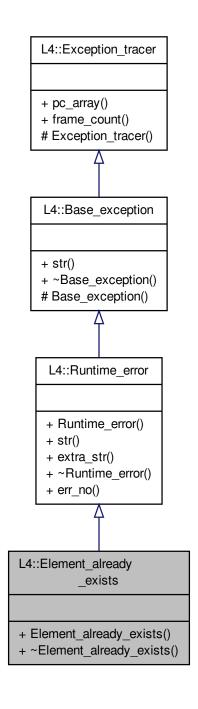
• I4/I4re_vfs/vfs.h

11.44 L4::Element_already_exists Class Reference

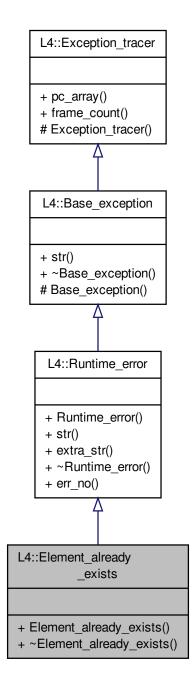
Exception for duplicate element insertions.

#include <14/cxx/exceptions>

Inheritance diagram for L4::Element_already_exists:



Collaboration diagram for L4::Element_already_exists:



Additional Inherited Members

11.44.1 Detailed Description

Exception for duplicate element insertions.

Definition at line 186 of file exceptions.

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

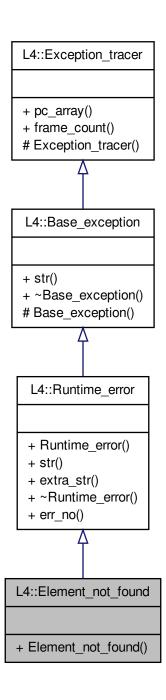
• I4/cxx/exceptions

11.45 L4::Element_not_found Class Reference

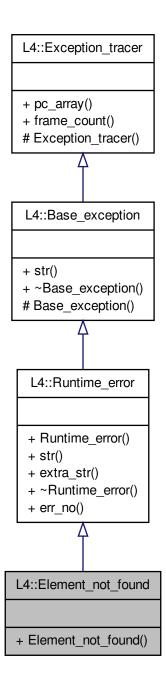
Exception for a failed lookup (element not found).

#include <14/cxx/exceptions>

Inheritance diagram for L4::Element_not_found:



Collaboration diagram for L4::Element_not_found:



Additional Inherited Members

11.45.1 Detailed Description

Exception for a failed lookup (element not found).

Definition at line 215 of file exceptions.

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

• I4/cxx/exceptions

11.46 Elf32_Dyn Struct Reference

ELF32 dynamic entry.

#include <elf.h>

Collaboration diagram for Elf32_Dyn:

+ d_tag + d_val + d_ptr + d_un

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

11.46.1 Detailed Description

ELF32 dynamic entry.

Definition at line 460 of file elf.h.

11.46.2 Field Documentation

11.46.2.1 Elf32_Word Elf32_Dyn::d_val

integer values with various interpret.

Definition at line 463 of file elf.h.

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

• I4/util/elf.h

11.47 Elf32_Ehdr Struct Reference

ELF32 header.

#include <elf.h>

Collaboration diagram for Elf32_Ehdr:

Elf32_Ehdr

- + e_ident
- + e_type
- + e_machine
- + e_version
- + e_entry
- + e_phoff
- + e_shoff
- + e_flags
- + e_ehsize
- + e_phentsize
- + e_phnum
- + e_shentsize
- + e_shnum
- + e_shstrndx

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

11.47.1 Detailed Description

ELF32 header.

Definition at line 118 of file elf.h.

11.47.2 Field Documentation

11.47.2.1 Elf32_Half Elf32_Ehdr::e_phnum

of entries in prog.

head. tab.

Definition at line 129 of file elf.h.

11.47.2.2 Elf32_Half Elf32_Ehdr::e_shnum

of entries in sect.

head. tab.

Definition at line 131 of file elf.h.

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

• I4/util/elf.h

11.48 Elf32_Phdr Struct Reference

ELF32 program header.

#include <elf.h>

Collaboration diagram for Elf32_Phdr:

Elf32_Phdr

- + p_type
- + p_offset
- + p_vaddr
- + p_paddr
- + p_filesz
- + p_memsz
- + p_flags
- + p_align

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

11.48.1 Detailed Description

ELF32 program header.

Definition at line 379 of file elf.h.

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

• I4/util/elf.h

11.49 Elf32_Shdr Struct Reference

ELF32 section header - figure 1-9, page 1-9.

#include <elf.h>

Collaboration diagram for Elf32_Shdr:

Elf32_Shdr

- + sh_name
- + sh_type
- + sh_flags
- + sh_addr
- + sh_offset
- + sh_size
- + sh_link
- + sh_info
- + sh_addralign
- + sh_entsize

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

11.49.1 Detailed Description

ELF32 section header - figure 1-9, page 1-9.

Definition at line 302 of file elf.h.

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

• I4/util/elf.h

11.50 Elf32_Sym Struct Reference

ELF32 symbol table entry.

#include <elf.h>

Collaboration diagram for Elf32_Sym:

Elf32_Sym

- + st_name
- + st_value
- + st_size
- + st_info
- + st_other
- + st_shndx

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

11.50.1 Detailed Description

ELF32 symbol table entry.

Definition at line 761 of file elf.h.

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

• I4/util/elf.h

11.51 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

11.51.1 Detailed Description

ELF64 dynamic entry.

Definition at line 469 of file elf.h.

11.51.2 Field Documentation

11.51.2.1 Elf64_Xword Elf64_Dyn::d_val

integer values with various interpret.

Definition at line 472 of file elf.h.

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

• I4/util/elf.h

11.52 Elf64_Ehdr Struct Reference

ELF64 header.

#include <elf.h>

Collaboration diagram for Elf64_Ehdr:

Elf64_Ehdr

- + e_ident
- + e_type
- + e_machine
- + e_version
- + e_entry
- + e_phoff
- + e_shoff
- + e_flags
- + e_ehsize
- + e_phentsize
- + e_phnum
- + e_shentsize
- + e_shnum
- + e_shstrndx

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

11.52.1 Detailed Description

ELF64 header.

Definition at line 138 of file elf.h.

11.52.2 Field Documentation

11.52.2.1 Elf64_Half Elf64_Ehdr::e_phnum

of entries in prog.

head. tab.

Definition at line 149 of file elf.h.

11.52.2.2 Elf64_Half Elf64_Ehdr::e_shnum

of entries in sect.

head. tab.

Definition at line 151 of file elf.h.

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

• I4/util/elf.h

11.53 Elf64_Phdr Struct Reference

ELF64 program header.

#include <elf.h>

Collaboration diagram for Elf64_Phdr:

Elf64_Phdr

- + p_type
- + p_flags
- + p_offset
- + p_vaddr
- + p_paddr
- + p_filesz
- + p_memsz
- + p_align

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

11.53.1 Detailed Description

ELF64 program header.

Definition at line 391 of file elf.h.

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

• I4/util/elf.h

11.54 Elf64_Shdr Struct Reference

ELF64 section header.

#include <elf.h>

Collaboration diagram for Elf64_Shdr:

Elf64_Shdr

- + sh_name
- + sh_type
- + sh_flags
- + sh_addr
- + sh_offset
- + sh_size
- + sh_link
- + sh_info
- + sh_addralign
- + sh_entsize

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

11.54.1 Detailed Description

ELF64 section header.

Definition at line 316 of file elf.h.

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

• I4/util/elf.h

11.55 Elf64_Sym Struct Reference

ELF64 symbol table entry.

#include <elf.h>

Collaboration diagram for Elf64_Sym:

+ st_name + st_info + st_other + st_shndx + st_value + st_size

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

11.55.1 Detailed Description

ELF64 symbol table entry.

Definition at line 771 of file elf.h.

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

• I4/util/elf.h

11.56 L4Re::Env Class Reference

Initial Environment (C++ version).

Collaboration diagram for L4Re::Env:

+ parent() + mem_alloc() + rm() + log() + main_thread() + task() + factory() + first_free_cap() + utcb_area() + first_free_utcb() and 16 more... + 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< 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.

    I4_cap_idx_t first_free_cap () const throw ()

      First available capability selector.
• I4_fpage_t utcb_area () const throw ()
      UTCB area of the task.

    I4_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 I) const throw ()

      Get the Cap_entry for the object named name.
• template<typename T >
  L4::Cap< T > get_cap (char const *name, unsigned I) 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 (I4_cap_idx_t c) throw ()

      Set first available capability selector.

    void utcb_area (I4_fpage_t utcbs) throw ()

      Set UTCB area of the task.

    void first_free_utcb (I4_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.

11.56.1 Detailed Description

Initial Environment (C++ version).

This class provides an initial set of capabilities as well as information the first free UTCB and used capability slots.

See Also

Initial environment

Definition at line 85 of file env.

11.56.2 Member Function Documentation

```
11.56.2.1 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/clintsrv/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 103 of file env.

```
11.56.2.2 L4::Cap<Parent> L4Re::Env::parent( ) const throw) [inline]
```

Object-capability to the parent.

Returns

Parent object-capability

Definition at line 110 of file env.

```
11.56.2.3 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 116 of file env.

```
11.56.2.4 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 122 of file env.
11.56.2.5 L4::Cap<Log> L4Re::Env::log( ) const throw) [inline]
 Object-capability to the logging service.
 Returns
      Log object-capability
 Definition at line 128 of file env.
11.56.2.6 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 134 of file env.
11.56.2.7 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 140 of file env.
References L4_BASE_TASK_CAP.
11.56.2.8 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 146 of file env.
```

```
11.56.2.9 I4_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 154 of file env.

```
11.56.2.10 I4_fpage_t L4Re::Env::utcb_area() const throw) [inline]
```

UTCB area of the task.

Returns

UTCB area

Definition at line 160 of file env.

```
11.56.2.11 I4_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 169 of file env.

```
11.56.2.12 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 176 of file env.

```
11.56.2.13 Cap_entry const* L4Re::Env::get ( char const * name, unsigned / ) const throw ) [inline]
```

Get the Cap_entry for the object named name.

Parameters

name	is the name of the object.
1	is the length of the name, thus <i>name</i> might mot 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 187 of file env.

References I4re_env_get_cap_I().

Here is the call graph for this function:



11.56.2.14 template < typename T > L4::Cap < T > L4Re::Env::get_cap (char const * name, unsigned /) const throw)
[inline]

Get the capability selector for the object named name.

Parameters

name	is the name of the object.
1	is the length of the name, thus <i>name</i> might mot 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 199 of file env.

References L4_ENOENT.

11.56.2.15 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 214 of file env.

11.56.2.16 void L4Re::Env::parent (L4::Cap < Parent > const & c) throw) [inline]

Set parent object-capability.

```
Parameters
```

```
c Parent object-capability
```

Definition at line 221 of file env.

```
11.56.2.17 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 227 of file env.

```
11.56.2.18 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 233 of file env.

```
11.56.2.19 void L4Re::Env::log ( L4::Cap < Log > const & c ) throw ) [inline]
```

Set log object-capability.

Parameters

```
c Log object-capability
```

Definition at line 239 of file env.

```
11.56.2.20 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 245 of file env.

```
11.56.2.21 void L4Re::Env::factory ( L4::Cap < L4::Factory > const & c ) throw ) [inline]
```

Set factory object-capability.

Parameters

```
c Factory object-capability
```

Definition at line 251 of file env.

```
11.56.2.22 void L4Re::Env::first_free_cap( I4_cap_idx_t c ) throw) [inline]
```

Set first available capability selector.

Parameters

c First capability selector available to the application.

Definition at line 257 of file env.

11.56.2.23 void L4Re::Env::utcb_area (I4_fpage_t utcbs) throw) [inline]

Set UTCB area of the task.

Parameters

```
utcbs UTCB area
```

Definition at line 263 of file env.

```
11.56.2.24 void L4Re::Env::first_free_utcb( I4_addr_t u ) throw) [inline]
```

Set first free UTCB.

Parameters

```
u First UTCB available for the application to use.
```

Definition at line 269 of file env.

```
11.56.2.25 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 277 of file env.

```
11.56.2.26 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 284 of file env.

```
11.56.2.27 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 292 of file env.

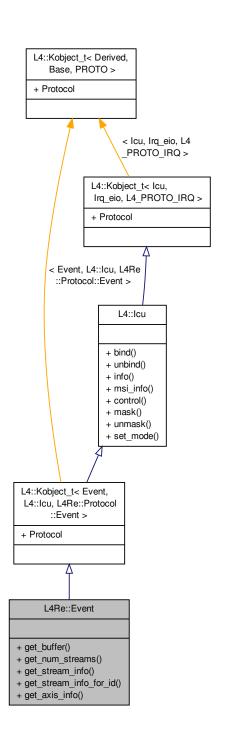
The documentation for this class was generated from the following file:

• I4/re/env

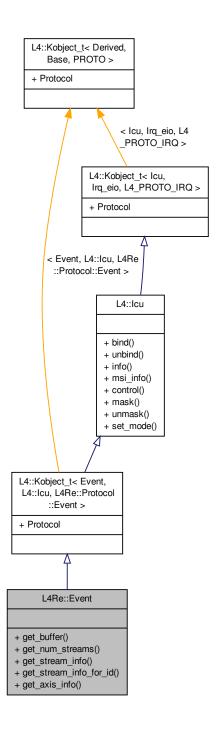
11.57 L4Re::Event Class Reference

Event class.

Inheritance diagram for L4Re::Event:



Collaboration diagram for L4Re::Event:



Public Member Functions

long get_buffer (L4::Cap < Dataspace > ds) const throw ()
 Get event signal buffer.

Additional Inherited Members

11.57.1 Detailed Description

Event class.

Definition at line 103 of file event.

11.57.2 Member Function Documentation

11.57.2.1 long L4Re::Event::get_buffer (L4::Cap < Dataspace > ds) const throw)

Get event signal buffer.

Return values

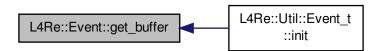
ds	Event buffer.

Returns

0 on success, negative error code otherwise.

Referenced by L4Re::Util::Event_t< PAYLOAD >::init().

Here is the caller graph for this function:



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

• I4/re/event

11.58 L4Re::Event_buffer_t< PAYLOAD >::Event Struct Reference

Event structure used in buffer.

Collaboration diagram for L4Re::Event_buffer_t< PAYLOAD >::Event:

L4Re::Event_buffer
_t< PAYLOAD >::Event
+ time
+ payload
+ free()

Public Member Functions

• void free () throw ()

Free the entry.

Data Fields

· long long time

Event time stamp.

11.58.1 Detailed Description

template<typename PAYLOAD = Default_event_payload>struct L4Re::Event_buffer_t< PAYLOAD >::Event

Event structure used in buffer.

Definition at line 144 of file event.

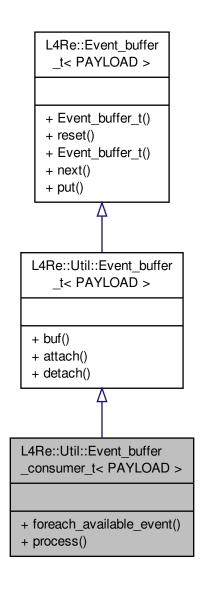
The documentation for this struct was generated from the following file:

• I4/re/event

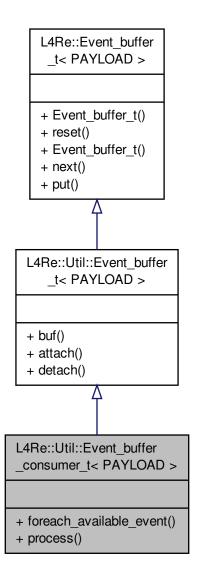
11.59 L4Re::Util::Event_buffer_consumer_t < PAYLOAD > Class Template Reference

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.

11.59.1 Detailed Description

template<typename PAYLOAD>class L4Re::Util::Event_buffer_consumer_t< PAYLOAD>

An event buffer consumer.

Definition at line 91 of file event_buffer.

11.59.2 Member Function Documentation

```
11.59.2.1 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

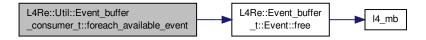
```
cb Function callback.
```

Definition at line 101 of file event_buffer.

References L4Re::Event buffer t< PAYLOAD >::Event::free().

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:



11.59.2.2 template < typename PAYLOAD > template < 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

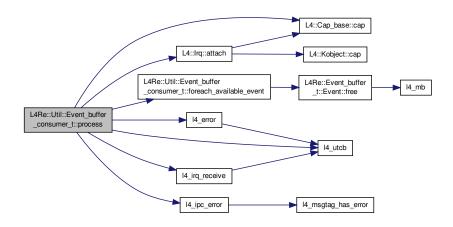
ire	Event signal to wait for.
threa	Thread capability of the thread calling this function.
C	Callback function that is called for each received event.

This function never returns.

Definition at line 121 of file event_buffer.

References L4::lrq::attach(), L4::Cap_base::cap(), L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::foreach_available_event(), I4_error(), I4_ipc_error(), L4_IPC_NEVER, I4_irq_receive(), and I4_utcb().

Here is the call graph for this function:



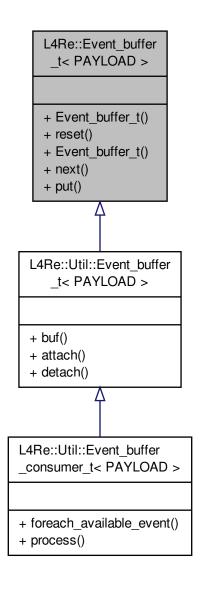
The documentation for this class was generated from the following file:

• I4/re/util/event_buffer

11.60 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 >:

L4Re::Event_buffer _t< PAYLOAD >

- + Event_buffer_t()
- + reset()
- + Event_buffer_t()
- + next()
- + put()

Data Structures

struct Event

Event structure used in buffer.

Public Member Functions

• Event_buffer_t (void *buffer, I4_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.

11.60.1 Detailed Description

template < typename PAYLOAD = Default_event_payload > class L4Re::Event_buffer_t < PAYLOAD >

Event buffer class.

Definition at line 137 of file event.

11.60.2 Constructor & Destructor Documentation

```
11.60.2.1 template<typename PAYLOAD = Default_event_payload> L4Re::Event_buffer_t< PAYLOAD >::Event_buffer_t ( void * buffer, I4_addr_t size ) [inline]
```

Initialize event buffer.

Parameters

buffer	Pointer to buffer.
size	Size of buffer in bytes.

Definition at line 184 of file event.

11.60.3 Member Function Documentation

Next event in buffer.

Returns

0 if no event available, event otherwise.

Definition at line 194 of file event.

References L4Re::Event_buffer_t< PAYLOAD >::Event::time.

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 211 of file event.

References I4_wmb(), and L4Re::Event_buffer_t< PAYLOAD >::Event::time.

Here is the call graph for this function:



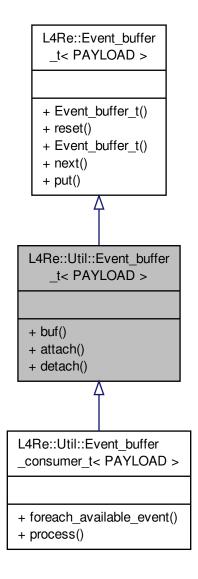
The documentation for this class was generated from the following file:

• I4/re/event

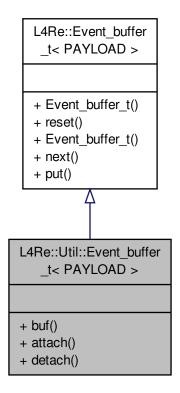
11.61 L4Re::Util::Event_buffer_t < PAYLOAD > Class Template Reference

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.

11.61.1 Detailed Description

 $template < typename\ PAYLOAD > class\ L4Re:: Util:: Event_buffer_t < PAYLOAD >$

Event_buffer utility class.

Definition at line 36 of file event_buffer.

11.61.2 Member Function Documentation

11.61.2.1 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.

```
11.61.2.2 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 L4Re::Rm::Search_addr.

```
11.61.2.3 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 76 of file event_buffer.

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

• I4/re/util/event buffer

11.62 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 >:

```
L4Re::Util::Event_t

< PAYLOAD >

+ init()
+ buffer()
+ irq()
```

Public Types

enum Mode { Mode_irq, Mode_polling }
 Modes of operation.

Public Member Functions

• int init (L4::Cap< L4Re::Event > event, Mode mode=Mode_irq, L4Re::Env const *env=L4Re::Env::env(), L4Re::Cap_alloc *ca=L4Re::Cap_alloc(L4Re::Util::cap_alloc))

Initialise an event object.

L4Re::Event_buffer_t< PAYLOAD > & buffer ()

Get event buffer.

• L4::Cap< L4::Irq> irq () const

Get event IRQ.

11.62.1 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 41 of file event.

11.62.2 Member Enumeration Documentation

11.62.2.1 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 47 of file event.

11.62.3 Member Function Documentation

11.62.3.1 template < typename PAYLOAD > int L4Re::Util::Event_t < PAYLOAD > ::init (L4::Cap < L4Re::Event > event, Mode mode = Mode_irq, 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.

Parameters

	event	Capability to event.
ĺ	env	Optional: Pointer to L4Re-Environment
ĺ	ca	Optional: Pointer to capability allocator.

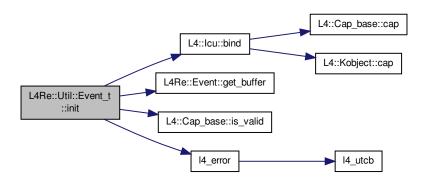
Returns

0 on success, error code on error

Definition at line 61 of file event.

References L4::lcu::bind(), L4Re::Event::get_buffer(), L4::Cap_base::is_valid(), L4_ENOMEM, l4_error(), L4Re::-Util::Event_t< PAYLOAD >::Mode_irq, and L4Re::Rm::Search_addr.

Here is the call graph for this function:



11.62.3.2 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 110 of file event.

11.62.3.3 template < typename PAYLOAD > L4::Cap< L4::Irq> L4Re::Util::Event_t< PAYLOAD >::irq () const [inline]

Get event IRQ.

Returns

Event IRQ.

Definition at line 116 of file event.

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

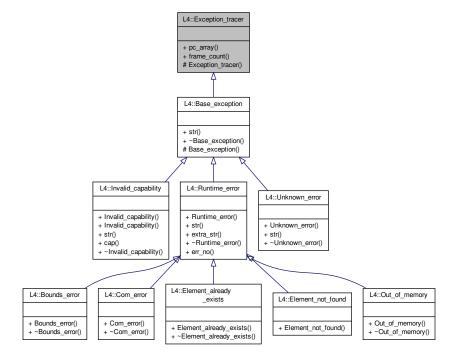
I4/re/util/event

11.63 L4::Exception_tracer Class Reference

Back-trace support for exceptions.

#include <14/cxx/exceptions>

Inheritance diagram for L4::Exception_tracer:



Collaboration diagram for L4::Exception_tracer:

L4::Exception_tracer

- + pc_array()
- + frame_count()
- # 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.

11.63.1 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 64 of file exceptions.

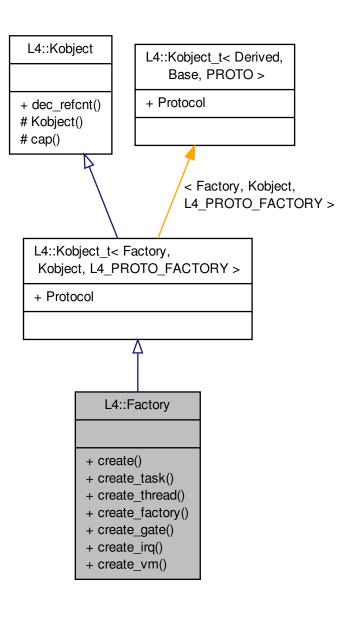
The documentation for this class was generated from the following file:

· I4/cxx/exceptions

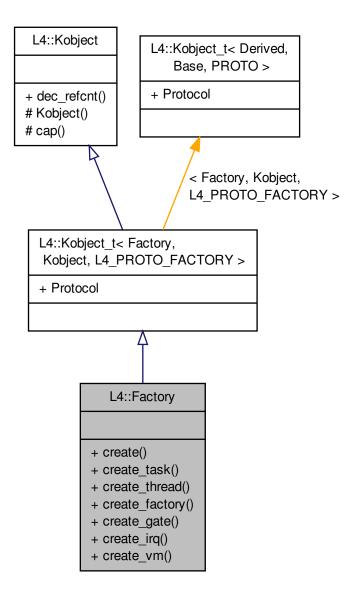
11.64 L4::Factory Class Reference

C++ L4 Factory, to create all kinds of 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< Kobject > target, long obj, I4_utcb_t *utcb=I4_utcb()) throw ()

Generic create call to the factory.

I4_msgtag_t create_task (Cap < Task > const &target_cap, I4_fpage_t const &utcb_area, I4_utcb_t *utcb=I4-utcb()) throw ()

Create a new task.

 $\bullet \ \ \mathsf{I4_msgtag_t} \ \mathsf{create_thread} \ (\mathsf{Cap} < \mathsf{Thread} > \mathsf{const} \ \& \mathsf{target_cap}, \ \mathsf{I4_utcb_t} \ * \mathsf{utcb=I4_utcb}()) \ \mathsf{throw} \ ()$

I4_msgtag_t create_factory (Cap< Factory > const &target_cap, unsigned long limit, I4_utcb_t *utcb=I4_utcb()) throw ()

Create a new factory.

Create a new thread.

I4_msgtag_t create_gate (Cap< Kobject > const &target_cap, Cap< Thread > const &thread_cap, I4_-umword_t label, I4_utcb_t *utcb=I4_utcb()) throw ()

Create a new IPC gate.

- $\bullet \ \ \mathsf{I4_msgtag_t} \ \mathsf{create_irq} \ (\mathsf{Cap} < \mathsf{Irq} > \mathsf{const} \ \& \mathsf{target_cap}, \ \mathsf{I4_utcb_t} \ * \mathsf{utcb=I4_utcb}()) \ \mathsf{throw} \ () \\$
 - Create a new IRQ.
- $\bullet \ \ \mathsf{I4_msgtag_t\ create_vm\ (Cap} < \mathsf{Vm} > \mathsf{const\ \&target_cap}, \ \mathsf{I4_utcb_t\ *utcb=I4_utcb())\ throw\ ()}$

Create a new virtual machine.

Additional Inherited Members

11.64.1 Detailed Description

C++ L4 Factory, to create all kinds of kernel objects.

```
#include <14/sys/factory>
```

See Also

Factory for an overview and C bindings.

Definition at line 41 of file factory.

11.64.2 Member Function Documentation

```
11.64.2.1 S L4::Factory::create ( Cap < Kobject > target, long obj, I4_utcb_t * utcb = I4_utcb () ) throw )
[inline]
```

Generic create call to the factory.

Parameters

target	is the target capability selector where the new object shall be received.
obj	is the protocol ID that specifies which kind of object shall be created.
utcb	is 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:

Definition at line 213 of file factory.

References L4::Kobject::cap().

Referenced by L4Re::Mem_alloc::alloc().

Here is the call graph for this function:



Here is the caller graph for this function:



11.64.2.2 I4_msgtag_t L4::Factory::create_task (Cap < Task > const & target_cap, I4_fpage_t const & utcb_area, I4_utcb_t * utcb = I4_utcb()) throw) [inline]

Create a new task.

Parameters

factory	Capability selector for factory to use for creation.
target_cap	Capability selector for the root capability of the new task.
utcb_area	Flexpage that describes the area for the UTCBs of the new task

Note

The size of the UTCB area specifies indirectly the maximum number of UTCBs available for this task and cannot be changed afterwards.

Returns

Syscall return tag

See Also

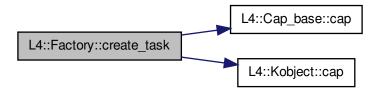
Task

factory is the implicit this pointer.

Definition at line 222 of file factory.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



11.64.2.3 I4_msgtag_t L4::Factory::create_thread (Cap < Thread > const & target_cap, I4_utcb_t * utcb = I4_utcb ()) throw) [inline]

Create a new thread.

Parameters

factory	Capability selector for factory to use for creation.
target_cap	Capability selector for the root capability of the new thread.

Returns

Syscall return tag

See Also

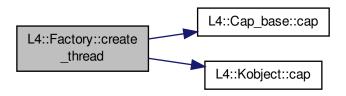
Thread

factory is the implicit this pointer.

Definition at line 231 of file factory.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



11.64.2.4 I4_msgtag_t L4::Factory::create_factory (Cap < Factory > const & target_cap, unsigned long limit, I4_utcb_t * utcb = I4_utcb()) throw) [inline]

Create a new factory.

Parameters

factory	Capability selector for factory to use for creation.
target_cap	Capability selector for the root capability of the new factory.
limit	Limit for the new factory in bytes

Note

The limit of the new factory is subtracted from the available amount of the factory used for creation.

Returns

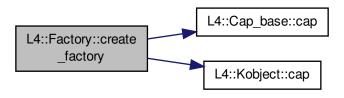
Syscall return tag

factory is the implicit this pointer.

Definition at line 239 of file factory.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



11.64.2.5 I4_msgtag_t L4::Factory::create_gate (Cap< Kobject > const & target_cap, Cap< Thread > const & thread_cap, I4_umword_t label, I4_utcb_t * utcb = I4_utcb ()) throw) [inline]

Create a new IPC gate.

Parameters

factory	Capability selector for factory to use for creation.
target_cap	Capability selector for the root capability of the new IPC gate.
thread_cap	Thread to bind the gate to
label	Label of the gate

Returns

Syscall return tag

See Also

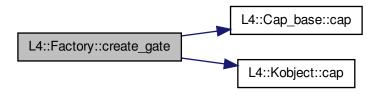
IPC-Gate API

factory is the implicit this pointer.

Definition at line 248 of file factory.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



11.64.2.6 I4_msgtag_t L4::Factory::create_irq (Cap < Irq > const & target_cap, I4_utcb_t * utcb = I4_utcb ()) throw) [inline]

Create a new IRQ.

Parameters

factory	Capability selector for factory to use for creation.
target_cap	Capability selector for the root capability of the new IRQ.

Returns

Syscall return tag

See Also

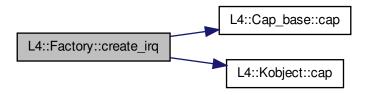
IRQs

factory is the implicit this pointer.

Definition at line 257 of file factory.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



Create a new virtual machine.

Parameters

factory	Capability selector for factory to use for creation.
target_cap	Capability selector for the root capability of the new VM.

Returns

Syscall return tag

See Also

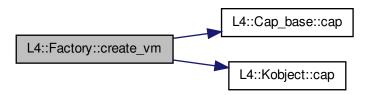
Virtual Machines

factory is the implicit this pointer.

Definition at line 265 of file factory.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



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

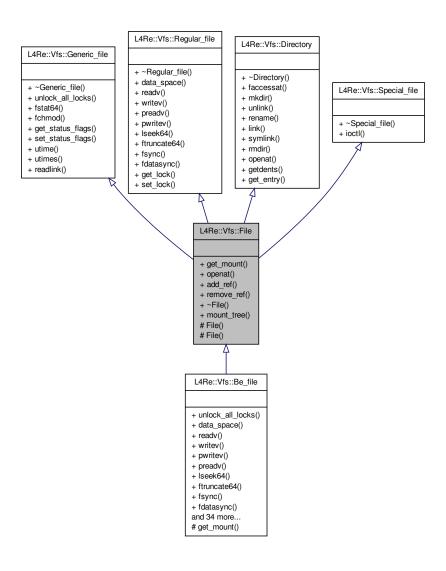
• I4/sys/factory

11.65 L4Re::Vfs::File Class Reference

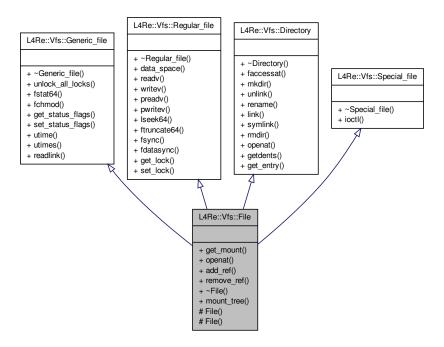
The basic interface for an open POSIX file.

#include < vfs.h >

Inheritance diagram for L4Re::Vfs::File:



Collaboration diagram for L4Re::Vfs::File:



Additional Inherited Members

11.65.1 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 430 of file vfs.h.

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

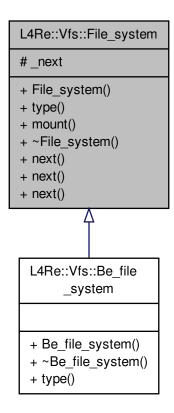
· I4/I4re vfs/vfs.h

11.66 L4Re::Vfs::File_system Class Reference

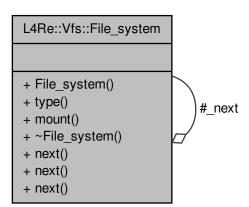
Basic interface for an L4Re::Vfs file system.

#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.

11.66.1 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 827 of file vfs.h.

11.66.2 Member Function Documentation

```
11.66.2.1 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.

```
11.66.2.2 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:



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

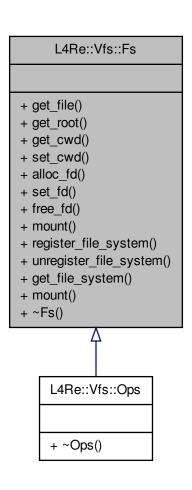
• I4/I4re_vfs/vfs.h

11.67 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:

+ get_file() + get_root() + get_cwd() + set_cwd() + alloc_fd() + set_fd() + free_fd() + mount() + register_file_system() + unregister_file_system() + get_file_system() + mount() + restribe_system() + get_file_system()

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.

 $\bullet \ \ \text{virtual cxx::Ref_ptr} < \ \ \text{File} > \ \ \text{set_fd} \ \ (\text{int fd, cxx::Ref_ptr} < \ \ \text{File} > \ \ \text{const \&f=cxx::Ref_ptr} < > :: \ \ \text{Nil}) = 0 \ \ \text{throw} \ \ ()$

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.

• int mount (char const *path, cxx::Ref_ptr< File > const &dir) 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.

11.67.1 Detailed Description

POSIX File-system related functionality.

This class usually exists as a singleton as a superclass of L4Re::Vfs::Ops (

See Also

L4Re::Vfs::vfs_ops).

Definition at line 879 of file vfs.h.

11.67.2 Member Function Documentation

```
11.67.2.1 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.

```
11.67.2.2 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.

```
11.67.2.3 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.

```
11.67.2.4 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.

11.67.2.5 int L4Re::Vfs::Fs::mount (char const * path, cxx::Ref_ptr< File > const & dir) throw) [inline]

Mount a given file object at the given global path in the VFS.

Parameters

path	The global path to mount <i>dir</i> at.
dir	A pointer to the file/directory object that shall be mounted at path.

Returns

0 on success, or <0 on error.

Definition at line 968 of file vfs.h.

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

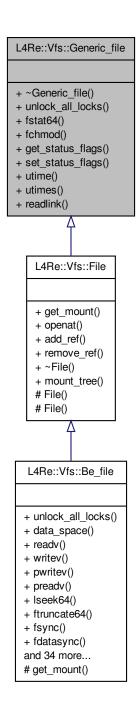
• I4/I4re_vfs/vfs.h

11.68 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:

L4Re::Vfs::Generic_file

- + ~Generic_file()
- + unlock_all_locks()
- + fstat64()
- + fchmod()
- + get_status_flags()
- + set_status_flags()
- + utime()
- + utimes()
- + readlink()

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).

11.68.1 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 63 of file vfs.h.

11.68.2 Member Function Documentation

11.68.2.1 virtual int L4Re::Vfs::Generic_file::unlock_all_locks() throw) [pure virtual]

Unlock all locks on the file.

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.

```
11.68.2.2 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.

Return values

```
buf This buffer is filled with the status information.
```

Returns

```
0 on success, or <0 on error.
```

Implemented in L4Re::Vfs::Be file.

```
11.68.2.3 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.

```
11.68.2.4 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.

```
11.68.2.5 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.

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

• I4/I4re vfs/vfs.h

11.69 gfxbitmap_offset Struct Reference

offsets in pmap[] and bmap[]

#include <bitmap.h>

Collaboration diagram for gfxbitmap_offset:

gfxbitmap_offset

+ preskip_x
+ preskip_y
+ endskip_x

Data Fields

- I4_uint32_t preskip_x
 skip pixels at beginning of line
- I4_uint32_t preskip_y skip lines
- I4_uint32_t endskip_x skip pixels at end of line

11.69.1 Detailed Description

offsets in pmap[] and bmap[]

Definition at line 69 of file bitmap.h.

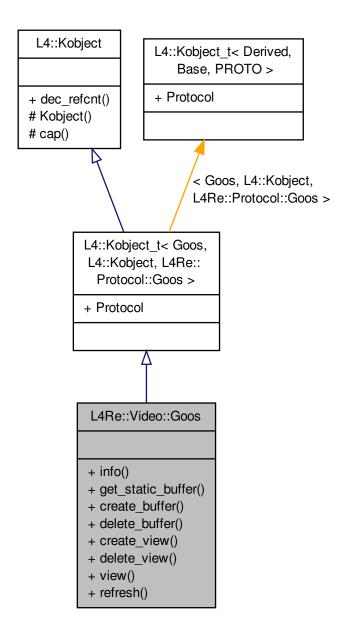
The documentation for this struct was generated from the following file:

• I4/libgfxbitmap/bitmap.h

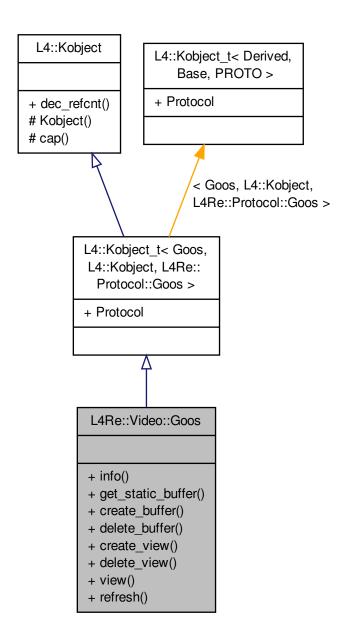
11.70 L4Re::Video::Goos Class Reference

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

• int info (Info *) const throw ()

Return the goos information of the goos.

 $\bullet \ \ \text{int get_static_buffer (unsigned idx, L4::Cap} < L4Re::Dataspace > rbuf) \ const \ throw \ () \\$

Return a static buffer of a goos.

int create_buffer (unsigned long size, L4::Cap< L4Re::Dataspace > rbuf) const throw ()
 Create a buffer.

• int delete_buffer (unsigned idx) const throw ()

Delete a buffer.

int create_view (View *view) const throw ()

Create a view.

int delete_view (View const &v) const throw ()

Delete a view.

View view (unsigned index) const throw ()

Return a view.

int refresh (int x, int y, int w, int h) throw ()

Trigger refreshing of the given area on the virtual screen.

Additional Inherited Members

11.70.1 Detailed Description

A goos.

Definition at line 39 of file goos.

11.70.2 Member Enumeration Documentation

11.70.2.1 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 46 of file goos.

11.70.3 Member Function Documentation

11.70.3.1 int L4Re::Video::Goos::info (Info *) const throw)

Return the goos information of the goos.

Return values

info	Goos information structure pointer.
------	-------------------------------------

Returns

0 on success, error otherwise

11.70.3.2 int L4Re::Video::Goos::get_static_buffer (unsigned idx, L4::Cap < L4Re::Dataspace > rbuf) const throw)

Return a static buffer of a goos.

Parameters

idx	Index of the static buffer.
rbuf	Capability slot to point the buffer dataspace to.

Returns

0 on success, error otherwise

11.70.3.3 int L4Re::Video::Goos::create_buffer (unsigned long size, L4::Cap< L4Re::Dataspace > rbuf) const throw)

Create a buffer.

Parameters

size	Size of buffer in bytes.
rbuf	Capability slot to point the buffer dataspace to.

Returns

Positive: buffer index, negative: Error code

11.70.3.4 int L4Re::Video::Goos::delete_buffer (unsigned idx) const throw)

Delete a buffer.

Parameters

idx	Buffer to delete.

Returns

0 on success, error otherwise

11.70.3.5 int L4Re::Video::Goos::create_view (View * view) const throw)

Create a view.

Return values

view	A view object.

Returns

Positive: view index, negative: Error code

11.70.3.6 int L4Re::Video::Goos::delete_view (View const & v) const throw)

Delete a view.

Parameters

V	The view object to delete.
---	----------------------------

Returns

0 on success, error otherwise

11.70.3.7 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 133 of file goos.

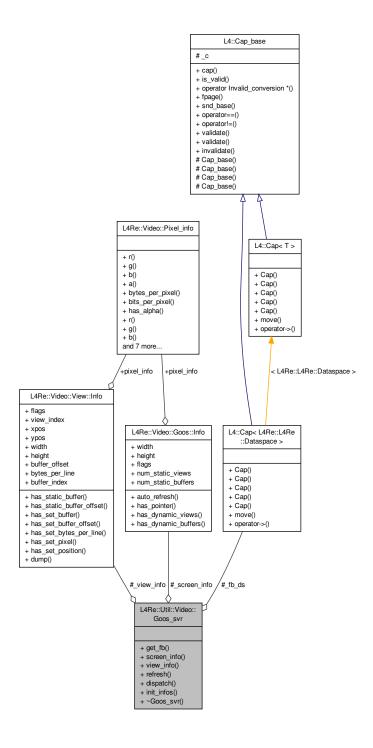
The documentation for this class was generated from the following file:

• I4/re/video/goos

11.71 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.

• int dispatch (I4_umword_t obj, L4::lpc::lostream &ios)

Server dispatch function.

• 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.

11.71.1 Detailed Description

Goos server class.

Definition at line 38 of file goos_svr.

11.71.2 Member Function Documentation

```
11.71.2.1 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.

11.71.2.2 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.

11.71.2.3 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.

11.71.2.4 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)
У	Y coordinate (pixels)
W	Width of area in pixels
h	Height of area in pixels

Returns

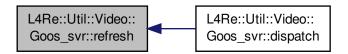
0 on success, negative error code otherwise

Definition at line 77 of file goos_svr.

References L4_ENOSYS.

Referenced by dispatch().

Here is the caller graph for this function:



11.71.2.5 int L4Re::Util::Video::Goos_svr::dispatch (I4_umword_t obj, L4::lpc::lostream & ios) [inline]

Server dispatch function.

Parameters

obj	Server object ID to work on
ios	Input/Output stream.

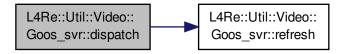
Returns

error code.

Definition at line 121 of file goos_svr.

References _fb_ds, _screen_info, _view_info, L4Re::Protocol::Goos, L4_CAP_FPAGE_RW, L4_EBADPROTO, L4-_ENOSYS, L4_EOK, L4_ERANGE, and refresh().

Here is the call graph for this function:



11.71.2.6 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 98 of file goos_svr.

References _screen_info, _view_info, L4Re::Video::View::Info::buffer_index, L4Re::Video::View::Info::flags, L4-Re::Video::Goos::Info::height, L4Re::Video::View::Info::height, L4Re::Video::Goos::Info::pixel_info, L4Re::Video::View::Info::view_index, L4Re::Video::Goos::Info::width, L4Re::Video::View::Info::width, L4Re::Video::View::Info::view::Info::yiew

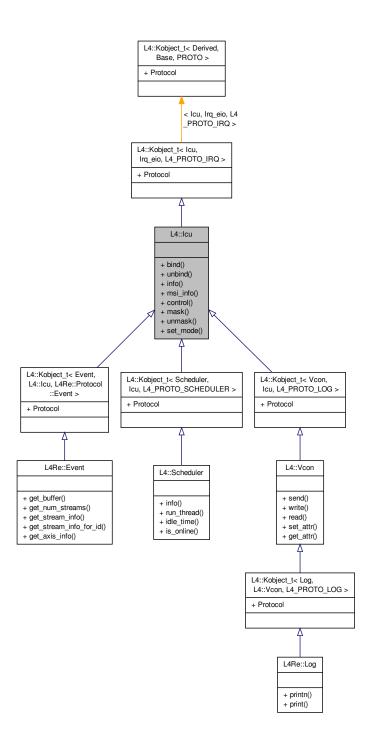
The documentation for this class was generated from the following file:

• I4/re/util/video/goos_svr

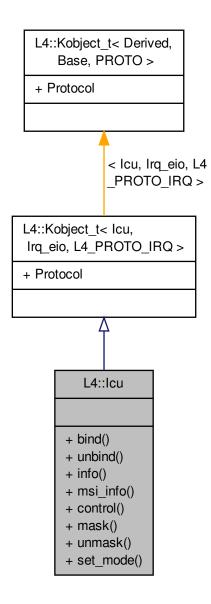
11.72 L4::lcu Class Reference

C++ version of an interrupt controller.

Inheritance diagram for L4::lcu:



Collaboration diagram for L4::lcu:



Data Structures

• class Info
Info for an ICU.

Public Member Functions

- I4_msgtag_t bind (unsigned irqnum, L4::Cap< Irq > irq, I4_utcb_t *utcb=I4_utcb()) throw ()

 Bind an interrupt vector of an interrupt controller to an interrupt object.
- I4_msgtag_t unbind (unsigned irqnum, L4::Cap< Irq > irq, I4_utcb_t *utcb=I4_utcb()) throw ()

 Remove binding of an interrupt vector from the interrupt controller object.

• I4_msgtag_t info (I4_icu_info_t *info, I4_utcb_t *utcb=I4_utcb()) throw ()

Get info about capabilites of ICU.

• I4_msgtag_t msi_info (unsigned irqnum, I4_umword_t *msg, I4_utcb_t *utcb=I4_utcb()) throw ()

Get MSI info about IRQ.

I4_msgtag_t mask (unsigned irqnum, I4_umword_t *label=0, I4_timeout_t to=L4_IPC_NEVER, I4_utcb_t *utcb=I4_utcb()) throw ()

Mask an IRQ vector.

• I4_msgtag_t unmask (unsigned irqnum, I4_umword_t *label=0, I4_timeout_t to=L4_IPC_NEVER, I4_utcb_t *utcb=I4_utcb()) throw ()

Unmask an IRQ vector.

• I4_msgtag_t set_mode (unsigned irqnum, I4_umword_t mode, I4_utcb_t *utcb=I4_utcb()) throw ()

Set mode of interrupt.

Additional Inherited Members

11.72.1 Detailed Description

C++ version of an interrupt controller.

```
#include <14/sys/icu>
```

See Also

Interrupt controller for an overview and C bindings.

Definition at line 125 of file irq.

11.72.2 Member Function Documentation

```
11.72.2.1 I4_msgtag_t L4::lcu::bind ( unsigned irqnum, L4::Cap < Irq > irq, I4_utcb_t * utcb = I4_utcb () ) throw) [inline]
```

Bind an interrupt vector of an interrupt controller to an interrupt object.

Parameters

icu	ICU to use.
irqnum	IRQ vector at the ICU.
irq	IRQ capability to bind the IRQ to.

Returns

Syscall return tag

Note

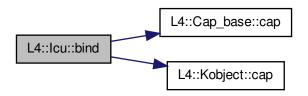
the icu argument is the implicit this pointer.

Definition at line 166 of file irq.

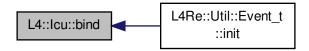
References L4::Cap_base::cap(), and L4::Kobject::cap().

Referenced by L4Re::Util::Event_t< PAYLOAD >::init().

Here is the call graph for this function:



Here is the caller graph for this function:



11.72.2.2 I4_msgtag_t L4::lcu::unbind (unsigned
$$irqnum$$
, L4::Cap < Irq > irq , I4_utcb_t * $utcb$ = I4_utcb ()) throw) [inline]

Remove binding of an interrupt vector from the interrupt controller object.

Parameters

icu	ICU to use.
irqnum	IRQ vector at the ICU.
irq	IRQ object to remove from the ICU.

Returns

Syscall return tag

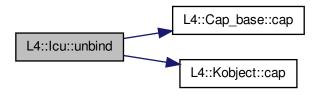
Note

the icu argument is the implicit this pointer.

Definition at line 174 of file irq.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



 $\textbf{11.72.2.3} \quad \textbf{I4_msgtag_t L4::} \\ \textbf{l4_icu_info_t} * \textit{info}, \\ \textbf{I4_utcb_t} * \textit{utcb} = \textbf{I4_utcb_()}) \\ \textbf{throw)} \quad [\texttt{inline}]$

Get info about capabilites of ICU.

Parameters

icu	ICU to use.
info	Pointer to an info structure to be filled with information.

Returns

Syscall return tag

Note

the icu argument is the implicit this pointer.

Definition at line 182 of file irq.

References L4::Kobject::cap().

Here is the call graph for this function:



11.72.2.4 I4_msgtag_t L4::lcu::msi_info (unsigned *irqnum*, I4_umword_t * *msg*, I4_utcb_t * *utcb* = I4_utcb ()) throw) [inline]

Get MSI info about IRQ.

Parameters

icu	ICU to use.
irqnum	IRQ vector at the ICU.
msg	Pointer to a word to receive the message that must be used for the PCI devices MSI message.

Returns

Syscall return tag

Note

the icu argument is the implicit this pointer.

Definition at line 189 of file irq.

References L4::Kobject::cap().

Here is the call graph for this function:



11.72.2.5 I4_msgtag_t L4::lcu::mask(unsigned irqnum, I4_umword_t * label = 0, I4_timeout_t to = L4_IPC_NEVER, I4_utcb_t * utcb = I4_utcb()) throw) [inline]

Mask an IRQ vector.

Parameters

icu	ICU to use.
irqnum	IRQ vector 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

Note

the icu argument is the implicit this pointer.

Definition at line 204 of file irq.

References L4::Kobject::cap().

Here is the call graph for this function:



Unmask an IRQ vector.

Parameters

icu	ICU to use.
irqnum	IRQ vector 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

Note

the icu argument is the implicit this pointer.

Definition at line 214 of file irq.

References L4::Kobject::cap().

Here is the call graph for this function:



11.72.2.7 I4_msgtag_t L4::lcu::set_mode (unsigned irqnum, I4_umword_t mode, I4_utcb_t * utcb = I4_utcb ()) throw) [inline]

Set mode of interrupt.

Parameters

icu	ICU to use.
irqnum	IRQ vector at the ICU.
mode	Mode, see L4_irq_mode.

Returns

Syscall return tag

Note

the icu argument is the implicit this pointer.

Definition at line 224 of file irq.

References L4::Kobject::cap().

Here is the call graph for this function:



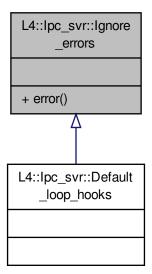
The documentation for this class was generated from the following file:

• I4/sys/irq

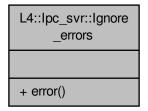
11.73 L4::lpc_svr::lgnore_errors Struct Reference

Mix in for LOOP_HOOKS to ignore IPC errors.

Inheritance diagram for L4::lpc_svr::lgnore_errors:



Collaboration diagram for L4::lpc_svr::lgnore_errors:



11.73.1 Detailed Description

Mix in for LOOP_HOOKS to ignore IPC errors.

Definition at line 61 of file ipc_server.

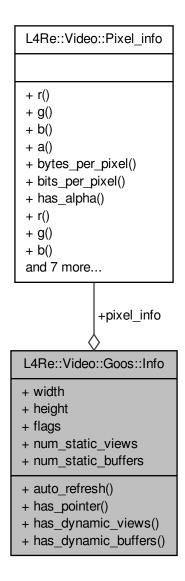
The documentation for this struct was generated from the following file:

• I4/cxx/ipc_server

11.74 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.

11.74.1 Detailed Description

Information structure of a goos.

Definition at line 55 of file goos.

11.74.2 Member Function Documentation

11.74.2.1 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 66 of file goos.

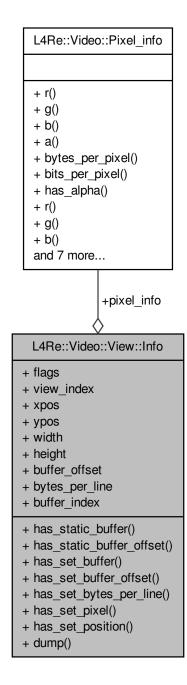
The documentation for this struct was generated from the following file:

• I4/re/video/goos

11.75 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 STREAM >

STREAM & dump (STREAM &s) const

Dump information on the view information to a stream.

Data Fields

· unsigned flags

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.

11.75.1 Detailed Description

Information structure of a view.

Definition at line 85 of file view.

11.75.2 Field Documentation

11.75.2.1 unsigned L4Re::Video::View::Info::flags

Flags,.

See Also

Flags and V_flags

Definition at line 87 of file view.

Referenced by L4Re::Util::Video::Goos_svr::init_infos().

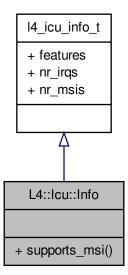
The documentation for this struct was generated from the following file:

• I4/re/video/view

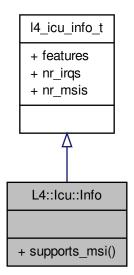
11.76 L4::lcu::lnfo Class Reference

Info for an ICU.

Inheritance diagram for L4::lcu::Info:



Collaboration diagram for L4::lcu::Info:



Additional Inherited Members

11.76.1 Detailed Description

Info for an ICU.

This class adds access functions.

See Also

I4_icu_info().

Definition at line 156 of file irq.

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

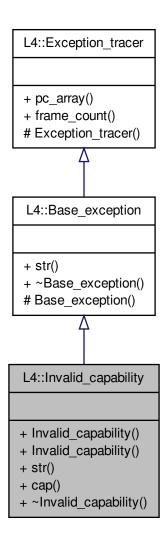
• I4/sys/irq

11.77 L4::Invalid_capability Class Reference

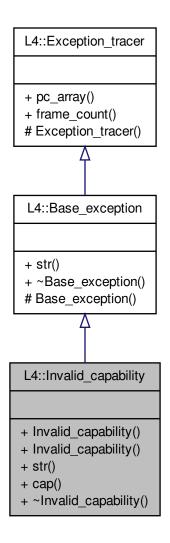
Indicates that an invalid object was invoked.

#include <14/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 ()

Should return a human readable string for the exception.

• Cap< void > const & cap () const throw ()

Get the object that caused the error.

Additional Inherited Members

11.77.1 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 229 of file exceptions.

11.77.2 Constructor & Destructor Documentation

```
11.77.2.1 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 239 of file exceptions.

11.77.3 Member Function Documentation

11.77.3.1 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 248 of file exceptions.

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

• I4/cxx/exceptions

11.78 L4::IOModifier Class Reference

Modifier class for the IO stream.

Collaboration diagram for L4::IOModifier:

+ mod + IOModifier() + operator==()

+ operator!=()

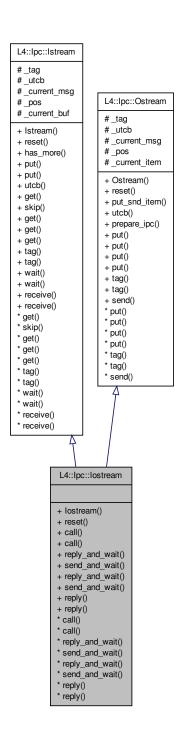
11.78.1 Detailed Description

Modifier class for the IO stream.

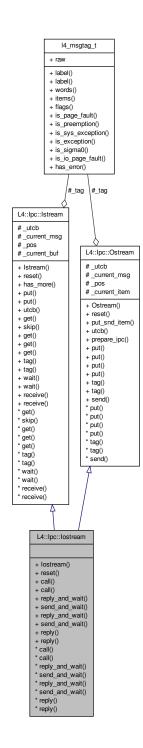
Definition at line 33 of file basic_ostream.	
The documentation for this class was generated from the following file:	
• I4/cxx/basic_ostream	
11.79 L4::lpc::lostream Class Reference	
Input/Output stream for IPC [un]marshalling.	

An IO Modifier can be used to change properties of an IO stream for example the number format.

Inheritance diagram for L4::lpc::lostream:



Collaboration diagram for L4::lpc::lostream:



Public Member Functions

lostream (I4_utcb_t *utcb)

Create an IPC IO stream with a single message buffer.

• void reset ()

Reset the stream to its initial state.

IPC operations.

- I4_msgtag_t call (I4_cap_idx_t dst, I4_timeout_t timeout, long proto=0)
 - Do an IPC call using the message in the output stream and receiving to the input stream.
- I4_msgtag_t call (I4_cap_idx_t dst, long proto=0)
- I4 msgtag t reply and wait (I4 umword t *src dst, long proto=0)

Do an IPC reply and wait.

- I4_msgtag_t send_and_wait (I4_cap_idx_t dest, I4_umword_t *src, long proto=0)
- I4_msgtag_t reply_and_wait (I4_umword_t *src_dst, I4_timeout_t timeout, long proto=0)

Do an IPC reply and wait.

- 14 msgtag t send and wait (14 cap idx t dest, 14 umword t *src, 14 timeout t timeout, long proto=0)
- 14 msgtag t reply (14 timeout t timeout, long proto=0)
- I4_msgtag_t reply (long proto=0)

11.79.1 Detailed Description

Input/Output stream for IPC [un]marshalling.

The lpc::lostream is part of the AW Env IPC framework as well as lpc::lstream and lpc::Ostream. In particular an lpc::lostream is a combination of an lpc::Istream and an lpc::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/clntsrv/client.cc, examples/clntsrv/server.cc, examples/libs/l4re/c++/shared_ds/ds_srv.cc, examples/libs/l4re/streammacc, and examples/libs/l4re/streammap/server.cc.

Definition at line 998 of file ipc_stream.

11.79.2 Constructor & Destructor Documentation

```
11.79.2.1 L4::lpc::lostream(l4_utcb_t * utcb) [inline], [explicit]
```

Create an IPC IO stream with a single message buffer.

Parameters

```
msg 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 1009 of file ipc stream.

11.79.3 Member Function Documentation

```
11.79.3.1 void L4::lpc::lostream::reset() [inline]
```

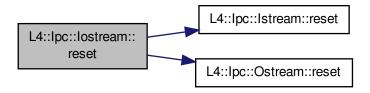
Reset the stream to its initial state.

Input as well as the output stream are reset.

Definition at line 1019 of file ipc stream.

References L4::lpc::lstream::reset(), and L4::lpc::Ostream::reset().

Here is the call graph for this function:



11.79.3.2 I4_msgtag_t L4::lpc::lostream::call (I4_cap_idx_t dst, I4_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:

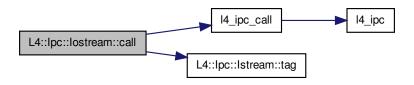
examples/clntsrv/client.cc, and examples/libs/l4re/streammap/client.cc.

Definition at line 1159 of file ipc_stream.

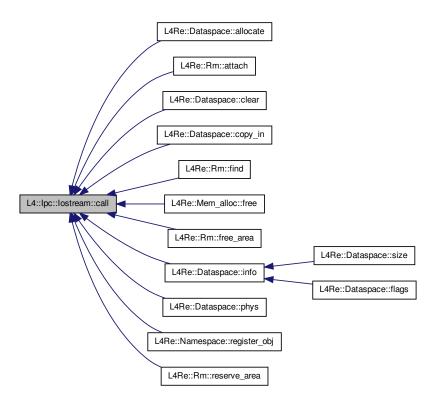
References I4_ipc_call(), and L4::lpc::lstream::tag().

Referenced by L4Re::Dataspace::allocate(), L4Re::Rm::attach(), L4Re::Dataspace::clear(), L4Re::Dataspace::copy_in(), L4Re::Rm::free_area(), L4Re::Dataspace::info(), L4Re::Dataspace::Dataspace::Dataspace::Info(), L4Re::Dataspace::Dataspace::Dataspace::Dataspace::Dataspace

Here is the call graph for this function:



Here is the caller graph for this function:



11.79.3.3 I4_msgtag_t L4::lpc::lostream::reply_and_wait (I4_umword_t * src_dst, long proto = 0) [inline]

Do an IPC reply and wait.

Parameters

src_dst	Input: the destination for the send operation.
	Output: the source of the received message.

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 1076 of file ipc_stream.

References L4_IPC_SEND_TIMEOUT_0.

11.79.3.4 | I4_msgtag_t L4::lpc::lostream::reply_and_wait (| I4_umword_t * src_dst, | I4_timeout_t timeout, | long proto = 0 | [inline]

Do an IPC reply and wait.

Parameters

src_dst	Input: the destination for the send operation.
	Output: the source of the received message.
timeout	Timeout used for IPC.

Returns

the result dope of the IPC operation.

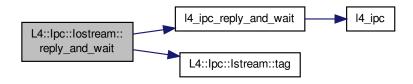
This is a combined IPC operation consisting of a send operation and an open wait for any message.

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 1174 of file ipc_stream.

References I4_ipc_reply_and_wait(), and L4::lpc::lstream::tag().

Here is the call graph for this function:



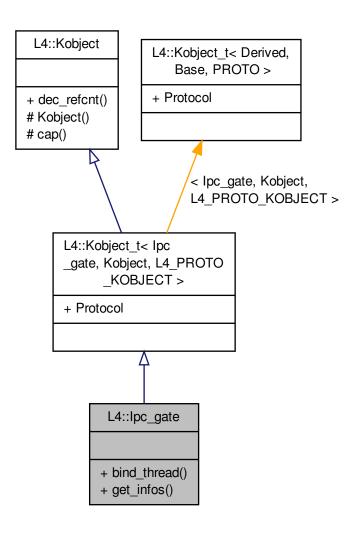
The documentation for this class was generated from the following file:

• I4/cxx/ipc_stream

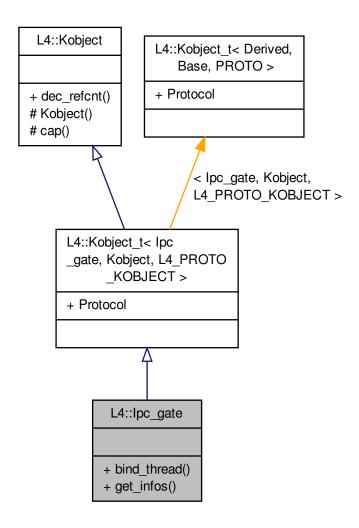
11.80 L4::lpc_gate Class Reference

L4 IPC gate.

Inheritance diagram for L4::lpc_gate:



Collaboration diagram for L4::lpc_gate:



Public Member Functions

- I4_msgtag_t bind_thread (Cap< Thread > t, I4_umword_t label, I4_utcb_t *utcb=I4_utcb()) throw ()

 Bind the IPC-gate to the thread.
- I4_msgtag_t get_infos (I4_umword_t *label, I4_utcb_t *utcb=I4_utcb()) throw () Get information on the IPC-gate.

Additional Inherited Members

11.80.1 Detailed Description

L4 IPC gate.

#include <14/sys/ipc_gate>

Definition at line 39 of file ipc_gate.

11.80.2 Member Function Documentation

```
11.80.2.1 I4_msgtag_t L4::lpc_gate::bind_thread ( Cap < Thread > t, I4_umword_t label, I4_utcb_t * utcb = I4_utcb() ) throw) [inline]
```

Bind the IPC-gate to the thread.

See Also

I4_ipc_gate_bind_thread

Definition at line 50 of file ipc_gate.

References L4::Cap_base::cap().

Here is the call graph for this function:



Get information on the IPC-gate.

See Also

I4_ipc_gate_get_infos

Definition at line 59 of file ipc_gate.

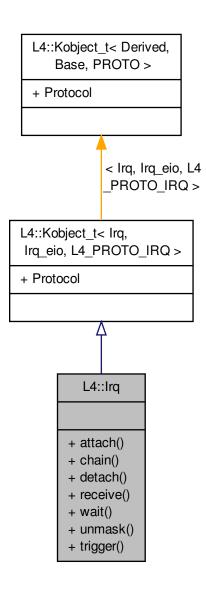
The documentation for this class was generated from the following file:

• I4/sys/ipc_gate

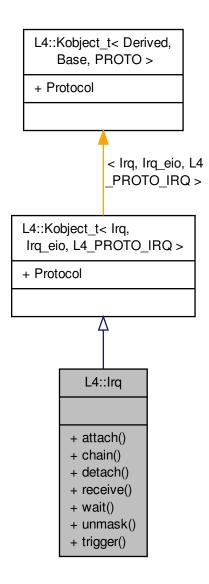
11.81 L4::Irq Class Reference

C++ version of an L4 IRQ.

Inheritance diagram for L4::Irq:



Collaboration diagram for L4::Irq:



Public Member Functions

- I4_msgtag_t attach (I4_umword_t label, Cap< Thread > const &thread, I4_utcb_t *utcb=I4_utcb()) throw ()

 Attach to an interrupt source.
- I4_msgtag_t chain (I4_umword_t label, Cap< Irq > const &slave, I4_utcb_t *utcb=I4_utcb()) throw ()
 Chain an IRQ to another master IRQ source.
- I4_msgtag_t detach (I4_utcb_t *utcb=I4_utcb()) throw ()
 - Detach from an interrupt source.
- I4_msgtag_t receive (I4_timeout_t to=L4_IPC_NEVER, I4_utcb_t *utcb=I4_utcb()) throw ()

 Unmask and wait for specified IRQ.
- I4_msgtag_t wait (I4_umword_t *label, I4_timeout_t to=L4_IPC_NEVER, I4_utcb_t *utcb=I4_utcb()) throw ()

 Unmask IRQ and wait for any message.

I4_msgtag_t unmask (I4_utcb_t *utcb=I4_utcb()) throw ()
 Unmask IRQ.

• I4_msgtag_t trigger (I4_utcb_t *utcb=I4_utcb()) throw ()

Trigger an IRQ.

Additional Inherited Members

11.81.1 Detailed Description

C++ version of an L4 IRQ.

#include <14/sys/irq>

See Also

IRQs for an overview and C bindings.

Examples:

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

Definition at line 54 of file irq.

11.81.2 Member Function Documentation

```
11.81.2.1 I4_msgtag_t L4::lrq::attach ( I4_umword_t label, Cap < Thread > const & thread, I4_utcb_t * utcb = I4_utcb() ) throw) [inline]
```

Attach to an interrupt source.

Parameters

irq	IRQ to attach to.
label	Identifier of the IRQ.
thread	Thread to attach the interrupt to.

Returns

Syscall return tag

Note

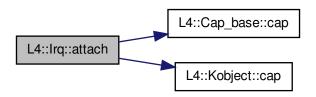
irq is the implicit this pointer.

Definition at line 64 of file irg.

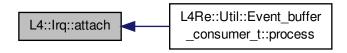
References L4::Cap_base::cap(), and L4::Kobject::cap().

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:



Chain an IRQ to another master IRQ source.

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.

Parameters

irq	The master IRQ object.
label	Identifier of the IRQ.
slave	The slave that shall be attached to the master.

Returns

Syscall return tag

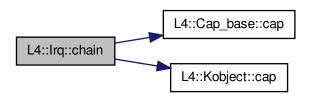
Note

irq is the implicit this pointer.

Definition at line 72 of file irq.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



Detach from an interrupt source.

Parameters

```
irq | IRQ to detach from.
```

Returns

Syscall return tag

Note

irq is the implicit this pointer.

Definition at line 80 of file irq.

References L4::Kobject::cap().

Here is the call graph for this function:



11.81.2.4 I4_msgtag_t L4::lrq::receive (I4_timeout_t to = L4_IPC_NEVER, I4_utcb_t * utcb = I4_utcb ()) throw) [inline]

Unmask and wait for specified IRQ.

Parameters

irq	IRQ to wait for.
to	Timeout.

Returns

Syscall return tag

Note

irq is the implicit this pointer.

Definition at line 88 of file irq.

References L4::Kobject::cap().

Here is the call graph for this function:



11.81.2.5 I4_msgtag_t L4::Irq::wait (I4_umword_t * label, I4_timeout_t to = L4_IPC_NEVER, I4_utcb_t * utcb = I4_utcb()) throw) [inline]

Unmask IRQ and wait for any message.

Parameters

ira	IPO to wait for
114	IRQ to wait for.
label	Receive label.
to	Timeout.

Returns

Syscall return tag

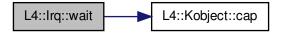
Note

irq is the implicit this pointer.

Definition at line 96 of file irq.

References L4::Kobject::cap().

Here is the call graph for this function:



Unmask IRQ.

Parameters

```
irq | IRQ to unmask.
```

Returns

Syscall return tag

Note

I4_irq_wait and I4_irq_receive are doing the unmask themselves. *irq* is the implicit *this* pointer.

Definition at line 104 of file irq.

References L4::Kobject::cap().

Here is the call graph for this function:



Trigger an IRQ.

Parameters

		_
irq	IRQ to trigger.	

Precondition

irq must be a reference to an IRQ.

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 I4_error() will return an error value from the message tag which still contains the IRQ protocol used for the send operation.

Use I4_ipc_error() to check for (send) errors.

Note

irq is the implicit this pointer.

Examples:

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

Definition at line 111 of file irq.

References L4::Kobject::cap().

Here is the call graph for this function:



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

• I4/sys/irq

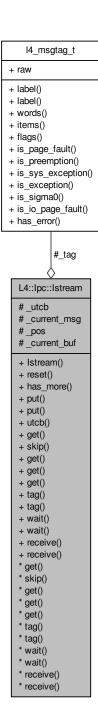
11.82 L4::lpc::lstream Class Reference

Input stream for IPC unmarshalling.

Inheritance diagram for L4::lpc::lstream:



Collaboration diagram for L4::lpc::lstream:



Public Member Functions

Istream (I4_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 ()
        Check whether a value of type T can be obtained from the stream.

    I4_utcb_t * utcb () const
        Return utcb pointer.
```

Get/Put Functions.

These functions are basically used to implement the extraction operators (>>) and should not be called directly. See IPC stream operators.

```
• 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 size)
      Skip size elements of type T in the stream.
template<typename T >
  unsigned long get (Msg_ptr< T > const &buf, unsigned long size=1)
      Read one size elements of type T from the stream and return a pointer.
• template<typename T >
  void get (T &v)
      Extract a single element of type T from the stream.

    bool get (lpc::Varg *va)

    I4_msgtag_t tag () const

      Get the message tag of a received IPC.
• I4_msgtag_t & tag ()
      Get the message tag of a received IPC.
```

IPC operations.

```
    I4_msgtag_t wait (I4_umword_t *src)
        Wait for an incoming message from any sender.
    I4_msgtag_t wait (I4_umword_t *src, I4_timeout_t timeout)
        Wait for an incoming message from any sender.
    I4_msgtag_t receive (I4_cap_idx_t src)
        Wait for a message from the specified sender.
    I4_msgtag_t receive (I4_cap_idx_t src, I4_timeout_t timeout)
```

11.82.1 Detailed Description

Input stream for IPC unmarshalling.

lpc::Istream is part of the dynamic IPC marshalling infrastructure, as well as lpc::Ostream and lpc::lostream.

lpc::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 573 of file ipc stream.

11.82.2 Constructor & Destructor Documentation

```
11.82.2.1 L4::lpc::lstream( I4_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

msg	The message buffer to receive IPC messages.

Definition at line 587 of file ipc_stream.

11.82.3 Member Function Documentation

```
11.82.3.1 void L4::lpc::lstream::reset() [inline]
```

Reset the stream to empty, and ready for receive()/wait().

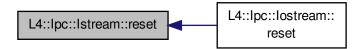
The stream is reset to the same state as on its creation.

Definition at line 597 of file ipc_stream.

References I4_msg_regs_t::mr.

Referenced by L4::lpc::lostream::reset().

Here is the caller graph for this function:



11.82.3.2 template < typename T > unsigned long L4::lpc::lstream::get (T * buf, unsigned long elems) [inline]

Copy out an array of type T with size elements.

Parameters

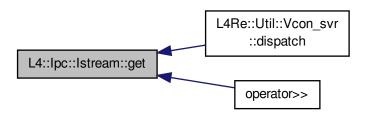
buf	Pointer to a buffer for size elements of type T.
size	number of elements of type T to copy out.

See IPC stream operators .

Definition at line 632 of file ipc_stream.

 $Referenced \ by \ L4Re:: Util:: Vcon_svr < SVR > :: dispatch(), \ and \ operator >> ().$

Here is the caller graph for this function:



11.82.3.3 template<typename T > void L4::lpc::lstream::skip (unsigned long *size*) [inline]

Skip size elements of type T in the stream.

Parameters

_		
	size	number of elements to skip.

Definition at line 650 of file ipc_stream.

11.82.3.4 template<typename T > unsigned long L4::lpc::lstream::get (Msg_ptr< T > const & buf, unsigned long size = 1) [inline]

Read one size elements of type T from the stream and return a pointer.

In contrast to a normal get, this version does actually not copy the data but returns a pointer to the data.

Parameters

buf	a Msg_ptr that is actually set to point to the element in the stream.
size	number of elements to extract (default is 1).

See IPC stream operators .

Definition at line 673 of file ipc_stream.

11.82.3.5 template<typename T > void L4::lpc::lstream::get(T & v) [inline]

Extract a single element of type T from the stream.

Parameters

V	Output: the element.

See IPC stream operators .

Definition at line 693 of file ipc_stream.

11.82.3.6 I4_msgtag_t L4::lpc::lstream::tag()const [inline]

Get the message tag of a received IPC.

Returns

The L4 message tag for the received IPC.

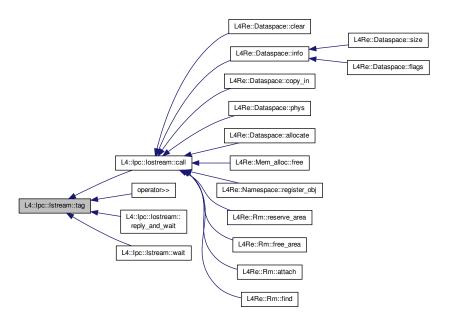
This is in particular useful for handling page faults or exceptions.

See IPC stream operators .

Definition at line 729 of file ipc_stream.

Referenced by L4::lpc::lostream::call(), operator>>(), L4::lpc::lostream::reply_and_wait(), and wait().

Here is the caller graph for this function:



11.82.3.7 I4_msgtag_t& L4::lpc::lstream::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 IPC stream operators .

Definition at line 740 of file ipc_stream.

11.82.3.8 I4_msgtag_t L4::lpc::lstream::wait(I4_umword_t * src) [inline]

Wait for an incoming message from any sender.

Parameters

src	contains the sender after a successful IPC operation.

Returns

The IPC result dope (I4_msgtag_t).

This wait is actually known as 'open wait'.

Definition at line 769 of file ipc_stream.

References L4_IPC_NEVER.

11.82.3.9 I4_msgtag_t L4::lpc::lstream::wait(I4_umword_t * src, I4_timeout_t timeout_) [inline]

Wait for an incoming message from any sender.

Parameters

src	contains the sender after a successful IPC operation.
timeout	Timeout used for IPC.

Returns

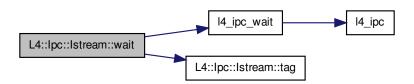
The IPC result dope (I4_msgtag_t).

This wait is actually known as 'open wait'.

Definition at line 1206 of file ipc_stream.

References I4_ipc_wait(), and tag().

Here is the call graph for this function:



11.82.3.10 I4_msgtag_t L4::lpc::lstream::receive(I4_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 (I4_msgtag_t).

This is commonly known as 'closed wait'.

Definition at line 789 of file ipc_stream.

References L4_IPC_NEVER.

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

· I4/cxx/ipc stream

11.83 L4Re::Util::Item_alloc_base Class Reference

Item allocator.

Inherited by L4Re::Util::Item_alloc< Bits >.

Collaboration diagram for L4Re::Util::Item_alloc_base:

+ is_allocated()
+ hint()
+ alloc()
+ free()
+ ltem_alloc_base()
+ alloc()
+ size()

11.83.1 Detailed Description

Item allocator.

Definition at line 38 of file item_alloc.

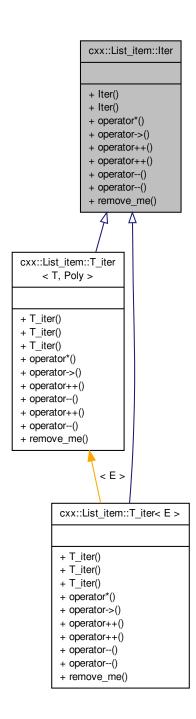
The documentation for this class was generated from the following file:

• I4/re/util/item_alloc

11.84 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:

+ Iter() + Iter() + operator*() + operator->() + operator++() + operator--() + operator--() + remove_me()

Public Member Functions

• List_item * remove_me () throw ()

Remove item pointed to by iterator, and return pointer to element.

11.84.1 Detailed Description

Iterator for a list of ListItem-s.

The Iterator iterates til it finds the first element again.

Definition at line 45 of file list.

11.84.2 Member Function Documentation

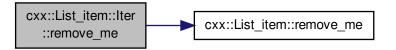
```
11.84.2.1 List_item* cxx::List_item::Iter::remove_me( ) throw) [inline]
```

Remove item pointed to by iterator, and return pointer to element.

Definition at line 86 of file list.

References cxx::List_item::remove_me().

Here is the call graph for this function:



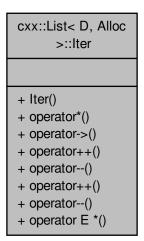
The documentation for this class was generated from the following file:

• I4/cxx/list

11.85 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 (syntactiaclly equal to pointers)

11.85.1 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 356 of file list.

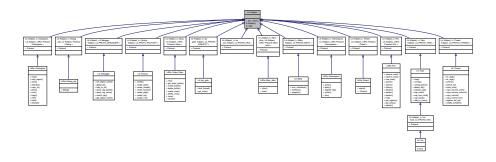
The documentation for this class was generated from the following file:

• I4/cxx/list

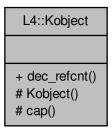
11.86 L4::Kobject Class Reference

Base class for all kinds of kernel objects, referred to by capabilities.

Inheritance diagram for L4::Kobject:



Collaboration diagram for L4::Kobject:



Public Member Functions

• I4_msgtag_t dec_refcnt (I4_mword_t diff, I4_utcb_t *utcb=I4_utcb())

Decrement the in kernel reference counter for the object.

Protected Member Functions

I4_cap_idx_t cap () const throw ()
 Return capability selector.

Friends

```
    template<typename T >

Type_info const * kobject_typeid ()

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

11.86.1 Detailed Description

Base class for all kinds of kernel objects, referred to by capabilities.

```
#include <14/sys/capability>
```

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 454 of file capability.

11.86.2 Member Function Documentation

```
11.86.2.1 | 4 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 480 of file capability.

Referenced by L4::Task::add_ku_mem(), L4::Irq::attach(), L4::Icu::bind(), L4::Task::cap_equal(), L4::Task::cap_has_child(), L4::Task::cap_valid(), L4::Irq::chain(), L4::Thread::control(), 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(), dec_refcnt(), L4::Task::delete_obj(), L4::Irq::detach(), L4::Thread::ex_regs(), L4::Vcon::get_attr(), L4::Debugger::get_object_name(), L4::Debugger::global_id(), L4::Scheduler::idle_time(), L4::Scheduler::info(), L4::Icu::mask(), L4::Task::map(), L4::Icu::mask(), L4::Thread::modify_senders(), L4::Icu::msi_info(), L4::Debugger::query_log_name(), L4::Debugger::query_log_name(), L4::Task::release_cap(), L4::Scheduler::run_thread(), L4::Vcon::send(), L4::Vcon::set_attr(), L4::Icu::set_mode(), L4::Debugger::set_object_name(), L4::Thread::stats_time(), L4::Debugger::switch_log(), L4::Thread::switch_to(), L4::Irq::trigger(), L4::Icu::unbind(), L4::Task::unmap(), L4::Task::unmap_batch(), L4::Irq::unmask(), L4::Irq::wait(), and L4::Vcon::write().

```
11.86.2.2 | 14 msqtaq t L4::Kobject::dec refcnt ( | 14 mword t diff, | 14 utcb t * utcb = | 14 utcb () ) [inline]
```

Decrement the in kernel reference counter for the object.

Parameters

diff	is the delta that shall be subtracted from the reference count.
utcb	is the utcb to use for the invocation.

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 505 of file capability.

References cap().

Here is the call graph for this function:



11.86.3 Friends And Related Function Documentation

11.86.3.1 template < typename T > Type_info const* kobject_typeid() [friend]

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

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 87 of file __typeinfo.h.

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

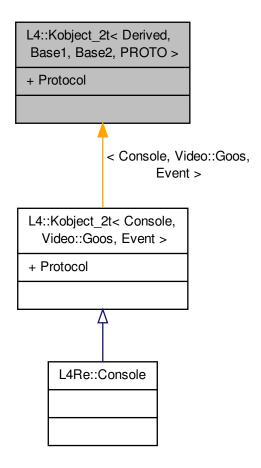
· I4/sys/capability

11.87 L4::Kobject_2t < Derived, Base1, Base2, PROTO > Class Template Reference

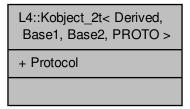
Helper class to create an L4Re interface class that is derived from two base classes.

#include <__typeinfo.h>

Inheritance diagram for L4::Kobject_2t< Derived, Base1, Base2, PROTO >:



Collaboration diagram for L4::Kobject_2t< Derived, Base1, Base2, PROTO >:



Friends

 $\bullet \ \ template {<} typename \ T >$

```
Type_info const * kobject_typeid ()

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

11.87.1 Detailed Description

 $template < typename \ Base1, \ typename \ Base2, \ long \ PROTO = 0 > class \ L4:: Kobject_2t < Derived, \ Base1, \ Base2, \ PROTO >$

Helper class to create an L4Re interface class that is derived from two base classes.

Parameters 4 8 1

Derived	is the name of the new interface.
Base1	is the name of the interfaces first base class.
Base2	is the name of the interfaces second base class.
PROTO	may be set to the statically assigned protocol number used to communicate with this interface.

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::lcu and L4Re::Dataspace.

```
* class My_iface : public L4::Kobject_2t<My_iface, L4::Icu, L4Re::Dataspace>
* {
* ...
* };
```

Definition at line 175 of file __typeinfo.h.

11.87.2 Friends And Related Function Documentation

11.87.2.1 template<typename Derived, typename Base1, typename Base2, long PROTO = 0> template<typename T > Type info const* kobject_typeid() [friend]

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

Parameters

Τ	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 87 of file __typeinfo.h.

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

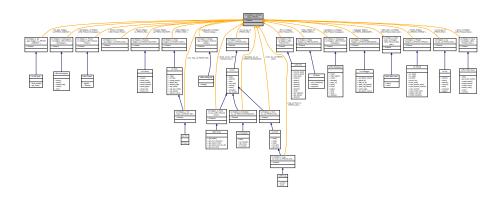
• I4/sys/__typeinfo.h

11.88 L4::Kobject_t < Derived, Base, PROTO > Class Template Reference

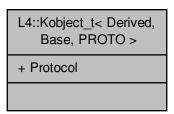
Helper class to create an L4Re interface class that is derived from a single base class.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Kobject_t< Derived, Base, PROTO >:



Collaboration diagram for L4::Kobject_t< Derived, Base, PROTO >:



Friends

```
    template<typename T >

Type_info const * kobject_typeid ()

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

11.88.1 Detailed Description

template<typename Derived, typename Base, long PROTO = 0>class L4::Kobject_t< Derived, Base, PROTO >

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

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.

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 137 of file __typeinfo.h.

11.88.2 Friends And Related Function Documentation

```
11.88.2.1 template<typename Derived, typename Base, long PROTO = 0> template<typename T > Type_info const* kobject_typeid( ) [friend]
```

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

Parameters

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 87 of file __typeinfo.h.

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

· I4/sys/__typeinfo.h

11.89 I4_buf_regs_t Struct Reference

Encapsulation of the buffer-registers block in the UTCB.

#include <14/sys/utcb.h>

Collaboration diagram for I4_buf_regs_t:



Data Fields

• I4_umword_t bdr

Buffer descriptor.

I4_umword_t br [L4_UTCB_GENERIC_BUFFERS_SIZE]

Buffer registers.

11.89.1 Detailed Description

Encapsulation of the buffer-registers block in the UTCB.

Definition at line 96 of file utcb.h.

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

• I4/sys/utcb.h

11.90 I4_exc_regs_t Struct Reference

UTCB structure for exceptions.

#include <utcb.h>

Collaboration diagram for I4_exc_regs_t:

+ pfa + err + tpidruro + r + sp + ulr + _dummy1 + pc + cpsr + r15 and 28 more...

Data Fields

```
I4_umword_t pfa
page fault address
```

• I4_umword_t err

error code

• I4_umword_t tpidruro

Thread-ID register.

• I4_umword_t r [13]

registers

I4_umword_t sp

stack pointer

I4_umword_t ulr

```
ulr
• I4_umword_t _dummy1
     dummy
• I4_umword_t pc

    I4_umword_t cpsr

     cpsr
• I4_umword_t r15
     r15
• I4_umword_t r14
     r14
• I4_umword_t r13
     r13
• I4_umword_t r12
     r12
• I4_umword_t r11
     r11
• I4_umword_t r10
     r10
• I4_umword_t r9
• I4_umword_t r8
• I4_umword_t rdi
• I4_umword_t rsi
     rsi
• I4_umword_t rbp
     rbp
• I4_umword_t rbx
     rbx
• I4_umword_t rdx
     rdx
• I4_umword_t rcx
     rcx
• I4_umword_t rax
     rax
• I4_umword_t trapno
     trap number
• I4_umword_t ip
     instruction pointer
• I4_umword_t dummy1
     dummy

    I4_umword_t flags

     rflags
• I4_umword_t ss
     stack segment register
• I4_umword_t gs
     gs register
• I4_umword_t fs
     fs register
```

```
I4_umword_t edi
edi register
I4_umword_t esi
esi register
I4_umword_t ebp
ebp register
I4_umword_t ebx
ebx register
I4_umword_t edx
edx register
I4_umword_t ecx
ecx register
I4_umword_t ecx
ecx register
I4_umword_t eax
```

11.90.1 Detailed Description

eax register

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 58 of file utcb.h.

```
11.90.2 Field Documentation
```

```
11.90.2.1 I4_umword_t I4_exc_regs_t::flags
```

rflags

eflags

Definition at line 80 of file utcb.h.

The documentation for this struct was generated from the following files:

- · arm/l4/sys/utcb.h
- · amd64/l4/sys/utcb.h
- x86/l4/sys/utcb.h

11.91 I4_fpage_t Union Reference

```
L4 flexpage type.
```

```
#include <__14_fpage.h>
```

Collaboration diagram for I4_fpage_t:

l4_fpage_t + fpage + raw

Data Fields

• I4_umword_t fpage

Raw value.

• I4_umword_t raw

Raw value.

11.91.1 Detailed Description

L4 flexpage type.

Definition at line 78 of file __l4_fpage.h.

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

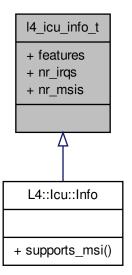
• I4/sys/__I4_fpage.h

11.92 I4_icu_info_t Struct Reference

Info structure for an ICU.

#include <icu.h>

Inheritance diagram for I4_icu_info_t:



Collaboration diagram for I4_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,.

11.92.1 Detailed Description

Info structure for an ICU.

This structure contains information about the features of an ICU.

See Also

```
I4_icu_info().
```

Definition at line 153 of file icu.h.

11.92.2 Field Documentation

11.92.2.1 unsigned I4_icu_info_t::features

Feature flags.

If L4_ICU_FLAG_MSI is set the ICU supports MSIs.

Definition at line 160 of file icu.h.

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

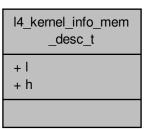
• I4/sys/icu.h

11.93 I4_kernel_info_mem_desc_t Struct Reference

Memory descriptor data structure.

```
#include <memdesc.h>
```

Collaboration diagram for I4_kernel_info_mem_desc_t:



11.93.1 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 64 of file memdesc.h.

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

· I4/sys/memdesc.h

11.94 I4_kernel_info_t Struct Reference

L4 Kernel Interface Page.

```
#include <__kip-32bit.h>
```

Collaboration diagram for I4_kernel_info_t:

I4_kernel_info_t

- + magic
- + version
- + offset_version_strings
- + fill0
- + kip_sys_calls
- + fill1
- + scheduler_granularity
- + _res00
- + sigma0_esp
- + sigma0_eip
- and 28 more...

Data Fields

• I4_uint32_t magic

Kernel Info Page

identifier ("L4μΚ").

• I4_uint32_t version

Kernel version.

• I4_uint8_t offset_version_strings

offset to version string

• I4_uint8_t fill0 [3]

reserved

• I4_uint8_t kip_sys_calls

pointer to system calls

• I4_uint8_t fill1 [3]

```
reserved

    I4_umword_t scheduler_granularity

     for rounding time slices

    I4_umword_t _res00 [3]

     default_kdebug_end
• I4_umword_t sigma0_esp
     Sigma0 start stack pointer.
• I4_umword_t sigma0_eip
     Sigma0 instruction pointer.

    I4_umword_t _res01 [2]

     reserved

    I4_umword_t sigma1_esp

     Sigma1 start stack pointer.

    I4_umword_t sigma1_eip

     Sigma1 instruction pointer.

    I4_umword_t _res02 [2]

     reserved
• I4_umword_t root_esp
     Root task stack pointer.
• I4_umword_t root_eip
     Root task instruction pointer.

    I4_umword_t _res03 [2]

     reserved

    I4_umword_t _res50 [1]

     reserved
• I4_umword_t mem_info
     memory information

    I4_umword_t _res58 [2]

     reserved

    I4_umword_t _res04 [16]

     reserved

    I4_umword_t _res05 [2]

     reserved
• I4_umword_t frequency_cpu
     CPU frequency in kHz.
• I4_umword_t frequency_bus
     Bus frequency.

    I4_umword_t _res06 [10]

     reserved
· 14 umword t user ptr
     user_ptr

    I4_umword_t vhw_offset

     offset to vhw structure

    I4_uint64_t magic

                                      Kernel Info Page
     identifier ("L4μK").
• I4_uint64_t version
     Kernel version.
• 14 uint8 t fill2 [7]
     reserved
```

• I4_uint8_t fill3 [7]

reserved

I4_umword_t _res_a0 [1]

reserved

I4_umword_t _res_b0 [2]

reserver

11.94.1 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:

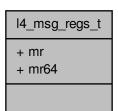
- · I4/sys/__kip-32bit.h
- I4/sys/__kip-64bit.h

11.95 I4_msg_regs_t Union Reference

Encapsulation of the message-register block in the UTCB.

#include <14/sys/utcb.h>

Collaboration diagram for I4_msg_regs_t:



Data Fields

- I4_umword_t mr [L4_UTCB_GENERIC_DATA_SIZE]
 - Message registers.
- I4_uint64_t mr64 [L4_UTCB_GENERIC_DATA_SIZE/(sizeof(I4_uint64_t)/sizeof(I4_umword_t))]
 Message registers 64bit alias.

11.95.1 Detailed Description

Encapsulation of the message-register block in the UTCB.

Examples:

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

Definition at line 80 of file utcb.h.

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

· I4/sys/utcb.h

11.96 I4_msgtag_t Struct Reference

Message tag data structure.

#include <types.h>

Collaboration diagram for I4_msgtag_t:

14_msgtag_t + raw + label() + label() + words() + items() + flags() + is_page_fault() + is_preemption() + is_sys_exception() + is_exception() + is_sigma0() + is_io_page_fault() + has_error()

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

• I4_mword_t raw

raw value

11.96.1 Detailed Description

Message tag data structure.

```
#include <14/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/clntsrv/server.cc, 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 158 of file types.h.

11.96.2 Member Function Documentation

```
11.96.2.1 unsigned I4_msgtag_t::flags() const throw) [inline]
```

Get the flags value.

The flags are a combination of the flags defined by I4 msgtag flags.

Definition at line 176 of file types.h.

References raw.

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

· I4/sys/types.h

11.97 I4_sched_cpu_set_t Struct Reference

CPU sets.

#include <scheduler.h>

Collaboration diagram for I4_sched_cpu_set_t:

I4_sched_cpu_set_t
+ offset
+ map
+ granularity

Data Fields

• I4_umword_t offset

First CPU of interest (must be aligned to 2\(^\)granularity).

• I4_umword_t map

Bitmap of online CPUs.

· unsigned char granularity

One bit in map represents 2\(^{\text{granularity CPUs}}.\)

11.97.1 Detailed Description

CPU sets.

Examples:

examples/sys/migrate/thread_migrate.cc.

Definition at line 40 of file scheduler.h.

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

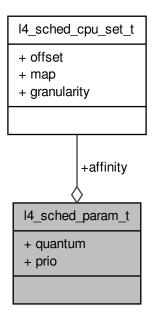
• I4/sys/scheduler.h

11.98 I4_sched_param_t Struct Reference

Scheduler parameter set.

#include <scheduler.h>

Collaboration diagram for I4_sched_param_t:



Data Fields

- I4_cpu_time_t quantum
 - Timeslice in micro seconds.
- unsigned prio
 - Priority for scheduling.
- I4_sched_cpu_set_t affinity CPU affinity.

11.98.1 Detailed Description

Scheduler parameter set.

Examples:

examples/sys/aliens/main.c, examples/sys/migrate/thread_migrate.cc, examples/sys/start-with-exc/main.c, and examples/sys/utcb-ipc/main.c.

examples/sys/singlestep/main.c,

Definition at line 101 of file scheduler.h.

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

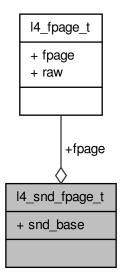
· I4/sys/scheduler.h

11.99 I4_snd_fpage_t Struct Reference

Send-flex-page types.

#include <__l4_fpage.h>

Collaboration diagram for I4 snd fpage t:



Data Fields

• I4_umword_t snd_base

Offset in receive window (send base)

• I4_fpage_t fpage

Source flex-page descriptor.

11.99.1 Detailed Description

Send-flex-page types.

Definition at line 95 of file __l4_fpage.h.

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

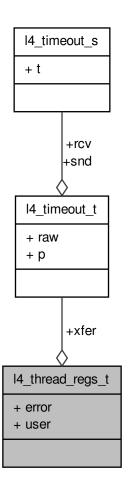
• I4/sys/__I4_fpage.h

11.100 I4_thread_regs_t Struct Reference

Encapsulation of the thread-control-register block of the UTCB.

#include <14/sys/utcb.h>

Collaboration diagram for I4_thread_regs_t:



Data Fields

• I4_umword_t error

System call error codes.

• I4_timeout_t xfer

Message transfer timeout.

• I4_umword_t user [3]

User values (ignored and preserved by the kernel)

11.100.1 Detailed Description

Encapsulation of the thread-control-register block of the UTCB.

Definition at line 114 of file utcb.h.

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

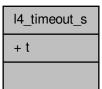
• I4/sys/utcb.h

11.101 I4_timeout_s Struct Reference

Basic timeout specification.

#include <__timeout.h>

Collaboration diagram for I4_timeout_s:



Data Fields

• I4_uint16_t t

timeout value

11.101.1 Detailed Description

Basic timeout specification.

Basically a floating point number with 10 bits mantissa and 5 bits exponent ($t = m*2^{\circ}e$).

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:

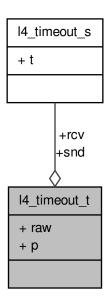
• I4/sys/__timeout.h

11.102 I4_timeout_t Union Reference

Timeout pair.

#include <__timeout.h>

Collaboration diagram for I4_timeout_t:



Data Fields

combined timeout

11.102.1 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:

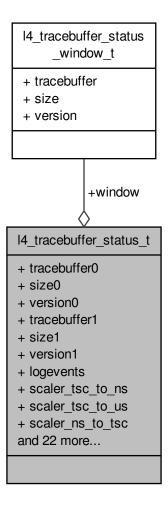
• I4/sys/__timeout.h

11.103 I4_tracebuffer_status_t Struct Reference

Trace buffer status.

#include <ktrace.h>

Collaboration diagram for I4 tracebuffer status t:



Data Fields

• I4_umword_t tracebuffer0

Address of trace buffer 0.

• I4_umword_t size0

Size of trace buffer 0.

I4_umword_t version0

Version number of trace buffer 0 (incremented if tb0 overruns)

• I4_umword_t tracebuffer1

Address of trace buffer 1 (there is no gap between tb0 and tb1)

• I4_umword_t size1

Size of trace buffer 1 (same as tb0)

• I4_umword_t version1

Version number of trace buffer 1 (incremented if tb1 overruns)

• I4_umword_t logevents [16]

Available LOG events.

I4_umword_t scaler_tsc_to_ns

Scaler used for translation of CPU cycles to nano seconds.

I4_umword_t scaler_tsc_to_us

Scaler used for translation of CPU cycles to micro seconds.

I4_umword_t scaler_ns_to_tsc

Scaler used for translation of nano seconds to CPU cycles.

I4_umword_t cnt_context_switch

Number of context switches (intra AS or inter AS)

· I4 umword t cnt addr space switch

Number of inter AS context switches.

• I4_umword_t cnt_shortcut_failed

How often was the IPC shortcut not taken.

· I4 umword t cnt shortcut success

How often was the IPC shortcut taken.

I4_umword_t cnt_irq

Number of hardware interrupts (without kernel scheduling interrupt)

• I4_umword_t cnt_ipc_long

Number of long IPCs.

• I4_umword_t cnt_page_fault

Number of page faults.

• 14 umword 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)

I4_umword_t cnt_task_create

Number of tasks created.

• I4_umword_t schedule

Number of reschedules.

volatile I4_tracebuffer_entry_t * current_entry

Address of the most current event in trace-buffer.

· volatile I4 umword t cnt context switch

Number of context switches (intra AS or inter AS)

volatile I4_umword_t cnt_addr_space_switch

Number of inter AS context switches.

volatile l4_umword_t cnt_shortcut_failed

How often was the IPC shortcut taken.

volatile I4_umword_t cnt_shortcut_success

How often was the IPC shortcut not taken.

volatile l4_umword_t cnt_irq

Number of hardware interrupts (without kernel scheduling interrupt)

volatile l4_umword_t cnt_ipc_long

Number of long IPCs.

volatile l4_umword_t cnt_page_fault

Number of page faults.

volatile I4_umword_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 in the I/O bitmap)

· volatile l4 umword t cnt task create

Number of tasks created.

volatile I4_umword_t cnt_schedule

Number of reschedules.

· volatile I4 umword t cnt iobmap tlb flush

Number of flushes of the I/O bitmap.

11.103.1 Detailed Description

Trace buffer status.

Trace-buffer status.

Tracebuffer status.

Definition at line 67 of file ktrace.h.

11.103.2 Field Documentation

11.103.2.1 I4_umword_t I4_tracebuffer_status_t::tracebuffer0

Address of trace buffer 0.

Address of tracebuffer 0.

Definition at line 70 of file ktrace.h.

11.103.2.2 I4_umword_t I4_tracebuffer_status_t::size0

Size of trace buffer 0.

Size of tracebuffer 0.

Definition at line 72 of file ktrace.h.

11.103.2.3 I4 umword t I4_tracebuffer_status_t::version0

Version number of trace buffer 0 (incremented if tb0 overruns)

Version number of tracebuffer 0 (incremented if tb0 overruns)

Definition at line 74 of file ktrace.h.

11.103.2.4 I4_umword_t I4_tracebuffer_status_t::tracebuffer1

Address of trace buffer 1 (there is no gap between tb0 and tb1)

Address of tracebuffer 1 (there is no gap between tb0 and tb1)

Definition at line 76 of file ktrace.h.

11.103.2.5 I4_umword_t I4_tracebuffer_status_t::size1

Size of trace buffer 1 (same as tb0)

Size of tracebuffer 1 (same as tb0)

Definition at line 78 of file ktrace.h.

11.103.2.6 I4_umword_tl4_tracebuffer_status_t::version1

Version number of trace buffer 1 (incremented if tb1 overruns)

Version number of tracebuffer 1 (incremented if tb1 overruns)

Definition at line 80 of file ktrace.h.

11.103.2.7 volatile I4_umword_t I4_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 99 of file ktrace.h.

The documentation for this struct was generated from the following files:

- · arm/l4/sys/ktrace.h
- · amd64/l4f/l4/sys/ktrace.h
- · x86/l4/sys/ktrace.h

11.104 I4_tracebuffer_status_window_t Struct Reference

Trace-buffer status window descriptor.

#include <ktrace.h>

Collaboration diagram for I4_tracebuffer_status_window_t:

I4_tracebuffer_status
__window_t

+ tracebuffer
+ size
+ version

Data Fields

• I4_tracebuffer_entry_t * tracebuffer

Address of trace-buffer.

• I4_umword_t size

Size of trace-buffer.

• volatile I4_uint64_t version

Version number of trace-buffer (incremented if trace-buffer overruns)

11.104.1 Detailed Description

Trace-buffer status window descriptor.

Definition at line 45 of file ktrace.h.

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

x86/l4/sys/ktrace.h

11.105 I4_vcon_attr_t Struct Reference

Vcon attribute structure.

#include <vcon.h>

Collaboration diagram for I4_vcon_attr_t:



Data Fields

• I4_umword_t i_flags

input flags

• I4_umword_t o_flags

output flags

• I4_umword_t l_flags

local flags

11.105.1 Detailed Description

Vcon attribute structure.

Definition at line 115 of file vcon.h.

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

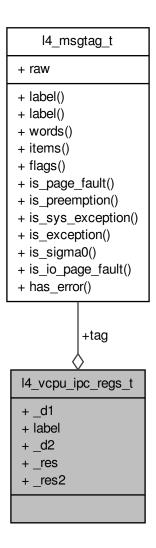
• I4/sys/vcon.h

11.106 I4_vcpu_ipc_regs_t Struct Reference

vCPU message registers.

#include <__vcpu-arch.h>

Collaboration diagram for I4_vcpu_ipc_regs_t:



11.106.1 Detailed Description

vCPU message registers.

Definition at line 46 of file __vcpu-arch.h.

The documentation for this struct was generated from the following files:

- arm/l4/sys/__vcpu-arch.h
- amd64/l4/sys/__vcpu-arch.h
- x86/l4/sys/__vcpu-arch.h

11.107 I4_vcpu_regs_t Struct Reference

vCPU registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for I4_vcpu_regs_t:

+ pfa + err + tpidruro + r + sp + Ir + _dummy + ip + flags + r15 and 21 more...

Data Fields

```
• I4_umword_t pfa
     page fault address
• I4_umword_t err
     error code
• I4_umword_t sp
     stack pointer
• I4_umword_t ip
     instruction pointer
• I4_umword_t flags
     eflags
• I4_umword_t r15
     r15 register
• I4_umword_t r14
     r14 register
• I4_umword_t r13
     r13 register
• I4_umword_t r12
     r12 register
• I4_umword_t r11
     r11 register
• I4_umword_t r10
     r10 register
• I4_umword_t r9
```

r9 register

• I4_umword_t r8

```
r8 reigster
    • I4_umword_t di
         rdi register
    • I4_umword_t si
         rsi register
    • I4_umword_t bp
         rbp register
    • I4_umword_t bx
         rbx register
    • I4_umword_t dx
         rdx register

    I4_umword_t cx

         rcx register

    I4_umword_t ax

         rax register

    I4_umword_t trapno

         trap number
    • I4_umword_t dummy1
         dummy
    • I4_umword_t ss
         ss register
    • I4_umword_t es
         gs register
    • I4_umword_t ds
         fs register
    • I4_umword_t gs
         gs register

    I4_umword_t fs

         fs register
11.107.1
          Detailed Description
vCPU registers.
Definition at line 27 of file __vcpu-arch.h.
11.107.2 Field Documentation
rdi register
edi register
Definition at line 44 of file __vcpu-arch.h.
rsi register
esi register
Definition at line 45 of file __vcpu-arch.h.
```

```
rbp register
ebp register
Definition at line 46 of file __vcpu-arch.h.
rbx register
ebx register
Definition at line 48 of file __vcpu-arch.h.
rdx register
edx register
Definition at line 49 of file __vcpu-arch.h.
rcx register
ecx register
Definition at line 50 of file __vcpu-arch.h.
rax register
eax register
Definition at line 51 of file __vcpu-arch.h.
The documentation for this struct was generated from the following files:
   • arm/l4/sys/__vcpu-arch.h
```

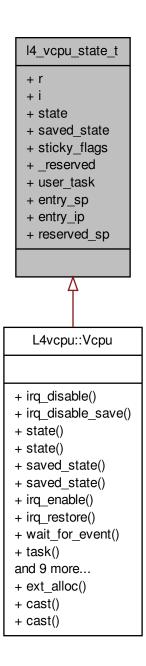
- amd64/l4/sys/__vcpu-arch.h
- x86/l4/sys/__vcpu-arch.h

11.108 I4_vcpu_state_t Struct Reference

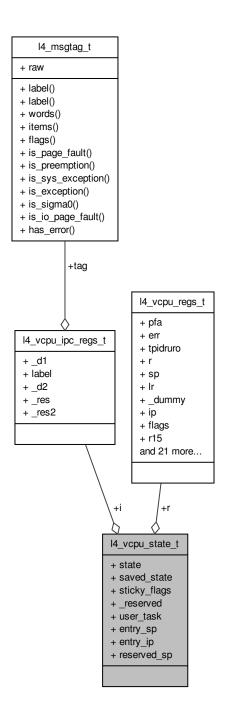
State of a vCPU.

#include <vcpu.h>

Inheritance diagram for I4_vcpu_state_t:



Collaboration diagram for I4_vcpu_state_t:



Data Fields

• I4_vcpu_regs_t r

Register state.

• I4_vcpu_ipc_regs_t i

IPC state.

• I4_uint16_t state

Current vCPU state.

• I4_uint16_t saved_state

Saved vCPU state.

• I4_uint16_t sticky_flags

Pending flags.

• I4_cap_idx_t user_task

User task to use.

• I4_umword_t entry_sp

Stack pointer for entry (when coming from user task)

• I4_umword_t entry_ip

IP for entry.

11.108.1 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:

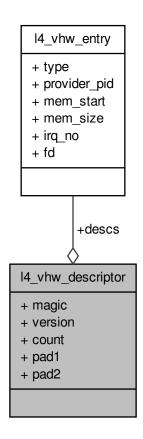
• I4/sys/vcpu.h

11.109 I4_vhw_descriptor Struct Reference

Virtual hardware devices description.

#include <vhw.h>

Collaboration diagram for I4_vhw_descriptor:



Data Fields

• I4_uint32_t magic

Magic.

• I4_uint8_t version

Version of the descriptor.

• I4_uint8_t count

Number of entries.

I4_uint8_t pad1

padding

I4_uint8_t pad2

padding

struct I4_vhw_entry descs []

Array of device descriptions.

11.109.1 Detailed Description

Virtual hardware devices description.

```
Examples:
```

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

Definition at line 70 of file vhw.h.

11.109.2 Field Documentation

11.109.2.1 I4_uint32_t I4_vhw_descriptor::magic

Magic.

Examples:

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

Definition at line 71 of file vhw.h.

11.109.2.2 I4_uint8_t I4_vhw_descriptor::version

Version of the descriptor.

Examples:

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

Definition at line 72 of file vhw.h.

11.109.2.3 I4_uint8_t I4_vhw_descriptor::count

Number of entries.

Examples:

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

Definition at line 73 of file vhw.h.

11.109.2.4 struct I4_vhw_entry I4_vhw_descriptor::descs[]

Array of device descriptions.

Definition at line 77 of file vhw.h.

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

• I4/sys/vhw.h

11.110 I4_vhw_entry Struct Reference

Description of a device.

#include <vhw.h>

Collaboration diagram for I4_vhw_entry:

I4_vhw_entry

- + type
- + provider_pid
- + mem_start
- + mem_size
- + irq_no
- + fd

Data Fields

• enum l4_vhw_entry_type type

Type of virtual hardware.

I4_uint32_t provider_pid

Host PID of the VHW provider.

• I4_addr_t mem_start

Start of memory region.

I4_addr_t mem_size

Size of memory region.

• I4_uint32_t irq_no

IRQ number.

• I4_uint32_t fd

File descriptor.

11.110.1 Detailed Description

Description of a device.

Examples:

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

Definition at line 55 of file vhw.h.

11.110.2 Field Documentation

11.110.2.1 enum I4_vhw_entry_type I4_vhw_entry::type

Type of virtual hardware.

Examples:

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

11.110 I4_vhw_entry Struct Reference Definition at line 56 of file vhw.h. 11.110.2.2 I4_uint32_t I4_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. 11.110.2.3 I4_addr_t I4_vhw_entry::mem_start Start of memory region. **Examples:** examples/sys/ux-vhw/main.c. Definition at line 59 of file vhw.h. 11.110.2.4 I4_addr_t I4_vhw_entry::mem_size Size of memory region. **Examples:** examples/sys/ux-vhw/main.c. Definition at line 60 of file vhw.h. IRQ number. **Examples:** examples/sys/ux-vhw/main.c. Definition at line 62 of file vhw.h.

File descriptor.

Definition at line 63 of file vhw.h.

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

· I4/sys/vhw.h

11.111 I4_vm_svm_vmcb_control_area Struct Reference

VMCB structure for SVM VMs.

#include <__vm-svm.h>

Collaboration diagram for I4_vm_svm_vmcb_control_area:

14_vm_svm_vmcb_control_area

- + intercept_rd_crX
- + intercept_wr_crX
- + intercept_rd_drX
- + intercept_wr_drX
- + intercept_exceptions
- + intercept_instruction0
- + intercept_instruction1
- + _reserved0
- + iopm_base_pa
- + msrpm_base_pa and 14 more...

11.111.1 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:

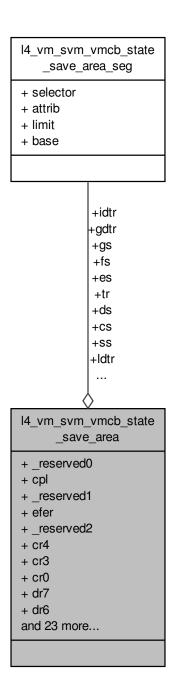
• I4/sys/__vm-svm.h

11.112 I4_vm_svm_vmcb_state_save_area Struct Reference

State save area structure for SVM VMs.

#include <__vm-svm.h>

Collaboration diagram for I4_vm_svm_vmcb_state_save_area:



11.112.1 Detailed Description

State save area structure for SVM VMs.

Definition at line 91 of file __vm-svm.h.

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

• I4/sys/__vm-svm.h

11.113 I4_vm_svm_vmcb_state_save_area_seg Struct Reference

State save area segment selector struct.

#include <__vm-svm.h>

Collaboration diagram for I4_vm_svm_vmcb_state_save_area_seg:

I4_vm_svm_vmcb_state
_save_area_seg

+ selector
+ attrib
+ limit
+ base

11.113.1 Detailed Description

State save area segment selector struct.

Definition at line 79 of file __vm-svm.h.

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

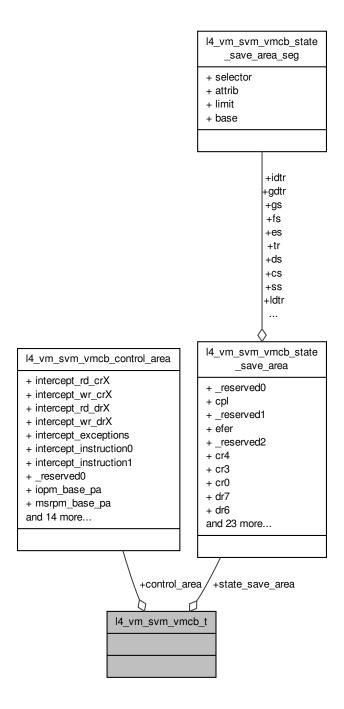
• I4/sys/__vm-svm.h

11.114 I4_vm_svm_vmcb_t Struct Reference

Control structure for SVM VMs.

#include <__vm-svm.h>

Collaboration diagram for I4_vm_svm_vmcb_t:



11.114.1 Detailed Description

Control structure for SVM VMs.

Definition at line 156 of file __vm-svm.h.

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

• I4/sys/__vm-svm.h

state structure for TrustZone VMs

#include <vm.h>

Collaboration diagram for I4_vm_tz_state:

+ r + sp_usr + lr_usr + irq + r_fiq + fiq + abt + und + svc + pc and 8 more...

11.115.1 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

Auxiliary descriptor.

#include <14aux.h>

Collaboration diagram for l4re_aux_t:

+ binary + kip_ds + dbg_lvl + ldr_flags

Data Fields

char const * binary

Binary name.

• I4_cap_idx_t kip_ds

Data space of the KIP.

• I4_umword_t dbg_lvl

Debug levels for l4re.

• I4_umword_t ldr_flags

Flags for l4re, see l4re_aux_ldr_flags_t.

11.116.1 Detailed Description

Auxiliary descriptor.

Definition at line 51 of file l4aux.h.

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

• I4/re/I4aux.h

11.117 | I4re_ds_stats_t Struct Reference

Information about the data space.

#include <dataspace.h>

Collaboration diagram for I4re_ds_stats_t:

l4re_ds_stats_t
+ size
+ flags

Data Fields

• unsigned long size

size

• unsigned long flags

flags

11.117.1 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:

• I4/re/c/dataspace.h

11.118 | I4re_elf_aux_mword_t Struct Reference

Auxiliary vector element for a single unsigned data word.

#include <elf_aux.h>

Collaboration diagram for I4re_elf_aux_mword_t:

| I4re_elf_aux_mword_t | + type | + length | + value

11.118.1 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:

• I4/re/elf_aux.h

11.119 | I4re_elf_aux_t Struct Reference

Generic header for each auxiliary vector element.

#include <elf_aux.h>

Collaboration diagram for l4re_elf_aux_t:

| l4re_elf_aux_t | + type | + length

11.119.1 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:

• I4/re/elf_aux.h

11.120 | I4re_elf_aux_vma_t Struct Reference

Auxiliary vector element for a reserved virtual memory area.

#include <elf_aux.h>

Collaboration diagram for I4re_elf_aux_vma_t:

+ type + length + start + end

11.120.1 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:

• I4/re/elf_aux.h

11.121 I4re_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:

l4re_env_cap_entry_t
+ cap + flags + name
+ I4re_env_cap_entry_t() + I4re_env_cap_entry_t() + is_valid_name()

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

• I4_cap_idx_t cap

The capability selector for the obeject.

I4_umword_t flags

Some flags for the object.

• char name [16]

The name of the object.

11.121.1 Detailed Description

Entry in the L4Re environment array for the named inital objects.

Definition at line 35 of file env.h.

11.121.2 Constructor & Destructor Documentation

```
11.121.2.1 | I4re_env_cap_entry_t::I4re_env_cap_entry_t ( char const * n, I4_cap_idx_t c, I4_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.
С	is the capability selector that refers the initial object.
f	are the additional flags for the object.

Definition at line 67 of file env.h.

References name.

11.121.3 Field Documentation

11.121.3.1 I4_umword_t l4re_env_cap_entry_t::flags

Some flags for the object.

Note

Currently unused.

Definition at line 46 of file env.h.

Referenced by I4re_env_get_cap_I().

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

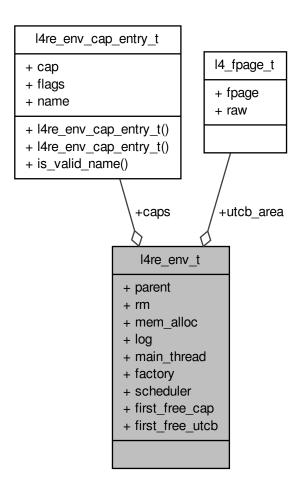
• I4/re/env.h

11.122 | I4re_env_t Struct Reference

Initial Environment structure (C version)

#include <env.h>

Collaboration diagram for I4re_env_t:



Data Fields

• I4_cap_idx_t parent

Parent object-capability.

• I4_cap_idx_t rm

Region map object-capability.

• I4_cap_idx_t mem_alloc

Memory allocator object-capability.

• I4_cap_idx_t log

Logging object-capability.

• I4_cap_idx_t main_thread

Object-capability of the first user thread.

• I4_cap_idx_t factory

Object-capability of the factory available to the task.

• I4_cap_idx_t scheduler

Object capability for the scheduler set to use.

I4_cap_idx_t first_free_cap

First capability index available to the application.

• I4_fpage_t utcb_area

UTCB area of the task.

• I4_addr_t first_free_utcb

First UTCB within the UTCB area available to the application.

11.122.1 Detailed Description

Initial Environment structure (C version)

See Also

Initial environment

Definition at line 96 of file env.h.

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

· I4/re/env.h

11.123 | I4re_event_t Struct Reference

Event structure used in buffer.

#include <event.h>

Collaboration diagram for I4re_event_t:

+ time + type + code + value + stream_id

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.

• I4_umword_t stream_id

Stream ID.

11.123.1 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:

• I4/re/c/event.h

11.124 | I4re_video_color_component_t Struct Reference

Color component structure.

#include <colors.h>

Collaboration diagram for I4re_video_color_component_t:

l4re_video_color_component_t
+ size
+ shift

Data Fields

· unsigned char size

Size in bits.

unsigned char shift

offset in pixel

11.124.1 Detailed Description

Color component structure.

Definition at line 29 of file colors.h.

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

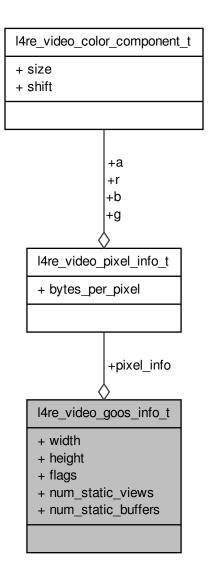
• I4/re/c/video/colors.h

11.125 | I4re_video_goos_info_t Struct Reference

Goos information structure.

#include <goos.h>

Collaboration diagram for I4re_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.

• 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.

11.125.1 Detailed Description

Goos information structure.

Definition at line 51 of file goos.h.

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

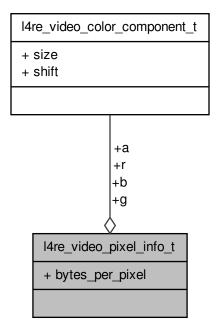
• I4/re/c/video/goos.h

11.126 | I4re_video_pixel_info_t Struct Reference

Pixel_info structure.

#include <colors.h>

Collaboration diagram for l4re_video_pixel_info_t:



Data F	ields
--------	-------

• l4re_video_color_component_t a

Colors.

• unsigned char bytes_per_pixel

Bytes per pixel.

11.126.1 Detailed Description

Pixel_info structure.

Definition at line 39 of file colors.h.

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

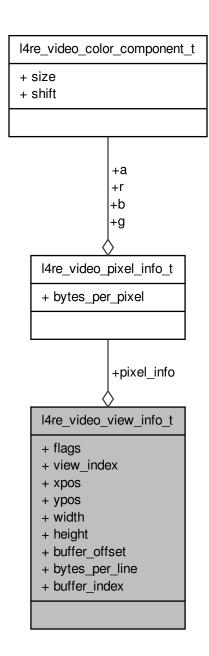
• I4/re/c/video/colors.h

11.127 I4re_video_view_info_t Struct Reference

View information structure.

#include <view.h>

Collaboration diagram for I4re_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.

11.127.1 Detailed Description

View information structure.

Definition at line 59 of file view.h.

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

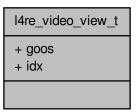
· I4/re/c/video/view.h

11.128 | I4re_video_view_t Struct Reference

C representation of a goos view.

#include <view.h>

Collaboration diagram for l4re_video_view_t:



11.128.1 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:

• I4/re/c/video/view.h

IDT entry.

#include <idt.h>

Collaboration diagram for I4util_idt_desc_t:



Data Fields

• I4_uint64_t b

see Intel doc

• I4_uint32_t b

see Intel doc

11.129.1 Detailed Description

IDT entry.

Definition at line 33 of file idt.h.

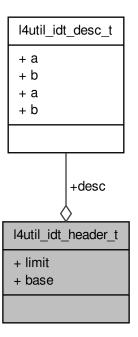
The documentation for this struct was generated from the following files:

- · amd64/l4/util/idt.h
- x86/l4/util/idt.h

Header of an IDT table.

#include <idt.h>

Collaboration diagram for I4util_idt_header_t:



Data Fields

• I4_uint16_t limit

limit field (see Intel doc)

void * base

idt base (see Intel doc)

11.130.1 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 files:

- · amd64/l4/util/idt.h
- x86/l4/util/idt.h

11.131 | I4util_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 I4util_mb_addr_range_t:

+ struct_size + addr + size + type

Data Fields

- I4_uint64_t addr
 - <Size of structure
- I4_uint64_t size
 - < Start address
- I4_uint32_t type
 - <Size of memory range

11.131.1 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 43 of file mb_info.h.

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

• I4/util/mb_info.h

11.132 | I4util_mb_apm_t Struct Reference

APM BIOS info.

#include <mb_info.h>

Collaboration diagram for I4util_mb_apm_t:

+ version + cseg + offset + cseg_16 + dseg_16 + cseg_len + cseg_16_len + dseg_16_len

11.132.1 Detailed Description

APM BIOS info.

Definition at line 91 of file mb_info.h.

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

• I4/util/mb_info.h

11.133 | I4util_mb_drive_t Struct Reference

Drive Info structure.

#include <mb_info.h>

Collaboration diagram for I4util_mb_drive_t:

I4util_mb_drive_t

- + size
- + drive number
- + drive_mode
- + drive_cylinders
- + drive_heads
- + drive_sectors
- + drive_ports

Data Fields

- I4_uint8_t drive_number
 - < The size of this structure.
- I4_uint8_t drive_mode
 - < The BIOS drive number.
- I4_uint16_t drive_cylinders
 - < The access mode (see below).
- I4_uint8_t drive_heads
 - < number of cylinders
- I4_uint8_t drive_sectors
 - < number of heads
- I4_uint16_t drive_ports [0]
 - < number of sectors per track

11.133.1 Detailed Description

Drive Info structure.

Definition at line 74 of file mb_info.h.

11.133.2 Field Documentation

11.133.2.1 I4_uint8_t I4util_mb_drive_t::drive_number

<The size of this structure.

Definition at line 77 of file mb_info.h.

11.133.2.2 I4_uint8_t I4util_mb_drive_t::drive_mode

<The BIOS drive number.

Definition at line 78 of file mb info.h.

11.133.2.3 I4_uint16_t I4util_mb_drive_t::drive_cylinders

<The access mode (see below).

Definition at line 79 of file mb_info.h.

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

• I4/util/mb_info.h

11.134 | I4util_mb_info_t Struct Reference

#include <mb_info.h>

Collaboration diagram for I4util_mb_info_t:

I4util_mb_info_t

- + flags
- + mem_lower
- + mem_upper
- + boot_device
- + cmdline
- + mods count
- + mods_addr
- + tabsize
- + strsize
- + addr
- and 20 more...

Data Fields

• I4_uint32_t flags

MultiBoot info version number.

- I4_uint32_t mem_lower
 - available memory below 1MB
- I4_uint32_t mem_upper

available memory starting from 1MB [kB]

• I4_uint32_t boot_device

"root" partition

• I4_uint32_t cmdline

Kernel command line.

• I4_uint32_t mods_count

number of modules

• I4_uint32_t mods_addr

module list

• I4_uint32_t mmap_length

size of memory mapping buffer

• I4_uint32_t mmap_addr

address of memory mapping buffer

I4_uint32_t drives_length

size of drive info buffer

• I4_uint32_t drives_addr

address of driver info buffer

• I4_uint32_t config_table

ROM configuration table.

• I4_uint32_t boot_loader_name

Boot Loader Name.

• I4_uint32_t apm_table

APM table.

• 14 uint32 t vbe ctrl info

VESA video contoller info.

• I4_uint32_t vbe_mode_info

VESA video mode info.

• I4_uint16_t vbe_mode

VESA video mode number.

• I4_uint16_t vbe_interface_seg

VESA segment of prot BIOS interface.

• I4_uint16_t vbe_interface_off

VESA offset of prot BIOS interface.

• I4_uint16_t vbe_interface_len

VESA lenght of prot BIOS interface.

• I4_uint32_t tabsize

(a.out) Kernel symbol table info

• 14_uint32_t num

(ELF) Kernel section header table

11.134.1 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 203 of file mb_info.h.

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

• I4/util/mb_info.h

11.135 | I4util_mb_mod_t Struct Reference

```
#include <mb info.h>
```

Collaboration diagram for I4util_mb_mod_t:

+ mod_start + mod end

- + cmdline
- + pad

Data Fields

• I4_uint32_t mod_start

Starting address of module in memory.

I4_uint32_t mod_end

End address of module in memory.

• I4_uint32_t cmdline

Module command line.

• I4_uint32_t pad

padding to take it to 16 bytes

11.135.1 Detailed Description

The structure type "mod_list" is used by the multiboot_info structure.

Definition at line 27 of file mb_info.h.

11.135.2 Field Documentation

Starting address of module in memory.

Definition at line 29 of file mb_info.h.

End address of module in memory.

Definition at line 30 of file mb_info.h.

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

• I4/util/mb_info.h

11.136 | I4util_mb_vbe_ctrl_t Struct Reference

VBE controller information.

#include <mb_info.h>

Collaboration diagram for I4util_mb_vbe_ctrl_t:

I4util_mb_vbe_ctrl_t

- + signature
- + version
- + oem_string
- + capabilities
- + video_mode
- + total_memory
- + oem_software_rev
- + oem_vendor_name
- + oem_product_name
- + oem_product_rev
- + reserved
- + oem_data

11.136.1 Detailed Description

VBE controller information.

Definition at line 105 of file mb_info.h.

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

• I4/util/mb_info.h

11.137 | I4util_mb_vbe_mode_t Struct Reference

VBE mode information.

#include <mb_info.h>

Collaboration diagram for I4util_mb_vbe_mode_t:

I4util_mb_vbe_mode_t + mode_attributes + win_a_attributes + win_b_attributes + win_granularity + win_size - reserved1 - reversed2 - linear_bytes_per_scanline - banked_number_of_image _pages + linear_number_of_image _position + linear_blue_mask_size + linear_blue_field_position + linear_reserved_mask_size + linear_reserved_field _position + max_pixel_clock + reserved3 mode attributes
win a, attributes
win a, attributes
win b, attributes
win granularity
win size
win a, segment
win func
bytes per scanline
x_resolution
y_resolution
x_char_size
y_char_size
number_of planes
bits_per_pixel
number_of banks
memory_model
bank_size
number_of lands
preserved
to did_position
tered mask_size
red_field_position
blue_mask_size
green_field_position
blue_mask_size
reserved_field_position
fiered_rodor_mode_inlo
phys_base
reserved_field_position
fiered_rodor_mode_inlo
phys_base

Data Fields

all VESA versions

- I4_uint16_t mode_attributes
- I4_uint8_t win_a_attributes
- I4_uint8_t win_b_attributes
- I4_uint16_t win_granularity

```
• I4_uint16_t win_size
   • 14_uint16_t win_a_segment
   • 14 uint16 t win b segment
   • 14 uint32 t win func

    I4 uint16 t bytes per scanline

>= VESA version 1.2
   • I4_uint16_t x_resolution
   • I4_uint16_t y_resolution
   • 14 uint8 t x char size
   • 14 uint8 t y char size
   • I4_uint8_t number_of_planes
   • I4_uint8_t bits_per_pixel
   • 14_uint8_t number_of_banks
   • 14 uint8 t memory model
   • 14 uint8 t bank size
   • I4_uint8_t number_of_image_pages
   • I4_uint8_t reserved0
direct color
   • 14 uint8 t red mask size

    I4 uint8 t red field position

   • I4_uint8_t green_mask_size
   • 14_uint8_t green_field_position
   • 14_uint8_t blue_mask_size
   • 14 uint8 t blue field position

    I4 uint8 t reserved mask size

   • I4_uint8_t reserved_field_position
   • 14_uint8_t direct_color_mode_info
>= VESA version 2.0
   • 14 uint32 t phys base
   • 14 uint32 t reserved1
   • I4_uint16_t reversed2
>= VESA version 3.0
   • 14 uint16 t linear bytes per scanline
   • 14 uint8 t banked number of image pages
   • 14 uint8 t linear_number_of_image_pages
   • 14_uint8_t linear_red_mask_size
   • I4_uint8_t linear_red_field_position
   • |4_uint8_t linear_green_mask_size
   • I4_uint8_t linear_green_field_position
   · 14 uint8 t linear blue mask size
   • I4_uint8_t linear_blue_field_position
   • |4_uint8_t linear_reserved_mask_size
   • I4_uint8_t linear_reserved_field_position
   • 14 uint32 t max pixel clock
```

11.137.1 Detailed Description

VBE mode information.

Definition at line 123 of file mb_info.h.

• 14 uint8 t reserved3 [189+1]

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

· I4/util/mb info.h

11.138 cxx::List < D, Alloc > Class Template Reference

Doubly linked list, with internal allocation.

Collaboration diagram for cxx::List< D, Alloc >:

cxx::List< D, Alloc >

- + List()
- + push_back()
- + push_front()
- + remove()
- + size()
- + operator[]()
- + operator[]()
- + items()

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.

11.138.1 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 335 of file list.

11.138.2 Member Function Documentation

```
11.138.2.1 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 381 of file list.

```
11.138.2.2 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 390 of file list.

```
11.138.2.3 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 399 of file list.

```
11.138.2.4 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 403 of file list.

```
11.138.2.5 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 406 of file list.

```
11.138.2.6 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 410 of file list.

11.138.2.7 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 414 of file list.

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

• I4/cxx/list

11.139 cxx::List_alloc Class Reference

Standard list-based allocator.

Collaboration diagram for cxx::List_alloc:

+ List_alloc()
+ free()
+ alloc()
+ avail()

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.

unsigned long avail ()

Get the amount of available memory.

11.139.1 Detailed Description

Standard list-based allocator.

Definition at line 30 of file list_alloc.

11.139.2 Constructor & Destructor Documentation

11.139.2.1 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 55 of file list alloc.

11.139.3 Member Function Documentation

11.139.3.1 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 200 of file list_alloc.

11.139.3.2 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 238 of file list_alloc.

11.139.3.3 unsigned long cxx::List_alloc::avail() [inline]

Get the amount of available memory.

Returns

Available memory in bytes

Definition at line 309 of file list_alloc.

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

• I4/cxx/list_alloc

11.140 cxx::List_item Class Reference

Basic list item.

Inherited by cxx::List< D, Alloc >::E, and cxx::T_list_item< T >.

Collaboration diagram for cxx::List_item:

cxx::List_item

- + List_item()
- + get_prev_item()
- + get_next_item()
- + insert_prev_item()
- + insert_next_item()
- + remove_me()
- + push_back()
- + push_front()
- + remove()
- # 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 ()

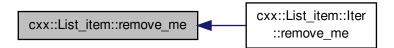
Remove this item from the list.

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.
11.140.1 Detailed Description
Basic list item.
Basic item that can be member of a doubly linked, cyclic list.
Definition at line 37 of file list.
11.140.2 Member Function Documentation
11.140.2.1 List_item* cxx::List_item::get_prev_item( ) const throw) [inline]
Get previous item.
Definition at line 174 of file list.
11.140.2.2 List item* cxx::List_item::get_next_item( ) const throw) [inline]
Get next item.
Definition at line 177 of file list.
11.140.2.3 void cxx::List_item::insert_prev_item ( List_item * p ) throw) [inline]
Insert item p before this item.
Definition at line 180 of file list.
11.140.2.4 void cxx::List_item::insert_next_item ( List_item * p ) throw ) [inline]
Insert item p after this item.
Definition at line 190 of file list.
11.140.2.5 void cxx::List_item::remove_me( ) throw) [inline]
Remove this item from the list.
Definition at line 199 of file list.
Referenced by cxx::List_item::Iter::remove_me().
```

Here is the caller graph for this function:



11.140.2.6 template < typename C , typename N > C * cxx::List_item::push_back(C * head, N *
$$p$$
) throw) [inline], [static]

Append item to a list.

Convinience function for empty-head corner case.

Parameters

h	pointer to the current list head.
р	pointer to new item.

Returns

the pointer to the new head.

Definition at line 249 of file list.

11.140.2.7 template < typename
$$N > C * cxx::List_item::push_front(C * head, N * p) throw) [inline], [static]$$

Prepend item to a list.

Convinience function for empty-head corner case.

Parameters

head	pointer to the current list head.
р	pointer to new item.

Returns

the pointer to the new head.

Definition at line 260 of file list.

11.140.2.8 template < typename C , typename N > C * cxx::List_item::remove (C * head, N *
$$p$$
) throw) [inline], [static]

Remove item from a list.

Convinience function for remove-head corner case.

Parameters

head	pointer to the current list head.
р	pointer to the item to remove.

Returns

the pointer to the new head.

Definition at line 270 of file list.

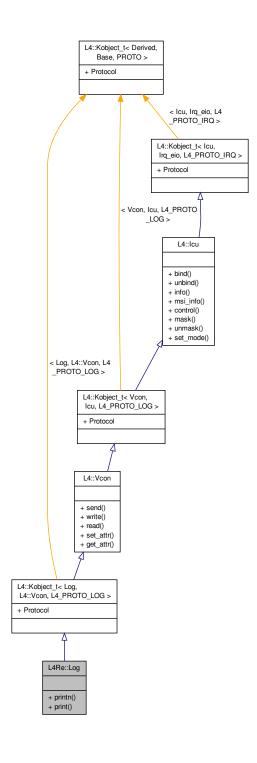
The documentation for this class was generated from the following file:

• I4/cxx/list

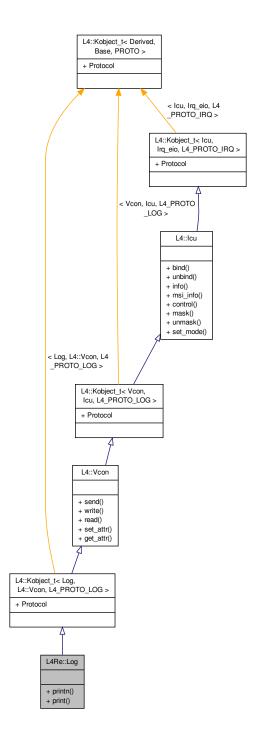
11.141 L4Re::Log Class Reference

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

11.141.1 Detailed Description

Log interface class.

Definition at line 44 of file log.

11.141.2 Member Function Documentation

11.141.2.1 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

11.141.2.2 void L4Re::Log::print (char const * string) const throw)

Print NULL-terminated string.

Parameters

strina	string to print
oung	ouring to print

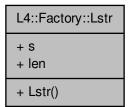
The documentation for this class was generated from the following file:

• I4/re/log

11.142 L4::Factory::Lstr Struct Reference

Special type to add a pascal string into the factory create stream.

 $Collaboration\ diagram\ for\ L4:: Factory:: Lstr:$



Data Fields

• char const * s

The character buffer.

• int len

The number of characters in the buffer.

11.142.1 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 61 of file factory.

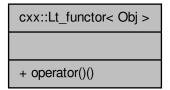
The documentation for this struct was generated from the following file:

· I4/sys/factory

11.143 cxx::Lt_functor< Obj > Struct Template Reference

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

Collaboration diagram for cxx::Lt_functor< Obj >:



11.143.1 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.

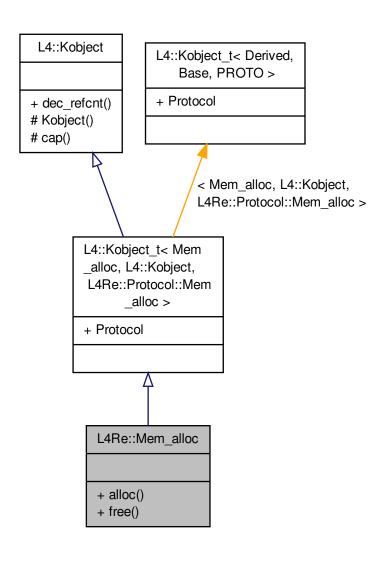
The documentation for this struct was generated from the following file:

I4/cxx/std_ops

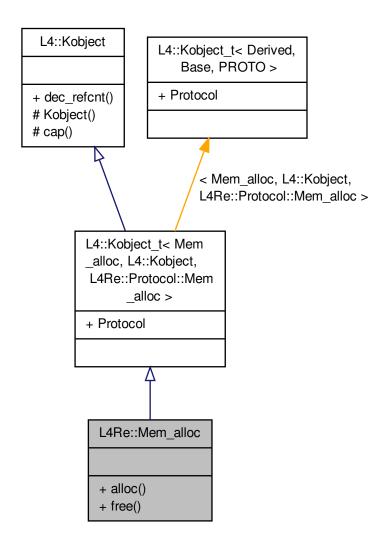
11.144 L4Re::Mem_alloc Class Reference

Memory allocator.

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 (unsigned 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 data space.

Additional Inherited Members

11.144.1 Detailed Description

Memory allocator.

Memory-allocator interface, for more information see Memory-allocator API.

Definition at line 69 of file mem_alloc.

11.144.2 Member Enumeration Documentation

11.144.2.1 enum L4Re::Mem_alloc::Mem_alloc_flags

Flags for the allocator.

Enumerator

Continuous Allocate physically contiguous data space, if supported by the allocator.

Pinned Allocate pinned data space, if supported by the allocator.

Super_pages Allocate super pages, if supported by the allocator.

Definition at line 78 of file mem_alloc.

11.144.3 Member Function Documentation

11.144.3.1 long L4Re::Mem_alloc::alloc (unsigned long *size*, L4::Cap< Dataspace > *mem*, unsigned long *flags* = 0, unsigned long *align* = 0) const throw)

Allocate anonymous memory.

Parameters

size	Size to be requested in bytes (granularity is (super)pages and the size is rounded up to this
	granularity).
mem	Object capability for the data space to be allocated.
flags	Flags, see Mem_alloc_flags, default none
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

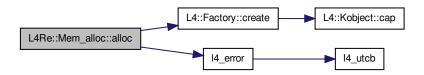
0 on success, <0 on error

- -L4_ENOMEM
- · IPC errors

Definition at line 38 of file mem_alloc_impl.h.

References L4::Factory::create(), and I4_error().

Here is the call graph for this function:



11.144.3.2 long L4Re::Mem_alloc::free (L4::Cap < Dataspace > mem) const throw)

Free data space.

Parameters

mem	Data space that contains the memory.

Returns

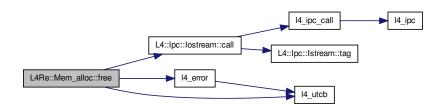
0 on success, <0 on error

- -L4_EINVAL
- · IPC errors

Definition at line 50 of file mem_alloc_impl.h.

 $References\ L4::lpc::lostream::call(),\ l4_error(),\ l4_utcb(),\ and\ L4Re::Protocol::Mem_alloc.$

Here is the call graph for this function:



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

- I4/re/mem_alloc
- I4/re/impl/mem_alloc_impl.h

11.145 L4::Kip::Mem_desc Class Reference

Memory descriptors stored in the kernel interface page.

Collaboration diagram for L4::Kip::Mem_desc:

+ Mem_desc()
+ start()
+ end()
+ size()
+ type()
+ sub_type()
+ is_virtual()
+ set()
+ first()
+ count()

Public Types

enum Mem_type
 Memory types.

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.

11.145.1 Detailed Description

Memory descriptors stored in the kernel interface page.

```
#include <14/sys/kip>
```

Definition at line 51 of file kip.

11.145.2 Constructor & Destructor Documentation

11.145.2.1 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 125 of file kip.

11.145.3 Member Function Documentation

```
11.145.3.1 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 83 of file kip.

11.145.3.2 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

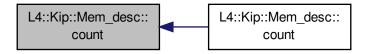
Returns

Number of memory descriptors in the kernel info page.

Definition at line 100 of file kip.

Referenced by count().

Here is the caller graph for this function:



11.145.3.3 static void L4::Kip::Mem_desc::count (void * kip, unsigned count) throw) [inline], [static]

Set number of memory descriptors.

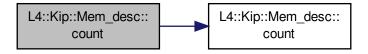
Parameters

kip	Pointer to the kernel info page
count	Number of memory descriptors

Definition at line 110 of file kip.

References count().

Here is the call graph for this function:



11.145.3.4 unsigned long L4::Kip::Mem_desc::start() const throw) [inline]

Return start address of memory descriptor.

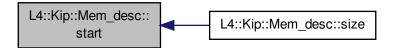
Returns

Start address of memory descriptor

Definition at line 135 of file kip.

Referenced by size().

Here is the caller graph for this function:



11.145.3.5 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 141 of file kip.

Referenced by size().

Here is the caller graph for this function:



11.145.3.6 unsigned long L4::Kip::Mem_desc::size() const throw) [inline]

Return size of region described by the memory descriptor.

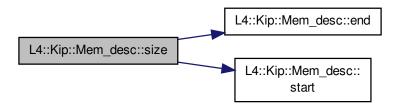
Returns

Size of the region described by the memory descriptor

Definition at line 147 of file kip.

References end(), and start().

Here is the call graph for this function:



```
11.145.3.7 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 153 of file kip.

11.145.3.8 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 159 of file kip.

11.145.3.9 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 166 of file kip.

11.145.3.10 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 176 of file kip.

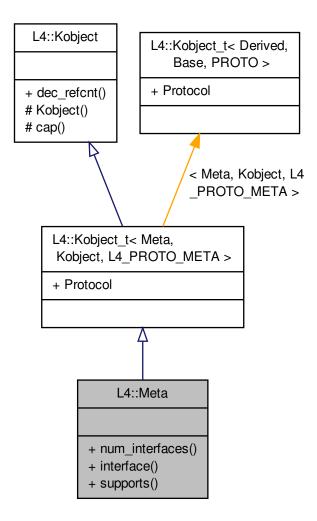
The documentation for this class was generated from the following file:

• I4/sys/kip

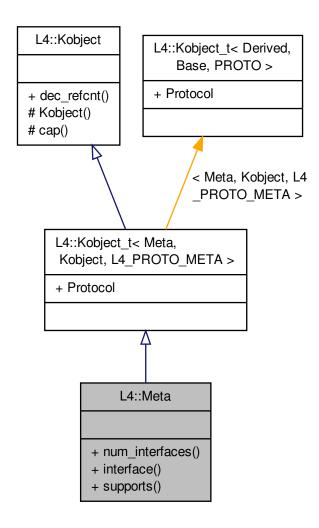
11.146 L4::Meta Class Reference

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

- I4_msgtag_t num_interfaces (I4_utcb_t *utcb=I4_utcb()) throw ()
 - Get the number of interfaces implemented by this object.

• I4_msgtag_t interface (int idx, I4_utcb_t *u=I4_utcb()) throw ()

Get the protocol number that must be used for the interface with the index idx.

• I4_msgtag_t supports (long protocol, I4_utcb_t *u=I4_utcb()) throw ()

Figure out if the object supports the given protocol (number).

Additional Inherited Members

11.146.1 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 41 of file meta.

11.146.2 Member Function Documentation

11.146.2.1 I4_msgtag_t L4::Meta::num_interfaces (I4_utcb_t * utcb = I4_utcb ()) throw) [inline]

Get the number of interfaces implemented by this object.

Parameters

utcb	is the utcb to use for sending the message.

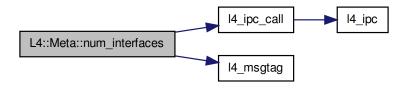
Returns

The message tag for the operation, the label (I4_msgtag_t::label()) is set to the number of interfaces if successful, or to -error when an error occured.

Definition at line 89 of file meta.

References I4_ipc_call(), L4_IPC_NEVER, I4_msgtag(), and I4_msg_regs_t::mr.

Here is the call graph for this function:



11.146.2.2 I4_msgtag_t L4::Meta::interface (int idx, I4_utcb_t * u = I4_utcb ()) throw) [inline]

Get the protocol number that must be used for the interface with the index idx.

Parameters

idx	is the index of the interface to get the protocol number for. idx must be $>= 0$ and $<$ the return value of num_interfaces().
utcb	is the utcb to use for sending the message.

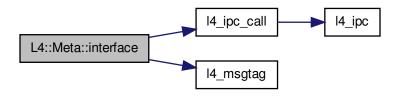
Returns

The message tag for the operation, the label (I4_msgtag_t::label()) is set to the protocol number of interface idx.

Definition at line 98 of file meta.

References I4_ipc_call(), L4_IPC_NEVER, I4_msgtag(), and I4_msg_regs_t::mr.

Here is the call graph for this function:



11.146.2.3 I4_msgtag_t L4::Meta::supports (long protocol, I4_utcb_t * u = I4_utcb ()) throw) [inline]

Figure out if the object supports the given *protocol* (number).

Parameters

protocol	is the protocol number to check for.
utcb	is the utcb to use for sending the message.

Returns

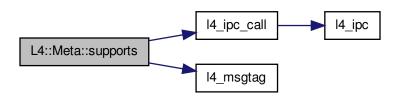
The message tag for the operation, the label (I4_msgtag_t::label()) is set to 1 if protocol is supported to 0 if not

This method is intended to be used for statically assigned protocol numbers.

Definition at line 108 of file meta.

References I4_ipc_call(), L4_IPC_NEVER, I4_msgtag(), and I4_msg_regs_t::mr.

Here is the call graph for this function:



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

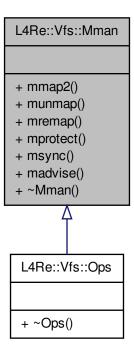
· I4/sys/meta

11.147 L4Re::Vfs::Mman Class Reference

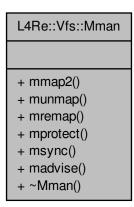
Interface for the POSIX memory management.

#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_adr)=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.

11.147.1 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 \(\frac{14}{14} \text{re_vfs/impl/vfs_impl.h} \) and used by the \(\frac{14}{14} \text{re_vfs library or by the VFS implementation in ldso.} \)

Definition at line 785 of file vfs.h.

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

I4/I4re_vfs/vfs.h

11.148 L4::Thread::Modify_senders Class Reference

Wrapper class for modifying senders.

Collaboration diagram for L4::Thread::Modify_senders:

L4::Thread::Modify _senders + Modify_senders() + add()

Public Member Functions

int add (I4_umword_t match_mask, I4_umword_t match, I4_umword_t del_bits, I4_umword_t add_bits) throw
 ()

Add a rule.

11.148.1 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 220 of file thread.

11.148.2 Member Function Documentation

11.148.2.1 int L4::Thread::Modify_senders::add (I4_umword_t match_mask, I4_umword_t match, I4_umword_t del_bits, I4_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

I4_thread_modify_sender_add()

Definition at line 249 of file thread.

References L4_ENOMEM, L4_UTCB_GENERIC_DATA_SIZE, and I4_msg_regs_t::mr.

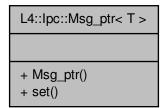
The documentation for this class was generated from the following file:

· I4/sys/thread

11.149 L4::lpc::Msg_ptr < T > Class Template Reference

Pointer to an element of type T in an Ipc::Istream.

Collaboration diagram for L4::lpc::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.

11.149.1 Detailed Description

template<typename T>class L4::lpc::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 loc::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 170 of file ipc stream.

11.149.2 Constructor & Destructor Documentation

11.149.2.1 template < typename T > L4::lpc::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 180 of file ipc stream.

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

• I4/cxx/ipc_stream

11.150 L4Re::Util::Names::Name Class Reference

Name class.

Inherits cxx::String.

Collaboration diagram for L4Re::Util::Names::Name:

+ Name() + Name() + Name() + name() + name()

+ operator<()

11.150.1 Detailed Description

Name class.

Definition at line 42 of file name_space_svr.

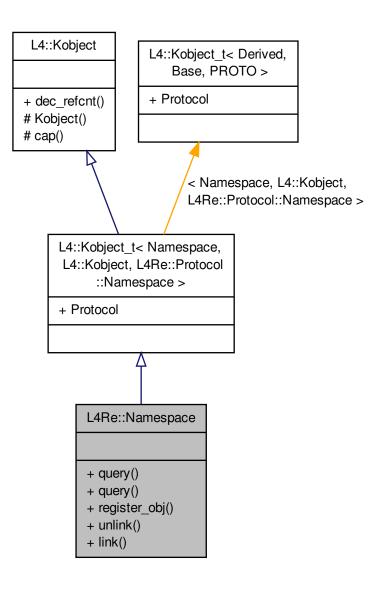
The documentation for this class was generated from the following file:

• I4/re/util/name_space_svr

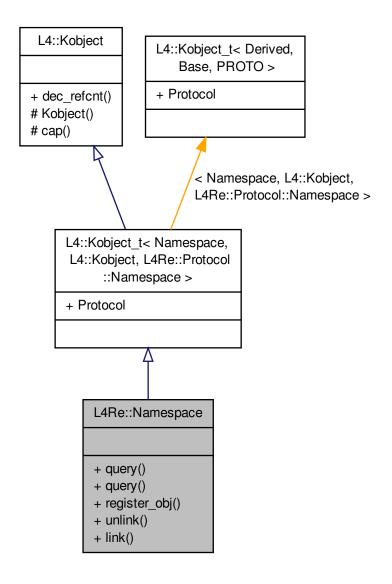
11.151 L4Re::Namespace Class Reference

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 , Strong = L4_CAP_FPAGE_RS }

Flags for registering name spaces.

Public Member Functions

• long query (char const *name, L4::Cap< void > const &cap, int timeout=To_default, I4_umword_t *local_id=0, bool iterate=true) const throw ()

Query a name.

 long query (char const *name, unsigned len, L4::Cap< void > const &cap, int timeout=To_default, I4_umword_t *local_id=0, bool iterate=true) const throw () Query a name.

long register_obj (char const *name, L4::Cap< void > const &obj, unsigned flags=Rw) const throw ()
 Register an object with a name.

Additional Inherited Members

11.151.1 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 57 of file namespace.

11.151.2 Member Enumeration Documentation

11.151.2.1 enum L4Re::Namespace::Register_flags

Flags for registering name spaces.

Enumerator

Ro Read-only.

Rw Read-write.

Strong Strong.

Definition at line 66 of file namespace.

11.151.3 Member Function Documentation

11.151.3.1 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 a name.

Parameters

name	String to query
сар	Capability slot to put object into.
timeout	Timeout of query in milliseconds.

Return values

local_id	Local id.

Returns

<0 on failure, see I4_error_code_t.

- · -L4 ENOENT
- IPC errors == 0 if name could be fully resolved > 0 if name could not be fully resolved

Definition at line 118 of file namespace_impl.h.

11.151.3.2 long L4Re::Namespace::query (char const * name, unsigned len, L4::Cap < void > const & cap, int timeout = To_default, I4_umword_t * local_id = 0, bool iterate = true) const throw)

Query a name.

Parameters

name	String to query
len	Length of the string to query.
сар	Capability slot to put object into.
timeout	Timeout of query in milliseconds.

Return values

local_id	Local id.

Returns

<0 on failure, see I4_error_code_t.

- -L4 ENOENT
- IPC errors == 0 if name could be fully resolved > 0 if name could not be fully resolved

Definition at line 84 of file namespace_impl.h.

References EXPECT_FALSE, and L4_EAGAIN.

11.151.3.3 long L4Re::Namespace::register_obj (char const * name, L4::Cap< void > const & obj, unsigned flags = Rw) const throw)

Register an object with a name.

Parameters

name	String to register.
obj	Object to register.
flags	Flags to use, see Register_flags, default is rw.

Returns

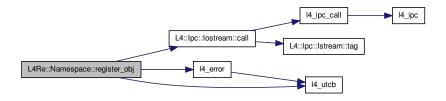
0 on success, <0 on failure, see I4_error_code_t.

- -L4_EEXIST
- -L4_EPERM
- -L4_ENOMEM
- -L4_EINVAL
- IPC errors

Definition at line 124 of file namespace_impl.h.

References L4::lpc::lostream::call(), I4_error(), I4_utcb(), and L4Re::Protocol::Namespace.

Here is the call graph for this function:



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

- I4/re/namespace
- I4/re/impl/namespace_impl.h

11.152 cxx::New_allocator< _Type > Class Template Reference

Standard allocator based on operator new () .

Collaboration diagram for cxx::New_allocator< _Type >:

11.152.1 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.

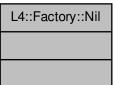
The documentation for this class was generated from the following file:

• I4/cxx/std_alloc

11.153 L4::Factory::Nil Struct Reference

Special type to add a void argument into the factory create stream.

Collaboration diagram for L4::Factory::Nil:



11.153.1 Detailed Description

Special type to add a void argument into the factory create stream.

Definition at line 53 of file factory.

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

· I4/sys/factory

11.154 cxx::Avl_set< Item, Compare, Alloc >::Node Class Reference

A smart pointer to a tree item.

Collaboration diagram for cxx::Avl_set< Item, Compare, Alloc >::Node:

cxx::Avl_set< Item,
Compare, Alloc >::Node

+ Node()
+ operator=()
+ operator*()
+ operator->()
+ valid()
+ operator Item const *()

Public Member Functions

• Node ()

Default construction for NIL pointer.

• Node & operator= (Node const &o)

Default assignment.

• Item const & operator* ()

Dereference the pointer.

Item const * operator-> ()

Dereferenced member access.

· bool valid () const

Validity check.

operator Item const * ()

Cast to a real item pointer.

11.154.1 Detailed Description

 $template < typename \ A > class \ Alloc = New_allocator > class \\ cxx::Avl_set < Item, Compare, Alloc > ::Node$

A smart pointer to a tree item.

Definition at line 148 of file avl set.

11.154.2 Member Function Documentation

11.154.2.1 template<typename Item, class Compare = Lt_functor<Item>, template< typename A > class Alloc = New_allocator> bool cxx::Avl_set< Item, Compare, Alloc >::Node::valid () const [inline]

Validity check.

Returns

false if the pointer is NIL, true if valid.

Definition at line 172 of file avl_set.

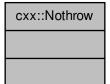
The documentation for this class was generated from the following file:

I4/cxx/avl_set

11.155 cxx::Nothrow Class Reference

Helper type to distinguish the opprator new version that does not throw exceptions.

Collaboration diagram for cxx::Nothrow:



11.155.1 Detailed Description

Helper type to distinguish the opprator 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:

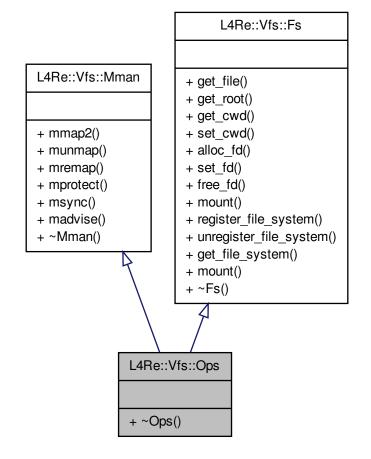
· I4/cxx/std alloc

11.156 L4Re::Vfs::Ops Class Reference

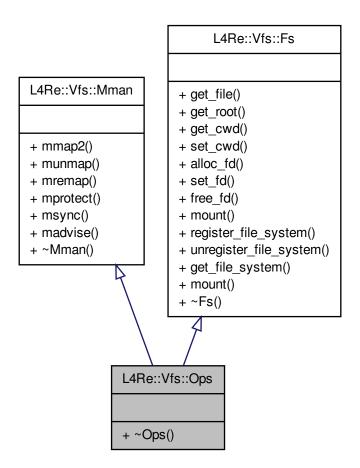
Interface for the POSIX backends for an application.

#include <vfs.h>

Inheritance diagram for L4Re::Vfs::Ops:



Collaboration diagram for L4Re::Vfs::Ops:



Additional Inherited Members

11.156.1 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 1016 of file vfs.h.

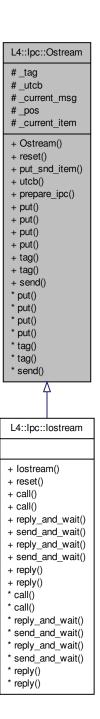
The documentation for this class was generated from the following file:

• I4/I4re_vfs/vfs.h

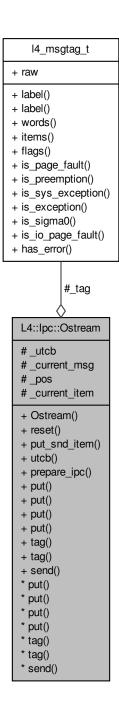
11.157 L4::lpc::Ostream Class Reference

Output stream for IPC marshalling.

Inheritance diagram for L4::lpc::Ostream:



Collaboration diagram for L4::lpc::Ostream:



Public Member Functions

• Ostream (I4_utcb_t *utcb)

Create an IPC output stream using the given message buffer msg.

• void reset ()

Reset the stream to empty, same state as a newly created stream.

• I4_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.

See IPC stream operators.

template<typename T >
 void 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)

• I4_msgtag_t tag () const

Extract the L4 message tag from the stream.

• I4_msgtag_t & tag ()

Extract a reference to the L4 message tag from the stream.

IPC operations.

• I4_msgtag_t send (I4_cap_idx_t dst, long proto=0, unsigned flags=0)

Send the message via IPC to the given receiver.

11.157.1 Detailed Description

Output stream for IPC marshalling.

lpc::Ostream is part of the dynamic IPC marshalling infrastructure, as well as lpc::Istream and lpc::lostream.

lpc::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 lpc::Buf_cp_out) and indirect strings (see Msg_out_buffer).

Definition at line 843 of file ipc_stream.

11.157.2 Member Function Documentation

11.157.2.1 template < typename T > void L4::lpc::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 881 of file ipc_stream.

Referenced by operator<<().

Here is the caller graph for this function:



11.157.2.2 template<typename T > bool L4::lpc::Ostream::put (T const & v) [inline]

Insert an element of type *T* into the stream.

Parameters

```
ν The element to insert.
```

Definition at line 897 of file ipc_stream.

11.157.2.3 I4_msgtag_t L4::lpc::Ostream::tag()const [inline]

Extract the L4 message tag from the stream.

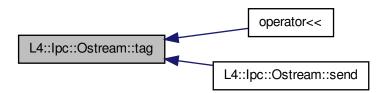
Returns

the extracted L4 message tag.

Definition at line 924 of file ipc_stream.

Referenced by operator<<(), and send().

Here is the caller graph for this function:



11.157.2.4 I4_msgtag_t&L4::lpc::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 930 of file ipc_stream.

11.157.2.5 I4_msgtag_t L4::lpc::Ostream::send (I4_cap_idx_t dst, long proto = 0, unsigned flags = 0) [inline]

Send the message via IPC to the given receiver.

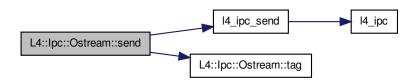
Parameters

dst	The destination for the message.
-----	----------------------------------

Definition at line 1152 of file ipc_stream.

References L4_IPC_NEVER, I4_ipc_send(), L4_MSGTAG_FLAGS, and tag().

Here is the call graph for this function:



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

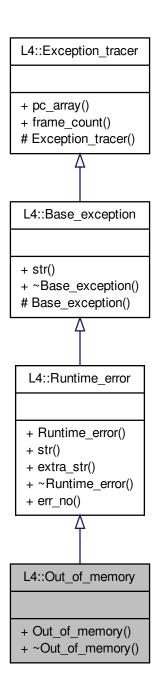
· I4/cxx/ipc_stream

11.158 L4::Out_of_memory Class Reference

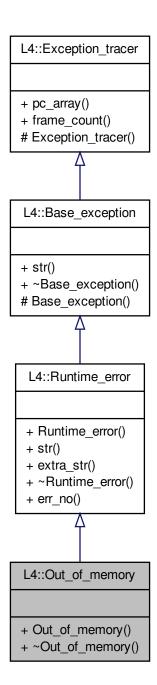
Exception signalling insufficient memory.

#include <14/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

11.158.1 Detailed Description

Exception signalling insufficient memory.

Definition at line 171 of file exceptions.

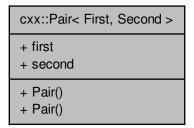
The documentation for this class was generated from the following file:

• I4/cxx/exceptions

11.159 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

Pair (First const &first, Second const &second)

Create a pair from the two values.

• Pair ()

Default construction.

Data Fields

First first

First value.

Second second

Second value.

11.159.1 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.

11.159.2 Constructor & Destructor Documentation

11.159.2.1 template<typename First, typename Second> cxx::Pair< First, Second>::Pair (First const & first, Second const & second) [inline]

Create a pair from the two values.

Parameters

first	The first value.
second	The second value.

Definition at line 53 of file pair.

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

• I4/cxx/pair

11.160 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 &I, Typ const &r) const

Do the comaprison based on the first value.

11.160.1 Detailed Description

template<typename Cmp, typename Typ>class cxx::Pair_first_compare< Cmp, Typ>

Comparison functor for Pair.

Parameters

Стр	Comparison functor for the first value of the pair.
Тур	The pair type.

This functor can be used to compare Pair values with respect to the first value.

Definition at line 74 of file pair.

11.160.2 Constructor & Destructor Documentation

```
11.160.2.1 template < typename Cmp , typename Typ > cxx::Pair_first_compare < Cmp, Typ >::Pair_first_compare ( Cmp const & cmp = Cmp () ) [inline]
```

Construction.

Parameters

стр	The comparison functor used for the first value.
CITIP	The companson functor used for the first value.

Definition at line 84 of file pair.

11.160.3 Member Function Documentation

11.160.3.1 template < typename Cmp , typename Typ > bool cxx::Pair_first_compare < Cmp, Typ >::operator() (Typ const & I, Typ const & r) const [inline]

Do the comaprison based on the first value.

Parameters

1	The lefthand value.
r	The righthand value.

Definition at line 91 of file pair.

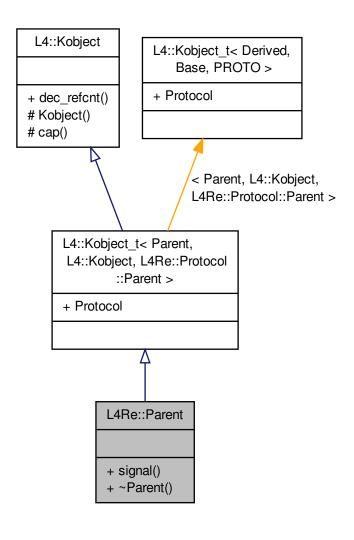
The documentation for this class was generated from the following file:

• I4/cxx/pair

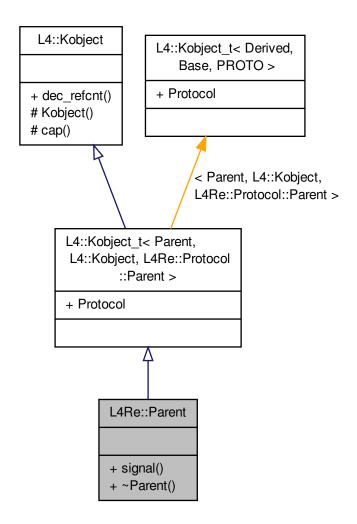
11.161 L4Re::Parent Class Reference

Parent interface.

Inheritance diagram for L4Re::Parent:



Collaboration diagram for L4Re::Parent:



Public Member Functions

long signal (unsigned long sig, unsigned long val) const throw ()
 Send a signal to the parent.

Additional Inherited Members

11.161.1 Detailed Description

Parent interface.

See Also

Parent API for more details about the purpose.

Definition at line 50 of file parent.

11.161.2 Member Function Documentation

11.161.2.1 long L4Re::Parent::signal (unsigned long sig, unsigned long val) const throw)

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:

• I4/re/parent

11.162 L4Re::Video::Pixel_info Class Reference

Pixel information.

Collaboration diagram for L4Re::Video::Pixel_info:

L4Re::Video::Pixel_info

+ r()
+ g()
+ b()
+ a()
+ bytes_per_pixel()
+ bits_per_pixel()
+ has_alpha()
+ r()
+ g()
+ b()
and 7 more...

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 equzality of the color sapce.

 $\bullet \ \ \mathsf{template} \mathord{<} \mathsf{typename} \ \mathsf{STREAM} >$

STREAM & dump (STREAM &s) const

Dump information on the pixel to a stream.

11.162.1 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 108 of file colors.

11.162.2 Constructor & Destructor Documentation

11.162.2.1 L4Re::Video::Pixel_info::Pixel_info (unsigned char *bpp*, char *r*, char *rs*, char *g*, char *g*, 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.
а	Alpha component size, defaults to 0.
as	Alpha component shift, defaults to 0.

Definition at line 205 of file colors.

11.162.2.2 template < typename VBI > L4Re::Video::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 217 of file colors.

11.162.3 Member Function Documentation

11.162.3.1 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 119 of file colors.

11.162.3.2 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 125 of file colors.

11.162.3.3 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 131 of file colors.

```
11.162.3.4 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 137 of file colors.
11.162.3.5 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 143 of file colors.
11.162.3.6 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 149 of file colors.
11.162.3.7 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 156 of file colors.
11.162.3.8 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 162 of file colors.
11.162.3.9 void L4Re::Video::Pixel_info::g( Color_component const & c) [inline]
```

Set the green color component of the pixel.

c Green color component.

Definition at line 168 of file colors.

11.162.3.10 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 174 of file colors.

11.162.3.11 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 180 of file colors.

11.162.3.12 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 186 of file colors.

11.162.3.13 bool L4Re::Video::Pixel_info::operator== (Pixel_info const & o) const [inline]

Compare for complete equzality of the color sapce.

Parameters

A Pixel info to compare to.

Returns

true if the both Pixel_info's are equal, false if not.

Definition at line 229 of file colors.

11.162.3.14 template<typename STREAM > STREAM& L4Re::Video::Pixel_info::dump(STREAM & s) const [inline]

Dump information on the pixel to a stream.

Parameters

s Stream

Returns

The stream

Definition at line 240 of file colors.

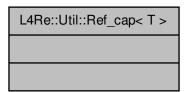
The documentation for this class was generated from the following file:

• I4/re/video/colors

11.163 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 >:



11.163.1 Detailed Description

template < typename T > struct L4Re::Util::Ref_cap < T >

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

Parameters

```
T | the 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:

Definition at line 227 of file cap_alloc.

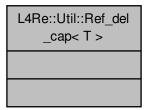
The documentation for this struct was generated from the following file:

I4/re/util/cap_alloc

11.164 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 >:



11.164.1 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.

Parameters

```
T | the 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>::Ca global_ds_cap;

* 
* 
* L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap

* ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Datasapce>));

* // 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 exiting).

* 
* 
* global_ds_cap = L4_INVALID_CAP;

* // reference count dropped to 0 (data space shall be deleted).

* *// reference count dropped to 0 (data space shall be deleted).
```

Definition at line 267 of file cap_alloc.

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

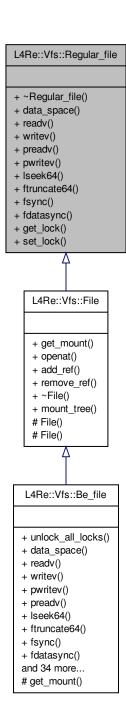
• I4/re/util/cap_alloc

11.165 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:

L4Re::Vfs::Regular_file

- + ~Regular_file()
- + data_space()
- + readv()
- + writev()
- + preadv()
- + pwritev()
- + lseek64()
- + ftruncate64()
- + fsync()
- + fdatasync()
- + get lock()
- + set_lock()

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.

11.165.1 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 262 of file vfs.h.

11.165.2 Member Function Documentation

```
11.165.2.1 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 functions returns an invalid capability.

Returns

A capability to an L4Re::Dataspace, that represents the files contents in an L4Re way.

Implemented in L4Re::Vfs::Be_file.

```
11.165.2.2 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.

```
11.165.2.3 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.

```
11.165.2.4 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, Iseek and friends.

Returns

The new file position, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

```
11.165.2.5 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.

pos	The offset at which the file shall be truncated.
-----	--

Returns

0 on success, or <0 on error.

Implemented in L4Re::Vfs::Be_file.

11.165.2.6 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.

11.165.2.7 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.

11.165.2.8 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 <i>l_type</i> member will contain #F_UNLCK if the
	lock could be placed.

Returns

0 on success, <0 on error.

Implemented in L4Re::Vfs::Be_file.

11.165.2.9 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.

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

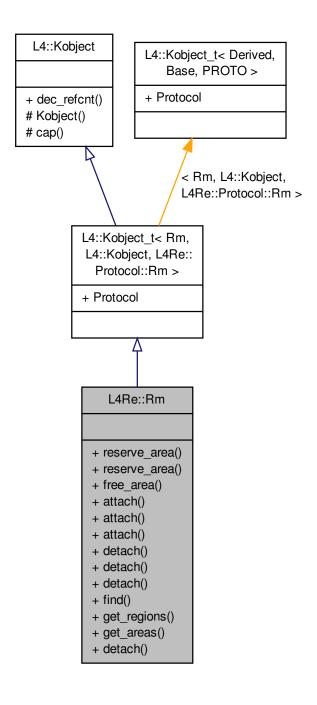
• I4/I4re_vfs/vfs.h

11.166 L4Re::Rm Class Reference

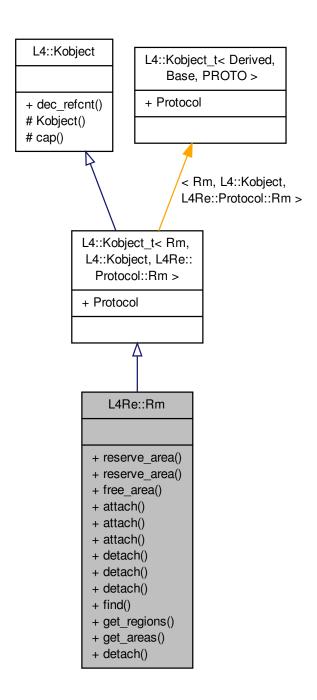
Region map.

#include <14/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,
        Region_flags = 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_PAGES-HIFT) 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) const throw ()

Free an area from the region map.

long attach (I4_addr_t *start, unsigned long size, unsigned long flags, L4::Cap < Dataspace > mem, I4_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::Cap< Dataspace > mem, I4_addr_t offs=0, unsigned char align=L4 PAGESHIFT) const throw ()

Attach a dataspace 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 (I4_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 (I4_addr_t *addr, unsigned long *size, I4_addr_t *offset, unsigned *flags, L4::Cap< Dataspace >
 *m) throw ()

Find a region given an address and size.

Additional Inherited Members

11.166.1 Detailed Description

Region map.

Definition at line 69 of file rm.

11.166.2 Member Enumeration Documentation

11.166.2.1 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 76 of file rm.

11.166.2.2 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)

Region_flags Mask of all region flags.

Definition at line 87 of file rm.

11.166.2.3 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 99 of file rm.

11.166.2.4 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 109 of file rm.

11.166.3 Member Function Documentation

11.166.3.1 long L4Re::Rm::reserve_area (I4_addr_t * start, unsigned long size, unsigned flags = 0, unsigned char align = L4_PAGESHIFT) const throw)

Reserve the given area in the region map.

start	The virtual start address of the area to reserve.
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

start	Start of address.

Returns

0 on success, <0 on error

- -L4 EADDRNOTAVAIL
- · 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.

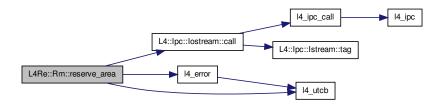
Note

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 41 of file rm_impl.h.

References L4::lpc::lostream::call(), EXPECT_FALSE, I4_error(), I4_utcb(), and L4Re::Protocol::Rm.

Here is the call graph for this function:



11.166.3.2 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.

start	The virtual start address of the area to reserve.
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

start	Start of address.

Returns

0 on success, <0 on error

- -L4_EADDRNOTAVAIL
- · IPC errors

For more information, please refer to the analogous function

See Also

L4Re::Rm::reserve_area.

Definition at line 241 of file rm.

11.166.3.3 long L4Re::Rm::free_area (I4_addr_t addr) const throw)

Free an area from the region map.

Parameters

addr	An address within the area to free.

Returns

0 on success, <0 on error

- -L4_ENOENT
- IPC errors

Note

The data spaces that are attached to that area are not detached by this operation.

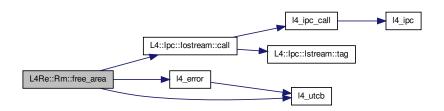
See Also

reserve_area() for more information about areas.

Definition at line 55 of file rm impl.h.

References L4::lpc::lostream::call(), I4_error(), I4_utcb(), and L4Re::Protocol::Rm.

Here is the call graph for this function:



11.166.3.4 long L4Re::Rm::attach (I4_addr_t * start, unsigned long size, unsigned long flags, L4::Cap< Dataspace > mem, I4_addr_t offs = 0, unsigned char align = L4_PAGESHIFT) const throw)

Attach a data space to a region.

Parameters

start	Virtual start address
size	Size of the data space to attach (in bytes)
flags	Flags, see Attach_flags and Region_flags
mem	Data space
offs	Offset into the data space to use
align	Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT), Only meaningful
	if the Search_addr flag is used.

Return values

start	Start of region if Search_addr was used.
-------	--

Returns

0 on success, <0 on error

- -L4_ENOENT
- -L4_EPERM
- -L4 EINVAL
- -L4_EADDRNOTAVAIL
- 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

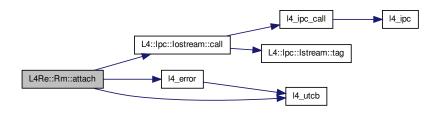
When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see find).

Definition at line 63 of file rm_impl.h.

References L4::lpc::lostream::call(), EXPECT_FALSE, I4_error(), I4_utcb(), L4Re::Dataspace::Map_ro, L4Re::Dataspace::Map_rw, and L4Re::Protocol::Rm.

Here is the call graph for this function:



11.166.3.5 template < typename T > long L4Re::Rm::attach (T ** start, unsigned long size, unsigned long flags, L4::Cap < Dataspace > mem, I4_addr_t offs = 0, unsigned char align = L4_PAGESHIFT) const throw) [inline]

Attach a dataspace to a region.

See Also

attach

Definition at line 301 of file rm.

11.166.3.6 int L4Re::Rm::detach (I4_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.

Return values

mem	Dataspace that is affected. Give 0 if not interested.

Parameters

task	If given, task specifies the task where the pages are unmapped. Give 0 for none. Default is
	current task.

Returns

Detach_result on success, <0 on error

- -L4_ENOENT
- 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 451 of file rm.

11.166.3.7 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.

addr	Virtual address of region, any address within the region is valid.
------	--

Return values

mem	Dataspace that is affected. Give 0 if not interested.

Parameters

task	If given, task specifies the task where the pages are unmapped. Give 0 for none. Default is]
	current task.	

Returns

Detach_result on success, <0 on error

- -L4_ENOENT
- · IPC errors

Frees a region in the virtual address space given by addr (void pointer type). The corresponding part of the address space is now available again.

Definition at line 456 of file rm.

```
11.166.3.8 int L4Re::Rm::detach ( I4_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).

Return values

mem	Dataspace that is affected. Give 0 if not interested.

Parameters

task Specifies the task where the pages are unmapped. Give 0 for none.
--

Returns

Detach_result on success, <0 on error

- · -L4 ENOENT
- · 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 seperate regions.

Definition at line 461 of file rm.

```
11.166.3.9 int L4Re::Rm::find ( I4_addr_t * addr, unsigned long * size, I4_addr_t * offset, unsigned * flags, L4::Cap  
Dataspace > * m ) throw )
```

Find a region given an address and size.

addr	Address to look for
size	Size of the area to look for (in bytes).

Return values

addr	Start address of the found region.
size	Size of the found region (in bytes).
offset	Offset at the beginning of the region within the associated dataspace.
flags	Region flags, see Region_flags.
m	Associated dataspace or paging service.

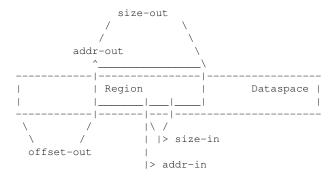
Returns

0 on success, <0 on error

-L4_EPERM: not allowed-L4_ENOENT: not found

· IPC errors

This function returns the properties of the region that contains the area described by the addr and size parameter.



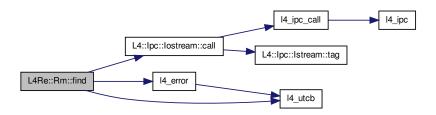
Note

The value of the size input parameter should be 1 to assure that a region can be determined unambiguously.

Definition at line 143 of file rm_impl.h.

References L4::lpc::lostream::call(), EXPECT_FALSE, I4_error(), I4_utcb(), and L4Re::Protocol::Rm.

Here is the call graph for this function:



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

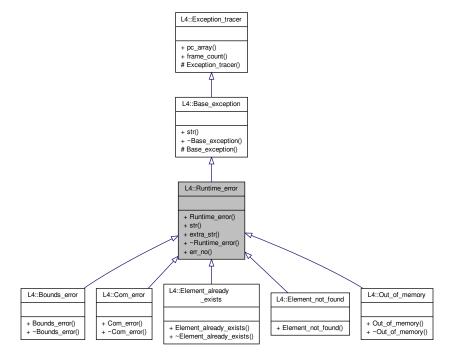
- I4/re/rm
- I4/re/impl/rm_impl.h

11.167 L4::Runtime_error Class Reference

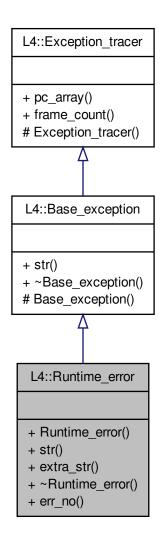
Exception for an abstract runtime error.

#include <14/cxx/exceptions>

Inheritance diagram for L4::Runtime_error:



Collaboration diagram for L4::Runtime_error:



Public Member Functions

char const * str () const throw ()
 Should return a human readable string for the exception.

Additional Inherited Members

11.167.1 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 I4_error_code_t).

Definition at line 140 of file exceptions.

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

• I4/cxx/exceptions

11.168 L4::Factory::S Class Reference

Stream class for the create() argument stream.

Collaboration diagram for L4::Factory::S:

L4::Factory::S + S() + S() + ~S() + operator I4_msgtag_t() + operator<<() + operator<<() + operator<<() + operator<<() + operator<<() + operator<<() + operator<<()

Public Member Functions

```
• S (S const &o)
```

create a copy.

• S (I4_cap_idx_t f, long obj, L4::Cap< L4::Kobject > target, I4_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 I4_msgtag_t ()

Explicitely commits the operation and returns the result.

• S & operator << (I4_mword_t i)

Put a single I4_mword_t as next argument.

• S & operator << (I4_umword_t i)

Put a single I4_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<< (I4_fpage_t d)

Add a flex page as next argument.

11.168.1 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 82 of file factory.

11.168.2 Constructor & Destructor Documentation

```
11.168.2.1 L4::Factory::S::S ( I4_cap_idx_t f, long obj, L4::Cap< L4::Kobject > target, I4_utcb_t * utcb ) throw )
[inline]
```

create a stream for a specific create() call.

Parameters

f	is the capability for the factory object (L4::Factory).
obj	is the protocol ID to describe the type of the object that shall be created.
target	is the capabilit selector for the new object.
utcb	is the UTCB that shall be used for the operation.

Definition at line 105 of file factory.

11.168.3 Member Function Documentation

```
11.168.3.1 L4::Factory::S::operator I4_msgtag_t() [inline]
```

Explicitely commits the operation and returns the result.

Returns

The result of the create() operation.

Definition at line 124 of file factory.

References I4_msgtag_t::raw.

```
11.168.3.2 S& L4::Factory::S::operator<<( I4_mword_t i ) [inline]
```

Put a single I4_mword_t as next argument.

Parameters

```
i is the value to add as next argument.
```

Definition at line 135 of file factory.

```
11.168.3.3 S& L4::Factory::S::operator<< ( I4 umword t i ) [inline]
```

Put a single I4_umword_t as next argument.

Parameters

i	is the value to add as next argument.

Definition at line 145 of file factory.

11.168.3.4 S& L4::Factory::S::operator<< (char const * s) [inline]

Add a zero-terminated string as next argument.

s is the string to add as next argument.

Definition at line 155 of file factory.

11.168.3.5 S& L4::Factory::S::operator<< (Lstr const & s) [inline]

Add a pascal string as next argument.

Parameters

s is the string to add as next argument.

Definition at line 165 of file factory.

References L4::Factory::Lstr::len, and L4::Factory::Lstr::s.

11.168.3.6 S&L4::Factory::S::operator<<**(I4_fpage_t** *d* **)** [inline]

Add a flex page as next argument.

Parameters

d is the flex page to add (there will be no map operation).

Definition at line 184 of file factory.

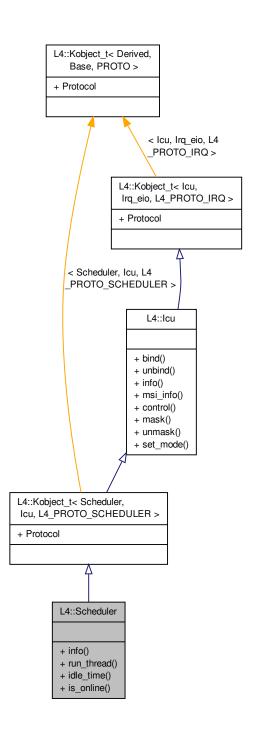
The documentation for this class was generated from the following file:

• I4/sys/factory

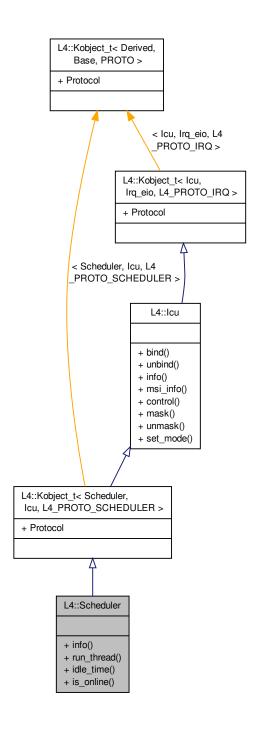
11.169 L4::Scheduler Class Reference

Scheduler object.

Inheritance diagram for L4::Scheduler:



Collaboration diagram for L4::Scheduler:



Public Member Functions

• I4_msgtag_t info (I4_umword_t *cpu_max, I4_sched_cpu_set_t *cpus, I4_utcb_t *utcb=I4_utcb()) const throw ()

Get scheduler information.

I4_msgtag_t run_thread (Cap< Thread > const &thread, I4_sched_param_t const &sp, I4_utcb_t *utcb=I4_utcb()) const throw ()

Run a thread on a Scheduler.

- I4_msgtag_t idle_time (I4_sched_cpu_set_t const &cpus, I4_utcb_t *utcb=I4_utcb()) const throw ()

 Query idle time of a CPU, in \(\mu s. \)
- bool is_online (I4_umword_t cpu, I4_utcb_t *utcb=I4_utcb()) const throw ()

 Query if a CPU is online.

Additional Inherited Members

11.169.1 Detailed Description

Scheduler object.

#include <14/sys/scheduler>

See Also

Scheduler for an overview description.

Definition at line 41 of file scheduler.

11.169.2 Member Function Documentation

```
11.169.2.1 I4_msgtag_t L4::Scheduler::info ( I4_umword_t * cpu_max, I4_sched_cpu_set_t * cpus, I4_utcb_t * utcb = I4_utcb () ) const throw) [inline]
```

Get scheduler information.

Parameters

scheduler	Scheduler object.

Return values

Parameters

		cous	cpus.offset is first CPU of interest. cpus.granularity (see 4_sched_cpu_set_t).
--	--	------	--

Return values

cpus	cpus.map Bitmap of online CPUs.

Returns

0 on success, <0 error code otherwise.

Note

scheduler is the implicit this pointer.

Definition at line 50 of file scheduler.

References L4::Kobject::cap().

Here is the call graph for this function:



11.169.2.2 I4_msgtag_t L4::Scheduler::run_thread (Cap < Thread > const & thread, I4_sched_param_t const & sp, I4_utcb_t * utcb = I4_utcb ()) const throw) [inline]

Run a thread on a Scheduler.

Parameters

scheduler	Scheduler object.
thread	Thread to run.
sp	Scheduling parameters.

Returns

0 on success, <0 error code otherwise.

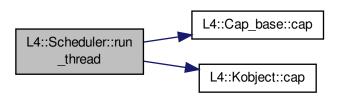
Note

scheduler is the implicit this pointer.

Definition at line 58 of file scheduler.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



11.169.2.3 I4_msgtag_t L4::Scheduler::idle_time (I4_sched_cpu_set_t const & cpus, I4_utcb_t * utcb = I4_utcb ()) const throw) [inline]

Query idle time of a CPU, in μ s.

scheduler	Scheduler object.
cpus	Set of CPUs to query.

The consumed time is returned as I4_kernel_clock_t at UTCB message register 0.

Note

scheduler is the implicit this pointer.

Definition at line 67 of file scheduler.

References L4::Kobject::cap().

Here is the call graph for this function:



11.169.2.4 bool L4::Scheduler::is_online (I4_umword_t cpu, I4_utcb_t * utcb = I4_utcb ()) const throw)
[inline]

Query if a CPU is online.

Parameters

scheduler	Scheduler object.
сри	CPU number.

Returns

true if online, false if not (or any other query error).

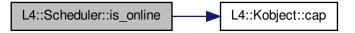
Note

scheduler is the implicit this pointer.

Definition at line 76 of file scheduler.

References L4::Kobject::cap().

Here is the call graph for this function:



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

• I4/sys/scheduler

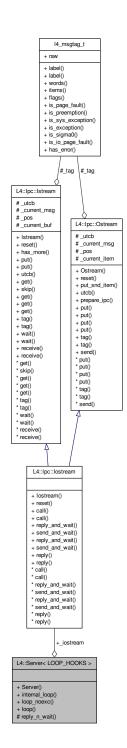
11.170 L4::Server < LOOP_HOOKS > Class Template Reference

Basic server loop for handling client requests.

Inherits LOOP_HOOKS.

Inherited by L4Re::Util::Registry_server< LOOP_HOOKS >.

Collaboration diagram for L4::Server< LOOP_HOOKS >:



Public Member Functions

• Server (I4_utcb_t *utcb)

Initializes the server loop and its underlying lpc::lostream.

 $\bullet \ \ \mathsf{template} \mathord{<} \mathsf{typename} \ \mathsf{DISPATCH} >$

L4_NORETURN void internal_loop (DISPATCH dispatch)

The server loop.

11.170.1 Detailed Description

template<typename LOOP_HOOKS = lpc_svr::Default_loop_hooks>class L4::Server< LOOP_HOOKS >

Basic server loop for handling client requests.

Parameters

LOOP_HOOKS	the server inherits from LOOP_HOOKS and calls the hooks defined in LOOP_HOOKS in
	the server loop. See lpc_svr::Default_loop_hooks, lpc_svr::Ignore_errors, lpc_svr::Default
	timeout, lpc_svr::Compound_reply, and lpc_svr::Default_setup_wait.

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 183 of file ipc_server.

11.170.2 Constructor & Destructor Documentation

Initializes the server loop and its underlying lpc::lostream.

Parameters

```
utcb The UTCB of the thread running the server loop.
```

Definition at line 191 of file ipc_server.

11.170.3 Member Function Documentation

```
11.170.3.1 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 243 of file ipc server.

References I4_msgtag_t::has_error(), and L4_ENOREPLY.

Here is the call graph for this function:



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

· I4/cxx/ipc_server

11.171 L4::Server_object Class Reference

Abstract server object to be used with L4::Server and L4::Basic_registry.

Collaboration diagram for L4::Server_object:

+ Server_object() + dispatch() + ~Server_object() + obj_cap() + obj_cap() + obj_cap()

Public Member Functions

virtual int dispatch (unsigned long obj, lpc::lostream &ios)=0
 The abstract handler for client requests to the object.

11.171.1 Detailed Description

Abstract server object to be used with L4::Server and L4::Basic_registry.

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.

Examples:

 $examples/clntsrv/server.cc, examples/libs/l4re/c++/shared_ds/ds_srv.cc, and examples/libs/l4re/streammap/server.-cc.\\$

Definition at line 275 of file ipc server.

11.171.2 Member Function Documentation

11.171.2.1 virtual int L4::Server_object::dispatch (unsigned long obj, lpc::lostream & ios) [pure virtual]

The abstract handler for client requests to the object.

Parameters

obj	The object ID used for looking up the object.

ios The lpc::lostream for reading the request and writing the reply.

Returns

#Reply, #No_reply, or #Invalid_opcode.

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

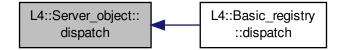
You need to extract the complete message from the *ios* stream before inserting any reply data or before doing any function call that may use the UTCB. Otherwise, the incoming message may get lost.

Examples:

 $examples/clntsrv/server.cc, examples/libs/l4re/c++/shared_ds/ds_srv.cc, \\ and \\ examples/libs/l4re/streammap/server.cc.$

Referenced by L4::Basic_registry::dispatch().

Here is the caller graph for this function:



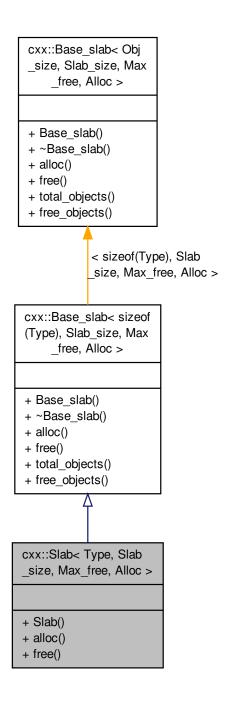
The documentation for this class was generated from the following file:

I4/cxx/ipc_server

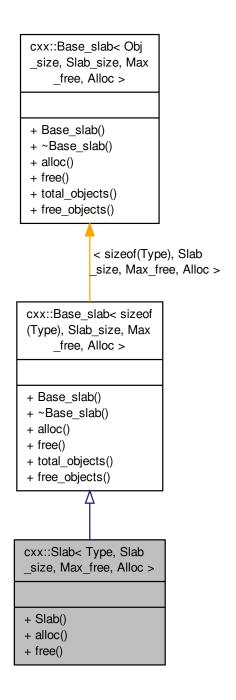
11.172 cxx::Slab < Type, Slab_size, Max_free, Alloc > Class Template Reference

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

11.172.1 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

Туре	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.

11.172.2 Member Function Documentation

11.172.2.1 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.

Free the object addressed by o.

Parameters

0	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.

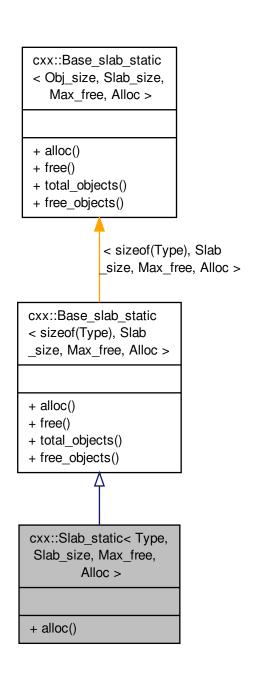
The documentation for this class was generated from the following file:

• I4/cxx/slab_alloc

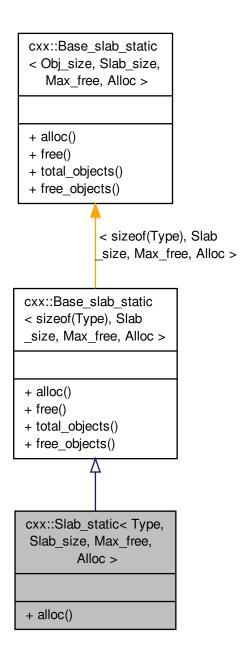
11.173 cxx::Slab_static < Type, Slab_size, Max_free, Alloc > Class Template Reference

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.

11.173.1 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

Туре	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.

11.173.2 Member Function Documentation

11.173.2.1 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.

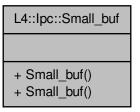
The documentation for this class was generated from the following file:

• I4/cxx/slab_alloc

11.174 L4::lpc::Small buf Class Reference

A receive item for receiving a single capability.

Collaboration diagram for L4::lpc::Small buf:



11.174.1 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 257 of file ipc_stream.

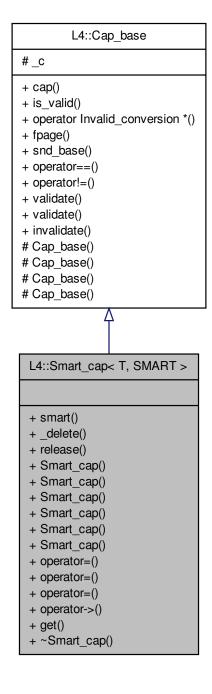
The documentation for this class was generated from the following file:

• I4/cxx/ipc_stream

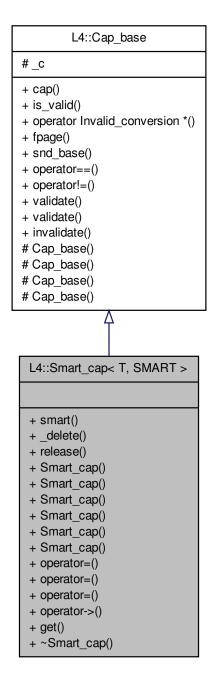
11.175 L4::Smart_cap < T, SMART > Class Template Reference

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

11.175.1 Detailed Description

template<typename T, typename SMART>class L4::Smart_cap< T, SMART>

Smart capability class.

Definition at line 36 of file smart capability.

11.175.2 Constructor & Destructor Documentation

```
11.175.2.1 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 67 of file smart capability.

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

· I4/sys/smart_capability

11.176 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 >:

L4Re::Util::Smart_cap
_auto< Unmap_flags >

+ free()
+ invalidate()
+ copy()

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.

11.176.1 Detailed Description

 $template < unsigned\ long\ Unmap_flags = L4_FP_ALL_SPACES > class\ L4Re::Util::Smart_cap_auto < Unmap_flags > class\ Unmap_flags$

Helper for Auto_cap and Auto_del_cap.

Definition at line 58 of file cap_alloc.

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

• I4/re/util/cap_alloc

11.177 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 >:

L4Re::Smart_cap_auto < Unmap_flags > + Smart_cap_auto() + Smart_cap_auto() + free()

+ invalidate()
+ copy()

11.177.1 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 110 of file cap_alloc.

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

· I4/re/cap_alloc

11.178 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 >:

L4Re::Util::Smart_count _cap< Unmap_flags >

- + free()
- + invalidate()
- + copy()

Static Public Member Functions

static void free (L4::Cap_base &c)

free operation for L4::Smart_cap (decrement ref count and delete if 0).

• 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 (increment ref count).

11.178.1 Detailed Description

 $template < unsigned\ long\ Unmap_flags = L4_FP_ALL_SPACES > class\ L4Re::Util::Smart_count_cap < Unmap_flags > class\ Unmap_fla$

Helper for Ref_cap and Ref_del_cap.

Definition at line 98 of file cap_alloc.

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

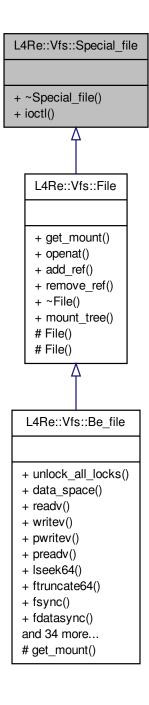
• I4/re/util/cap_alloc

11.179 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:

L4Re::Vfs::Special_file

+ ~Special_file()
+ ioctl()

Public Member Functions

virtual int ioctl (unsigned long cmd, va_list args)=0 throw ()
 The famous IO control.

11.179.1 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 395 of file vfs.h.

11.179.2 Member Function Documentation

11.179.2.1 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:

• I4/I4re_vfs/vfs.h

11.180 L4vcpu::State Class Reference

C++ implementation of state word in the vCPU area.

Collaboration diagram for L4vcpu::State:

L4vcpu::State + State()

- + State()
- + add()
- + clear()
- + set()

Public Member Functions

• State (I4vcpu_state_t v)

Initialize state.

• void add (unsigned bits) throw ()

Add flags.

• void clear (unsigned bits) throw ()

Clear flags.

void set (l4vcpu_state_t v) throw () Set flags.

11.180.1 Detailed Description

C++ implementation of state word in the vCPU area.

Definition at line 31 of file vcpu.

11.180.2 Constructor & Destructor Documentation

```
11.180.2.1 L4vcpu::State::State ( l4vcpu_state_t v ) [inline], [explicit]
```

Initialize state.

Parameters

```
v Initial state.
```

Definition at line 41 of file vcpu.

11.180.3 Member Function Documentation

```
11.180.3.1 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.

11.180.3.2 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.

11.180.3.3 void L4vcpu::State::set (l4vcpu_state_t 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:

• I4/vcpu/vcpu

11.181 L4Re::Dataspace::Stats Struct Reference

Information about the data space.

Collaboration diagram for L4Re::Dataspace::Stats:

L4Re::Dataspace::Stats
+ size
+ flags

Data Fields

• unsigned long size

size

· unsigned long flags

flags

11.181.1 Detailed Description

Information about the data space.

Definition at line 93 of file dataspace.

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

• I4/re/dataspace

11.182 L4::String Class Reference

A null-terminated string container class.

#include <string.h>

Collaboration diagram for L4::String:

+ String() + length() + p_str()

11.182.1 Detailed Description

A null-terminated string container class.

Definition at line 35 of file string.h.

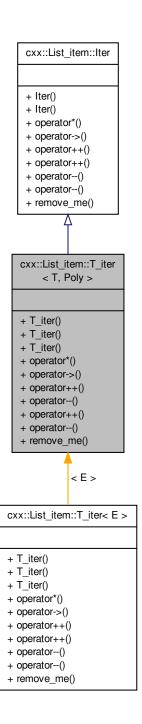
The documentation for this class was generated from the following file:

• I4/cxx/string.h

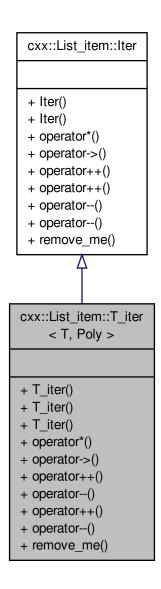
11.183 cxx::List_item::T_iter< T, Poly > Class Template Reference

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

11.183.1 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 \ast operator.

Example: class Foo : public ListItem { public: typedef T_iter<Foo> Iter; ... };

Definition at line 119 of file list.

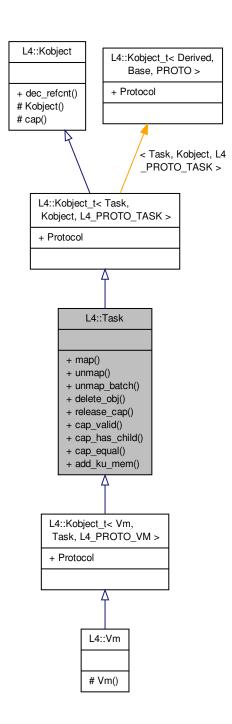
The documentation for this class was generated from the following file:

• I4/cxx/list

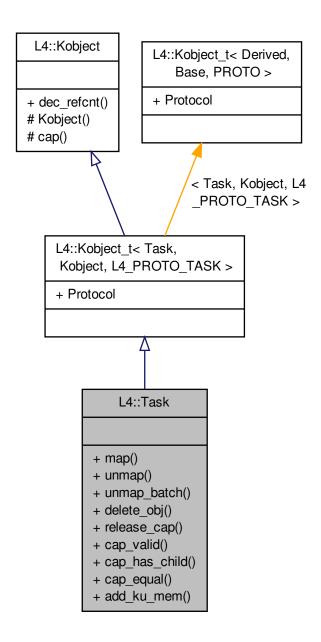
11.184 L4::Task Class Reference

An L4 Task.

Inheritance diagram for L4::Task:



Collaboration diagram for L4::Task:



Public Member Functions

I4_msgtag_t map (Cap< Task > const &src_task, I4_fpage_t const &snd_fpage, I4_addr_t snd_base, I4_utcb_t *utcb=I4_utcb()) throw ()

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

- I4_msgtag_t unmap (I4_fpage_t const &fpage, I4_umword_t map_mask, I4_utcb_t *utcb=I4_utcb()) throw ()

 Revoke rights from the task.
- I4_msgtag_t unmap_batch (I4_fpage_t const *fpages, unsigned num_fpages, I4_umword_t map_mask, I4_utcb_t *utcb=I4_utcb()) throw ()

Revoke rights from a task.

I4_msgtag_t delete_obj (L4::Cap< void > obj, I4_utcb_t *utcb=I4_utcb()) throw ()

Release capability and delete object.

I4_msgtag_t release_cap (L4::Cap< void > cap, I4_utcb_t *utcb=I4_utcb()) throw ()

Release capability.

• I4_msgtag_t cap_valid (Cap< void > const &cap, I4_utcb_t *utcb=I4_utcb()) throw ()

Test whether a capability selector points to a valid capability.

• I4_msgtag_t cap_has_child (Cap< void > const &cap, I4_utcb_t *utcb=I4_utcb()) throw ()

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

I4_msgtag_t cap_equal (Cap< void > const &cap_a, Cap< void > const &cap_b, I4_utcb_t *utcb=I4_utcb())
 throw ()

Test whether two capabilities point to the same object with the same rights.

• I4_msgtag_t add_ku_mem (I4_fpage_t const &fpage, I4_utcb_t *utcb=I4_utcb()) throw ()

Add kernel-user memory.

Additional Inherited Members

11.184.1 Detailed Description

An L4 Task.

```
#include <14/sys/task>
```

See Also

Task for an overview description.

Definition at line 41 of file task.

11.184.2 Member Function Documentation

```
11.184.2.1 I4_msgtag_t L4::Task::map ( Cap < Task > const & src_task, I4_fpage_t const & snd_fpage, I4_addr_t snd_base, I4_utcb_t * utcb = I4_utcb () ) throw) [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.

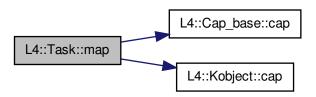
Note

dst_task is the implicit this pointer.

Definition at line 50 of file task.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



11.184.2.2 I4_msgtag_t L4::Task::unmap(I4_fpage_t const & fpage, I4_umword_t map_mask, I4_utcb_t * utcb = I4_utcb()) throw) [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 I4_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.

task is the implicit this pointer.

Definition at line 59 of file task.

References L4::Kobject::cap().

Here is the call graph for this function:



11.184.2.3 I4_msgtag_t L4::Task::unmap_batch (I4_fpage_t const * fpages, unsigned num_fpages, I4_umword_t map_mask, I4_utcb t * utcb = I4_utcb ()) throw) [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.

task is the implicit this pointer.

Definition at line 68 of file task.

References L4::Kobject::cap().

Here is the call graph for this function:



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 is operating calls I4_task_unmap() with L4_FP_DELETE_OBJ.

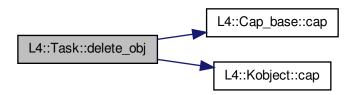
Note

task is the implicit this pointer.

Definition at line 78 of file task.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



Release capability.

Parameters

task	Capability selector of destination task
сар	Capability selector to release

Returns

Syscall return tag

This operation unmaps the capability from the specified task.

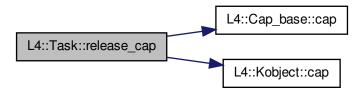
Note

task is the implicit this pointer.

Definition at line 86 of file task.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



Test whether a capability selector points to a valid capability.

Parameters

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

Returns

label contains >0 if valid, 0 if invalid

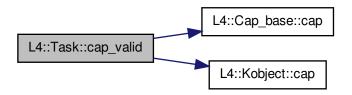
Note

task is the implicit this pointer.

Definition at line 94 of file task.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



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

Parameters

task	Capability selector of the destination task to do the lookup in
сар	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

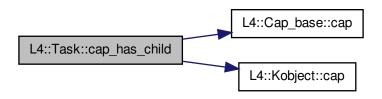
Note

task is the implicit this pointer.

Definition at line 102 of file task.

 $References\ L4:: Cap_base:: cap(),\ and\ L4:: Kobject:: cap().$

Here is the call graph for this function:



11.184.2.8 I4_msgtag_t L4::Task::cap_equal (Cap< void > const & cap_a, Cap< void > const & cap_b, I4_utcb_t * utcb = I4_utcb()) throw) [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
ĺ	cap_a	Capability selector to compare
ĺ	cap_b	Capability selector to compare

Returns

label contains 1 if equal, 0 if not equal

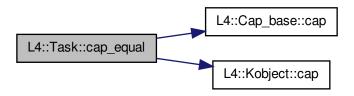
Note

task is the implicit this pointer.

Definition at line 110 of file task.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



11.184.2.9 I4_msgtag_t L4::Task::add_ku_mem (I4_fpage_t const & fpage, I4_utcb_t * utcb = I4_utcb ()) throw) [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

Note

task is the implicit this pointer.

Definition at line 119 of file task.

References L4::Kobject::cap().

Here is the call graph for this function:



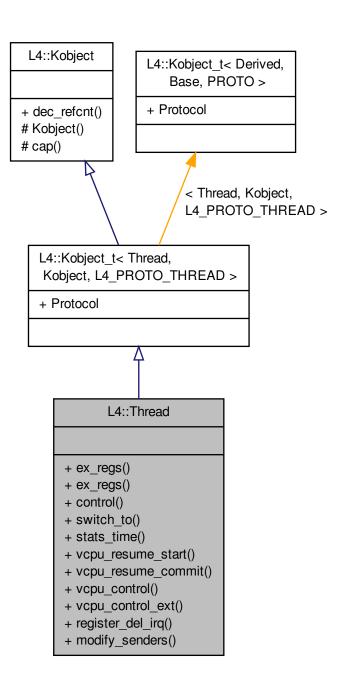
The documentation for this class was generated from the following file:

• I4/sys/task

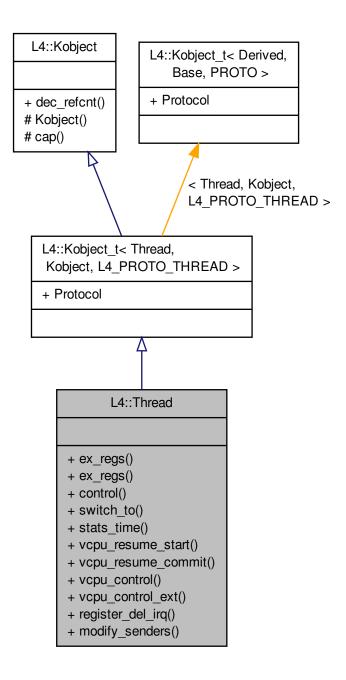
11.185 L4::Thread Class Reference

L4 kernel thread.

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

- I4_msgtag_t ex_regs (I4_addr_t ip, I4_addr_t sp, I4_umword_t flags, I4_utcb_t *utcb=I4_utcb()) throw () Exchange basic thread registers.
- I4_msgtag_t ex_regs (I4_addr_t *ip, I4_addr_t *sp, I4_umword_t *flags, I4_utcb_t *utcb=I4_utcb()) throw () Exchange basic thread registers and return previous values.
- I4_msgtag_t control (Attr const & attr) throw ()

Commit the given thread-attributes object.

• I4_msgtag_t switch_to (I4_utcb_t *utcb=I4_utcb()) throw ()

Switch execution to this thread.

I4_msgtag_t stats_time (I4_utcb_t *utcb=I4_utcb()) throw ()

Get consumed timed of thread in ns.

• I4_msgtag_t vcpu_resume_start (I4_utcb_t *utcb=I4_utcb()) throw ()

vCPU resume, start.

- I4_msgtag_t vcpu_resume_commit (I4_msgtag_t tag, I4_utcb_t *utcb=I4_utcb()) throw ()
 vCPU resume, commit.
- $\bullet \ \ \mathsf{I4_msgtag_t} \ \mathsf{vcpu_control} \ (\mathsf{I4_addr_t} \ \mathsf{vcpu_state}, \ \mathsf{I4_utcb_t} \ *\mathsf{utcb=I4_utcb}()) \ \mathsf{throw} \ ()$

Enable or disable the vCPU feature for the thread.

I4_msgtag_t vcpu_control_ext (I4_addr_t ext_vcpu_state, I4_utcb_t *utcb=I4_utcb()) throw ()

Enable or disable the extended vCPU feature for the thread.

• $I4_msgtag_t register_del_irq (Cap < Irq > irq, I4_utcb_t *u=I4_utcb()) throw ()$

Register an IRQ that will trigger upon deletion events.

I4_msgtag_t modify_senders (Modify_senders const &todo) throw ()

Apply sender modifiction rules.

Additional Inherited Members

11.185.1 Detailed Description

L4 kernel thread.

#include <14/sys/thread>

Definition at line 38 of file thread.

11.185.2 Member Function Documentation

```
11.185.2.1 I4_msgtag_t L4::Thread::ex_regs ( I4_addr_t ip, I4_addr_t sp, I4_umword_t flags, I4_utcb_t * utcb = I4_utcb () ) throw) [inline]
```

Exchange basic thread registers.

Parameters

thread	Thread to manipulate
ip	New instruction pointer, use $\sim\!\!$ 0UL to leave the instruction pointer unchanged
sp	New stack pointer, use $\sim\!\!$ 0UL to leave the stack pointer unchanged
flags	Ex-regs flags, see L4_thread_ex_regs_flags

Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see *flags*).

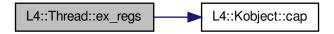
Note

the thread argument is the implicit this pointer.

Definition at line 47 of file thread.

References L4::Kobject::cap().

Here is the call graph for this function:



11.185.2.2 I4_msgtag_t L4::Thread::ex_regs (I4_addr_t *
$$ip$$
, I4_addr_t * sp , I4_umword_t * $flags$, I4_utcb_t * $utcb$ = I4_utcb()) throw) [inline]

Exchange basic thread registers and return previous values.

Parameters

in	thread	Thread to manipulate
in,out	ip	New instruction pointer, use $\sim\!\!0\text{UL}$ to leave the instruction pointer unchanged,
		return previous instruction pointer
in,out	sp	New stack pointer, use $\sim\!\!0 \text{UL}$ 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.

Note

the thread argument is the implicit this pointer.

Definition at line 56 of file thread.

References L4::Kobject::cap().

Here is the call graph for this function:



11.185.2.3 I4_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 152 of file thread.

References L4::Kobject::cap().

Here is the call graph for this function:



11.185.2.4 I4_msgtag_t L4::Thread::switch_to (I4_utcb_t * utcb = I4_utcb ()) throw) [inline]

Switch execution to this thread.

Parameters

utcb	the UTCB of the current thread.

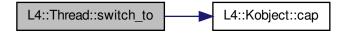
Note

The current time slice is inherited to this thread.

Definition at line 161 of file thread.

References L4::Kobject::cap().

Here is the call graph for this function:



```
11.185.2.5 I4_msgtag_t L4::Thread::stats_time( I4_utcb_t * utcb = I4_utcb() ) throw) [inline]
```

Get consumed timed of thread in ns.

Parameters

```
utcb the UTCB of the current thread.
```

The consumed time is return as I4_kernel_clock_t at UTCB message register 0.

Definition at line 171 of file thread.

References L4::Kobject::cap().

Here is the call graph for this function:



```
11.185.2.6 I4_msgtag_t L4::Thread::vcpu_resume_start( I4_utcb_t * utcb = I4_utcb() ) throw) [inline]
```

vCPU resume, start.

See Also

14 thread vcpu resume start

Definition at line 179 of file thread.

```
11.185.2.7 I4_msgtag_t L4::Thread::vcpu_resume_commit ( I4_msgtag_t tag, I4_utcb_t * utcb = I4_utcb () ) throw) [inline]
```

vCPU resume, commit.

See Also

14_thread_vcpu_resume_commit

Definition at line 187 of file thread.

References L4::Kobject::cap().

Here is the call graph for this function:



11.185.2.8 I4_msgtag_t L4::Thread::vcpu_control (I4_addr_t vcpu_state, I4_utcb_t * utcb = I4_utcb ()) throw)
[inline]

Enable or disable the vCPU feature for the thread.

Parameters

thread	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.

Returns

Systemcall result message 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.

Definition at line 194 of file thread.

References L4::Kobject::cap().

Here is the call graph for this function:



11.185.2.9 I4_msgtag_t L4::Thread::vcpu_control_ext(I4_addr_t ext_vcpu_state, I4_utcb_t * utcb = I4_utcb())
throw) [inline]

Enable or disable the extended vCPU feature for the thread.

Parameters

thread	The thread for which the extended 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.

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 *vcpu_state* is set to a valid kernel-user-memory address, or disables the vCPU feature if *vcpu_state* is 0.

Definition at line 201 of file thread.

References L4::Kobject::cap().

Here is the call graph for this function:



Register an IRQ that will trigger upon deletion events.

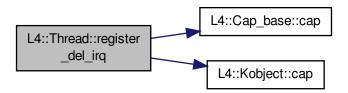
See Also

I4_thread_register_del_irq

Definition at line 210 of file thread.

References L4::Cap_base::cap(), and L4::Kobject::cap().

Here is the call graph for this function:



11.185.2.11 I4_msgtag_t L4::Thread::modify_senders (Modify_senders const & todo) throw) [inline]

Apply sender modifiction rules.

Parameters

todo	Prepared sender modification rules.
------	-------------------------------------

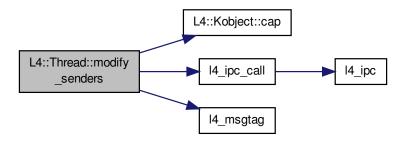
Returns

System call return tag.

Definition at line 270 of file thread.

References L4::Kobject::cap(), I4_ipc_call(), L4_IPC_NEVER, I4_msgtag(), and L4_PROTO_THREAD.

Here is the call graph for this function:



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

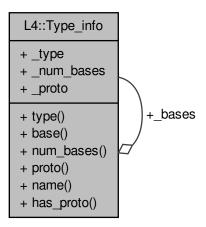
• I4/sys/thread

11.186 L4::Type_info Struct Reference

Dynamic Type Information for L4Re Interfaces.

#include <__typeinfo.h>

Collaboration diagram for L4::Type_info:



11.186.1 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 50 of file __typeinfo.h.

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

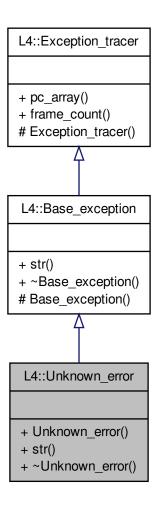
• I4/sys/__typeinfo.h

11.187 L4::Unknown_error Class Reference

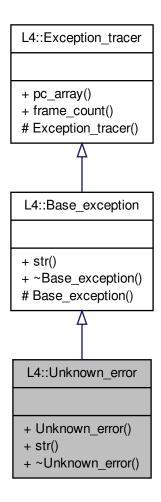
Exception for an unknown condition.

#include <14/cxx/exceptions>

Inheritance diagram for L4::Unknown_error:



Collaboration diagram for L4::Unknown_error:



Public Member Functions

char const * str () const throw ()
 Should return a human readable string for the exception.

Additional Inherited Members

11.187.1 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 202 of file exceptions.

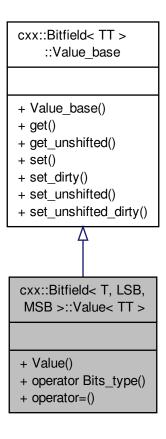
The documentation for this class was generated from the following file:

I4/cxx/exceptions

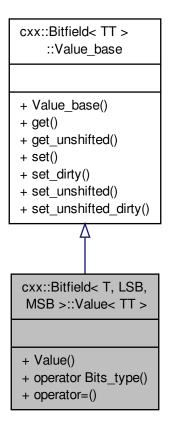
11.188 cxx::Bitfield< T, LSB, MSB >::Value< TT > Class Template Reference

Internal helper type.

Inheritance diagram for cxx::Bitfield< T, LSB, MSB >::Value< TT >:



Collaboration diagram for cxx::Bitfield < T, LSB, MSB >::Value < TT >:



11.188.1 Detailed Description

 $\label{template} \textbf{template} \small < \textbf{typename TT} \small > \textbf{class cxx::} \\ \textbf{Bitfield} \small < \textbf{T, LSB, MSB} \small > :: \textbf{Value} \small < \textbf{TT} \\ \small > \\ \\ \textbf{TT} B \small > \\ \\ \textbf{TT}$

Internal helper type.

Definition at line 199 of file bitfield.

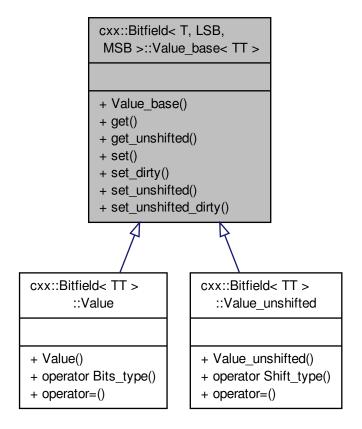
The documentation for this class was generated from the following file:

· I4/cxx/bitfield

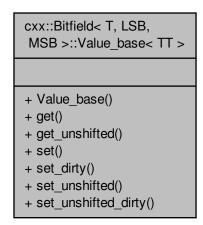
11.189 cxx::Bitfield < T, LSB, MSB >::Value_base < TT > Class Template Reference

Internal helper type.

Inheritance diagram for cxx::Bitfield< T, LSB, MSB >::Value_base< TT >:



 $Collaboration \ diagram \ for \ cxx::Bitfield < T, LSB, \ MSB > ::Value_base < TT > :$



11.189.1 Detailed Description

 $template < typename\ TT > class\ cxx:: Bitfield < T,\ LSB,\ MSB > :: Value_base < TT >$

Internal helper type.

Definition at line 181 of file bitfield.

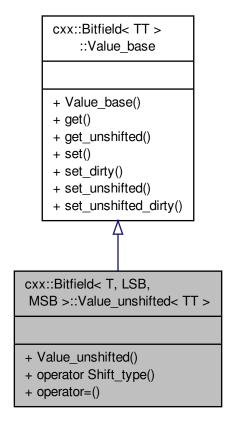
The documentation for this class was generated from the following file:

I4/cxx/bitfield

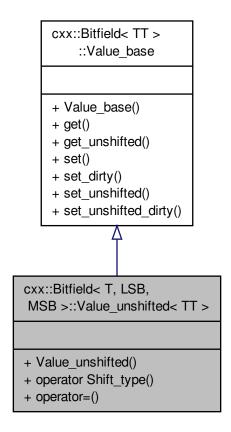
11.190 cxx::Bitfield < T, LSB, MSB >::Value_unshifted < TT > Class Template Reference

Internal helper type.

Inheritance diagram for cxx::Bitfield < T, LSB, MSB >::Value_unshifted < TT >:



 $Collaboration\ diagram\ for\ cxx::Bitfield<\ T,\ LSB,\ MSB>::Value_unshifted<\ TT>:$



11.190.1 Detailed Description

 $template < typename\ TT > class\ cxx:: Bitfield < T,\ LSB,\ MSB > :: Value_-unshifted < TT >$

Internal helper type.

Definition at line 209 of file bitfield.

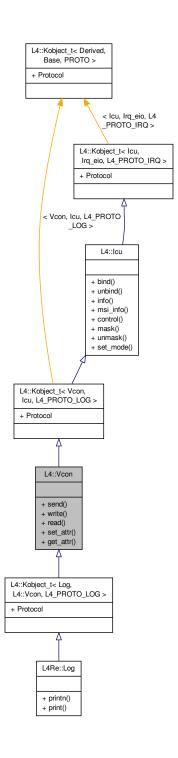
The documentation for this class was generated from the following file:

I4/cxx/bitfield

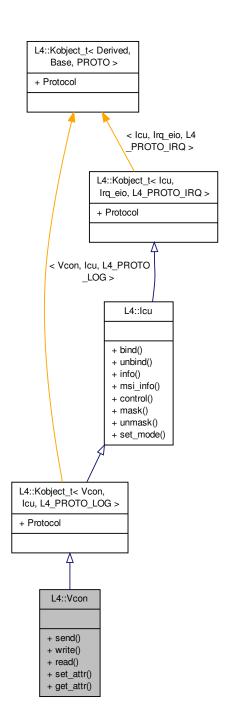
11.191 L4::Vcon Class Reference

C++ L4 Vcon.

Inheritance diagram for L4::Vcon:



Collaboration diagram for L4::Vcon:



Public Member Functions

- I4_msgtag_t send (char const *buf, int size, I4_utcb_t *utcb=I4_utcb()) const throw ()
 Send data to virtual console.
- long write (char const *buf, int size, I4_utcb_t *utcb=I4_utcb()) const throw ()
 Write data to virtual console.
- int read (char *buf, int size, I4_utcb_t *utcb=I4_utcb()) const throw ()

Read data from virtual console.

- I4_msgtag_t set_attr (I4_vcon_attr_t const *attr, I4_utcb_t *utcb=I4_utcb()) const throw () Set attributes of a Vcon.
- I4_msgtag_t get_attr (I4_vcon_attr_t *attr, I4_utcb_t *utcb=I4_utcb()) const throw () Get attributes of a Vcon.

Additional Inherited Members

11.191.1 Detailed Description

C++ L4 Vcon.

#include <14/sys/vcon>

See Also

Virtual Console for an overview and C bindings.

Definition at line 41 of file vcon.

11.191.2 Member Function Documentation

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. *vcon* is the implicit *this* pointer.

Definition at line 52 of file vcon.

References L4::Kobject::cap().

Here is the call graph for this function:



11.191.2.2 long L4::Vcon::write (char const * buf, int size, I4_utcb_t * utcb = I4_utcb ()) const throw) [inline]

Write data to virtual console.

Parameters

vcon	Vcon object.
buf	Pointer to data buffer.
size	Size of buffer in bytes.

Returns

Number of bytes written to the virtual console.

Note

vcon is the implicit this pointer.

Definition at line 60 of file vcon.

References L4::Kobject::cap().

Here is the call graph for this function:



11.191.2.3 int L4::Vcon::read (char * buf, int size, I4_utcb_t * utcb = I4_utcb ()) const throw) [inline]

Read data from virtual console.

Parameters

vcon	Vcon object.
buf	Pointer to data buffer.
size	Size of buffer in bytes.

Returns

Negative error code on error, > size if more to read, size bytes are in the buffer, <= size bytes read

Note

vcon is the implicit this pointer.

Definition at line 68 of file vcon.

References L4::Kobject::cap().

Here is the call graph for this function:



11.191.2.4 I4_msgtag_t L4::Vcon::set_attr(I4_vcon_attr_t const * attr, I4_utcb_t * utcb = I4_utcb()) const throw)
[inline]

Set attributes of a Vcon.

Parameters

vcon	Vcon object.
attr	Attribute structure.

Returns

Syscall return tag

Note

vcon is the implicit this pointer.

Definition at line 76 of file vcon.

References L4::Kobject::cap().

Here is the call graph for this function:



11.191.2.5 I4_msgtag_t L4::Vcon::get_attr(I4_vcon_attr_t * attr, I4_utcb_t * utcb = I4_utcb()) const throw)
[inline]

Get attributes of a Vcon.

Parameters

vcon	Vcon object.
------	--------------

Return values

attr	Attribute structure.

Returns

Syscall return tag

Note

vcon is the implicit this pointer.

Definition at line 84 of file vcon.

References L4::Kobject::cap().

Here is the call graph for this function:



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

• I4/sys/vcon

11.192 L4Re::Util::Vcon_svr< SVR > Class Template Reference

Console server template class.

Collaboration diagram for L4Re::Util::Vcon_svr< SVR >:

+ dispatch()
+ vcon_read()
+ vcon_write()
+ vcon_get_attr()

Public Member Functions

int dispatch (I4_umword_t obj, L4::lpc::lostream &ios)
 Server dispatch function.

11.192.1 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 return a value bigger than the given size to indicate that there's more data 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 function is given in bytes.

Definition at line 43 of file vcon svr.

11.192.2 Member Function Documentation

11.192.2.1 template < typename SVR > int L4Re::Util::Vcon_svr < SVR >::dispatch (I4_umword_t obj, L4::Ipc::lostream & ios)

Server dispatch function.

Parameters

obj	Server object ID to work on.
ios	Input/Output stream.

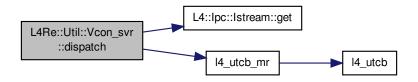
Returns

error code.

Definition at line 74 of file vcon_svr.

References L4::lpc::lstream::get(), L4_EBADPROTO, L4_ENOREPLY, L4_EOK, L4_PROTO_LOG, L4_UTCB_G-ENERIC_DATA_SIZE, I4_utcb_mr(), L4_VCON_GET_ATTR_OP, L4_VCON_SET_ATTR_OP, L4_VCON_WRIT-E_OP, and I4_msg_regs_t::mr.

Here is the call graph for this function:



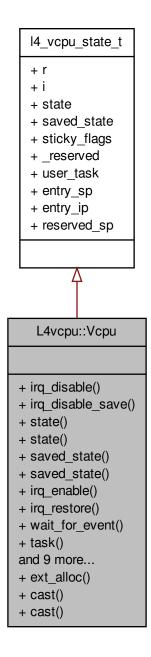
The documentation for this class was generated from the following file:

• I4/re/util/vcon svr

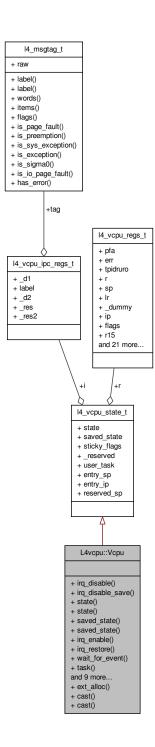
11.193 L4vcpu::Vcpu Class Reference

C++ implementation of the vCPU save state area.

Inheritance diagram for L4vcpu::Vcpu:



Collaboration diagram for L4vcpu::Vcpu:



Public Types

• typedef l4vcpu_irq_state_t lrq_state

IRQ status type.

Public Member Functions

```
    void irq_disable () throw ()

      Disable the vCPU for event delivery.
• Irq_state 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.
• void irg_enable (I4_utcb_t *utcb, I4vcpu_event_hndl_t do_event_work_cb, I4vcpu_setup_ipc_t setup_ipc)
  throw ()
      Enable the vCPU for event delivery.

    void irg_restore (Irg_state s, I4_utcb_t *utcb, I4vcpu_event_hndl_t do_event_work_cb, I4vcpu_setup_ipc_t

  setup_ipc) throw ()
      Restore a previously saved IRQ/event state.

    void wait_for_event (I4_utcb_t *utcb, I4vcpu_event_hndl_t do_event_work_cb, I4vcpu_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 ()
      Return whether the entry reason was a page fault.

    int is irq entry ()

      Return whether the entry reason was an IRQ/IPC message.

    I4_vcpu_regs_t * r () throw ()

      Return pointer to register state.
• 14 vcpu regs t const * r () const throw ()
      Return pointer to register state.
• I4_vcpu_ipc_regs_t * i () throw ()
      Return pointer to IPC state.

    I4_vcpu_ipc_regs_t const * i () const throw ()

      Return pointer to IPC state.
```

void entry_sp (I4_umword_t sp)

Set vCPU entry stack pointer.

void entry_ip (I4_umword_t ip)

Set vCPU entry instruction pointer.

void print_state (const char *prefix="") throw ()

Print the state of the vCPU.

Static Public Member Functions

```
    static int ext_alloc (Vcpu **vcpu, I4_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 (I4_addr_t x) throw ()

Cast an address to a class pointer.

11.193.1 Detailed Description

C++ implementation of the vCPU save state area.

Definition at line 72 of file vcpu.

11.193.2 Member Function Documentation

```
11.193.2.1 lrq_state 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 90 of file vcpu.

References I4vcpu_irq_disable_save().

Here is the call graph for this function:



```
11.193.2.2 State* L4vcpu::Vcpu::state()throw) [inline]
```

Get state word.

Returns

Pointer to state word in the vCPU

Definition at line 97 of file vcpu.

References I4_vcpu_state_t::state.

11.193.2.3 State L4vcpu::Vcpu::state()const throw) [inline]

Get state word.

Returns

Pointer to state word in the vCPU

Definition at line 108 of file vcpu.

References I4_vcpu_state_t::state.

11.193.2.4 State* L4vcpu::Vcpu::saved_state()throw) [inline]

Get saved_state word.

Returns

Pointer to saved_state word in the vCPU

Definition at line 115 of file vcpu.

References I4 vcpu state t::saved state.

11.193.2.5 State L4vcpu::Vcpu::saved_state() const throw) [inline]

Get saved_state word.

Returns

Pointer to saved_state word in the vCPU

Definition at line 125 of file vcpu.

References I4_vcpu_state_t::saved_state.

11.193.2.6 void L4vcpu::Vcpu::irq_enable (I4_utcb_t * utcb, I4vcpu_event_hndl_t do_event_work_cb, I4vcpu_setup_ipc_t setup_ipc) throw) [inline]

Enable the vCPU for event delivery.

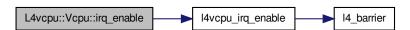
Parameters

utcb	The UTCB to use.
do_event_work-	Call-back function that is called in case an event (such as an interrupt) is pending.
_cb	
setup_ipc	Call-back function that is called before an IPC operation is called, and before event delivery
	is enabled.

Definition at line 138 of file vcpu.

References I4vcpu_irq_enable().

Here is the call graph for this function:



11.193.2.7 void L4vcpu::Vcpu::irq_restore (Irq_state s, I4_utcb_t * utcb, I4vcpu_event_hndl_t do_event_work_cb, I4vcpu_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-	Call-back function that is called in case an event (such as an interrupt) is pending.
_cb	
setup_ipc	Call-back function that is called before an IPC operation is called, and before event delivery
	is enabled.

Definition at line 153 of file vcpu.

References I4vcpu_irq_restore().

Here is the call graph for this function:



11.193.2.8 void L4vcpu::Vcpu::wait_for_event (I4_utcb_t * utcb, I4vcpu_event_hndl_t do_event_work_cb, I4vcpu_setup_ipc_t setup_ipc) throw) [inline]

Wait for event.

Parameters

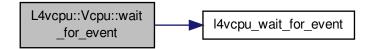
utcb	The UTCB to use.
do_event_work-	Call-back function that is called in case an event (such as an interrupt) is pending.
_cb	
setup_ipc	Call-back function that is called before an IPC operation is called.

Note that event delivery remains disabled after this function returns.

Definition at line 169 of file vcpu.

References I4vcpu_wait_for_event().

Here is the call graph for this function:



11.193.2.9 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 177 of file vcpu.

References I4_vcpu_state_t::user_task.

11.193.2.10 int L4vcpu::Vcpu::is_page_fault_entry() [inline]

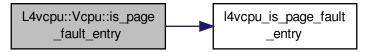
Return whether the entry reason was a page fault.

return 0 if not, !=0 otherwise.

Definition at line 184 of file vcpu.

References I4vcpu_is_page_fault_entry().

Here is the call graph for this function:



11.193.2.11 int L4vcpu::Vcpu::is_irq_entry() [inline]

Return whether the entry reason was an IRQ/IPC message.

return 0 if not, !=0 otherwise.

Definition at line 191 of file vcpu.

References I4vcpu_is_irq_entry().

Here is the call graph for this function:



11.193.2.12 I4_vcpu_regs_t* L4vcpu::Vcpu::r()throw) [inline]

Return pointer to register state.

```
Returns
```

```
Pointer to register state.
```

```
Definition at line 198 of file vcpu.
```

```
References I4_vcpu_state_t::r.
```

```
11.193.2.13 I4_vcpu_regs_t const* L4vcpu::Vcpu::r( ) const throw) [inline]
```

Return pointer to register state.

Returns

Pointer to register state.

Definition at line 205 of file vcpu.

References I4_vcpu_state_t::r.

Return pointer to IPC state.

Returns

Pointer to IPC state.

Definition at line 212 of file vcpu.

References I4_vcpu_state_t::i.

```
11.193.2.15 I4_vcpu_ipc_regs_t const* L4vcpu::Vcpu::i( ) const throw) [inline]
```

Return pointer to IPC state.

Returns

Pointer to IPC state.

Definition at line 219 of file vcpu.

References I4_vcpu_state_t::i.

```
11.193.2.16 void L4vcpu::Vcpu::entry_sp( I4_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 228 of file vcpu.

References I4_vcpu_state_t::entry_sp.

11.193.2.17 void L4vcpu::Vcpu::entry_ip (I4_umword_t ip) [inline]

Set vCPU entry instruction pointer.

Parameters

ip	Instruction pointer address to set.
----	-------------------------------------

Definition at line 235 of file vcpu.

References I4_vcpu_state_t::entry_ip.

```
11.193.2.18 static int L4vcpu::Vcpu::ext_alloc ( Vcpu ** vcpu, I4_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

```
11.193.2.19 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 261 of file vcpu.

```
11.193.2.20 static Vcpu* L4vcpu::Vcpu::cast(I4_addr_t x) throw) [inline], [static]
```

Cast an address to a class pointer.

Parameters

X	Pointer.

Returns

Pointer to Vcpu class.

Definition at line 271 of file vcpu.

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

• I4/vcpu/vcpu

11.194 L4Re::Video::View Class Reference

View.

Collaboration diagram for L4Re::Video::View:

+ View() + info() + set_info() + set_viewport() + stack() + push_top() + push_bottom() + refresh() + valid()

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.

Friends

class Goos

ID for goos objects.

11.194.1 Detailed Description

View.

Definition at line 34 of file view.

11.194.2 Member Enumeration Documentation

```
11.194.2.1 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 53 of file view.

```
11.194.2.2 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 76 of file view.

11.194.3 Member Function Documentation

11.194.3.1 int L4Re::Video::View::info (Info * info) const throw)

Return the view information of the view.

Return values

info	Information structure pointer.

Returns

0 on success, error otherwise

11.194.3.2 int L4Re::Video::View::set_info (Info const & info) const throw)

Set the information structure for this view.

Parameters

info	Information structure.
------	------------------------

Returns

0 on success, error otherwise

The function will also set the view port according to the values given in the information structure.

11.194.3.3 int L4Re::Video::View::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.

Parameters

so	X X position
SC	y Y position
	w Width
	h Height
buf of	offset in the buffer in bytes

Returns

0 on success, error otherwise

11.194.3.4 int L4Re::Video::View::stack (View const & pivot, bool behind = true) const throw)

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.

Returns

0 on success, error otherwise

11.194.3.5 int L4Re::Video::View::refresh (int x, int y, int w, int h) const throw)

Refresh/Redraw the view.

Parameters

X	X position.
У	Y position.
W	Width.
h	Height.

Returns

0 on success, error otherwise

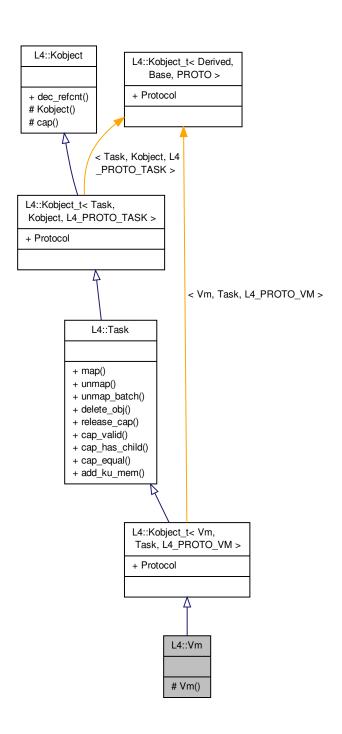
The documentation for this class was generated from the following file:

• I4/re/video/view

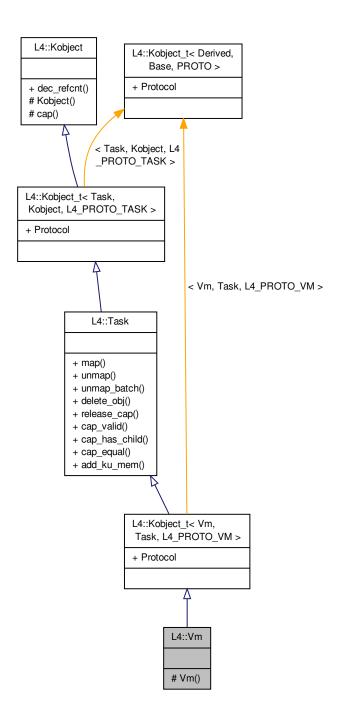
11.195 L4::Vm Class Reference

Virtual machine.

Inheritance diagram for L4::Vm:



Collaboration diagram for L4::Vm:



Additional Inherited Members

11.195.1 Detailed Description

Virtual machine.

Definition at line 36 of file __vm.

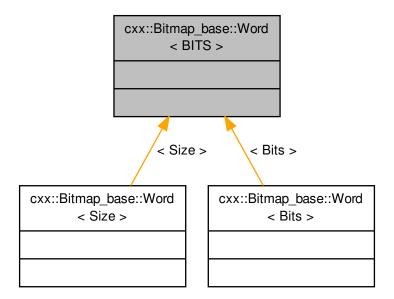
The documentation for this class was generated from the following file:

• I4/sys/__vm

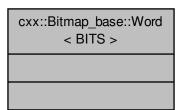
11.196 cxx::Bitmap_base::Word< BITS > Class Template Reference

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 >:



11.196.1 Detailed Description

 $template < long\ BITS > class\ cxx::Bitmap_base::Word < BITS >$

Helper abstraction for a word contained in the bitmap.

Definition at line 50 of file bitmap.

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

• I4/cxx/bitmap

Chapter 12

Example Documentation

12.1 examples/clntsrv/client.cc

Client/Server example using C++ infrastructure – Client implementation.

```
\star (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 <14/svs/err.h>
#include <14/sys/types.h>
#include <14/re/env>
#include <14/re/util/cap_alloc>
#include <14/cxx/ipc_stream>
#include <stdio.h>
#include "shared.h"
static int
func_neg_call(L4::Cap<void> const &server, 14_uint32_t *result,
              14_uint32_t val)
 L4::Ipc::Iostream s(14_utcb());
  s << 14_umword_t(Opcode::func_neg) << val;
  int r = 14_error(s.call(server.cap(), Protocol::Calc));
 return r; // failure
s >> *result;
  return 0; // ok
static int
func_sub_call(L4::Cap<void> const &server, 14_uint32_t *result,
              14_uint32_t val1, 14_uint32_t val2)
 L4::Ipc::Iostream s(14_utcb());
  s << 14_umword_t (Opcode::func_sub) << val1 << val2;
  int r = 14_error(s.call(server.cap(), Protocol::Calc));
  if (r)
  return r; // failure
s >> *result;
  return 0; // ok
main()
  L4::Cap<void> server = L4Re::Env::env()->get_cap<void>("calc_server");
  if (!server.is_valid())
     printf("Could not get server capability!\n");
      return 1;
  14_uint32_t val1 = 8;
```

```
14_uint32_t val2 = 5;
printf("Asking for %d - %d\n", val1, val2);
if (func_sub_call(server, &val1, val1, val2))
{
    printf("Error talking to server\n");
    return 1;
}
printf("Result of substract call: %d\n", val1);
printf("Asking for -%d\n", val1);
if (func_neg_call(server, &val1, val1))
{
    printf("Error talking to server\n");
    return 1;
}
printf("Result of negate call: %d\n", val1);
return 0;
```

12.2 examples/clntsrv/clntsrv.cfg

Sample configuration file for the client/server example.

12.3 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)
 \star This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 \star Please see the COPYING-GPL-2 file for details.
#include <stdio.h>
#include <14/re/env>
#include <14/re/util/cap_alloc>
#include <14/re/util/object_registry>
#include <14/cxx/ipc_server>
#include "shared.h"
static L4Re::Util::Registry_server<> server;
class Calculation_server : public L4::Server_object
public:
 int dispatch(14_umword_t obj, L4::Ipc::Iostream &ios);
```

```
Calculation_server::dispatch(14_umword_t, L4::Ipc::Iostream &ios)
  14_msgtag_t t;
  ios >> t;
  // We're only talking the calculation protocol
if (t.label() != Protocol::Calc)
    return -L4_EBADPROTO;
  L4::Opcode opcode;
  ios >> opcode;
  switch (opcode)
    case Opcode::func_neg:
      14_uint32_t val;
      ios >> val;
      val = -val;
      ios << val;
      return L4_EOK;
    case Opcode::func_sub:
      14_uint32_t val1, val2;
ios >> val1 >> val2;
      val1 -= val2;
      ios << vall;
       return L4_EOK;
    default:
      return -L4_ENOSYS;
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;
```

12.4 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)
 \star This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 \star Please see the COPYING-GPL-2 file for details.
#include <14/re/mem_alloc>
#include <14/re/rm>
#include <14/re/env>
#include <14/re/dataspace>
#include <14/re/util/cap_alloc>
#include <14/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 = 14_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, d, 0,
                                            flags & L4Re::Mem_alloc::Super_pages
                                              ? L4_SUPERPAGESHIFT :
      L4_PAGESHIFT)))
    return r;
  /* Done, virtual address is in virt_addr */
static int free_mem(void *virt_addr)
  L4::Cap<L4Re::Dataspace> ds;
  /\star Detach memory from our address space \star/
  if ((r = L4Re::Env::env()->rm()->detach(virt_addr, &ds)))
  /\star Free memory at our memory allocator, this is optional \star/
  if ((r = L4Re::Env::env()->mem_alloc()->free(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;
```

12.5 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 <14/re/util/cap_alloc> // L4::Cap
#include <14/re/dataspace> // L4Re::Dataspace
#include <14/re/rm> // L4::Rm
#include <14/re/env>
                                  // L4::Env
#include <14/sys/cache.h>
#include <cstdio>
#include <unistd.h>
#include "interface.h"
int main()
  /*
   \star 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");
  * Request shared data-space cap.
    (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, ds);
     printf("Error attaching data space: %s\n", 14sys_errtostr(err));
      return 1;
 printf("Content: %s\n", addr);
  // wait a bit for the demo effect
 sleep(3);
  * Fill in new stuff
 memset(addr, 0, ds->size()); char const \star const msg = "Hello from client, too!"; printf("Setting new content in shared memory\n");
  snprintf(addr, strlen(msg)+1, msg);
  14_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");
L4Re::Util::cap_alloc.free(ds, L4Re::This_task);
return 0;
```

12.6 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)
 \star 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 <14/re/env>
#include <14/re/namespace>
#include <14/re/util/cap_alloc>
#include <14/re/util/object_registry>
#include <14/re/dataspace>
#include <14/cxx/ipc_server>
#include <14/sys/typeinfo_svr>
#include <cstring>
#include <cstdio
#include <unistd.h>
#include "interface.h"
class My_server_obj : public L4::Server_object
  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(14_umword_t obj, L4::Ipc::Iostream &ios);
int My_server_obj::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
  \ensuremath{//} we don't care about the original object reference, however
  // we could read out the access rights from the lowest 2 bits
  (void) obj;
  14_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::Server_object
private:
 char * shm;
public:
 explicit Shm_observer(char *shm)
  : _shm(shm)
 int dispatch(14_umword_t obj, L4::Ipc::Iostream &ios);
int Shm_observer::dispatch(14_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;
 14_msgtag_t t;
ios >> t; // extract the tag
  switch (t.label())
    case L4::Irq::Protocol:
     // Got an IRQ so just print the new contents of the
      // shared memory.
printf("Content: %s\n", _shm);
      return 0:
    default:
     // every other protocol is not supported.
      return -L4_EBADPROTO;
static L4Re::Util::Registry_server<> server;
 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,
                                         * ds);
  if (err < 0)</pre>
    {
      printf("Error attaching data space: sn, 14sys_{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);
\ensuremath{//} 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;
```

12.7 examples/libs/l4re/c++/shared_ds/shared_ds.lua

Sharing memory between applications, configuration file.

12.8 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 <14/re/c/mem_alloc.h>
#include <14/re/c/rm.h>
#include <14/re/c/util/cap_alloc.h>
#include <14/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;
  14re_ds_t ds;
  /\star Allocate a free capability index for our data space \star/
  ds = 14re_util_cap_alloc();
if (14_is_invalid_cap(ds))
    return -L4_ENOMEM;
  size_in_bytes = 14_trunc_page(size_in_bytes);
  /\star Allocate memory via a dataspace \star/
  if ((r = 14re_ma_alloc(size_in_bytes, ds, flags)))
    return r;
  /\star Make the dataspace visible in our address space \star/
  *virt_addr = 0;
  if ((r = 14re_rm_attach(virt_addr, size_in_bytes,
                              L4RE_RM_SEARCH_ADDR, ds, 0, flags & L4RE_MA_SUPER_PAGES
                                  ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
    return r;
  /\star Done, virtual address is in virt_addr \star/
  return 0;
static int free_mem(void *virt_addr)
  int r;
  14re_ds_t ds;
 /* Detach memory from our address space */
if ((r = l4re_rm_detach_ds(virt_addr, &ds)))
    return r;
  /\star Free memory at our memory allocator \star/
  if ((r = 14re_ma_free(ds)))
    return r;
  14re_util_cap_free(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;
```

12.9 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)
 ^{\circ} This file is part of TUD:0S and distributed under the terms of the \star GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
#include <14/sys/err.h>
#include <14/sys/types.h>
#include <14/re/env>
#include <14/re/util/cap_alloc>
#include <14/cxx/ipc_stream>
#include <stdio.h>
#include "shared.h"
func_smap_call(L4::Cap<void> const &server)
  L4::Ipc::Iostream s(14_utcb());
  14\_addr\_t addr = 0;
  int err;
  if ((err = L4Re::Env::env()->rm()->reserve_area(&addr,
                                                    L4Re::Rm::Search_addr)))
      printf("The reservation of one page within our virtual memory failed with d^n, err);
      return 1:
  s << 14_umword_t (Opcode::Do_map)
   << (14_addr_t)addr;
  s << L4::Ipc::Rcv_fpage::mem((14_addr_t)addr, L4_PAGESHIFT, 0);
  int r = 14_error(s.call(server.cap(), Protocol::Map_example));
if (r)
    return r; // failure
 printf("String sent by server: s\n", (char *)addr);
  return 0; // ok
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;
```

12.10 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)
 \star 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 <14/re/env>
#include <14/re/util/cap_alloc>
#include <14/re/util/object_registry>
#include <14/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
public:
  int dispatch(14_umword_t obj, L4::Ipc::Iostream &ios);
Smap_server::dispatch(14_umword_t, L4::Ipc::Iostream &ios)
  14_msgtag_t t;
  // We're only talking the Map_example protocol
  if (t.label() != Protocol::Map_example)
    return -L4_EBADPROTO;
  L4::Opcode opcode;
  ios >> opcode;
  switch (opcode)
    case Opcode::Do_map:
      14 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((14_addr_t)page_to_map,</pre>
      L4_PAGESHIFT,
                                L4_FPAGE_RO, snd_base);
      return L4_EOK;
    default:
      return -L4_ENOSYS;
}
int
main()
  static Smap_server smap;
  // Register server
    (!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:
```

12.11 examples/libs/l4re/streammap/streammap.cfg

Sample configuration file for the client/server map example.

12.12 examples/libs/libirq/async_isr.c

libirq usage example using asychronous ISR handler functionality.

```
* (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
       economic rights: Technische Universität Dresden (Germany)
 \star This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 \star Please see the COPYING-GPL-2 file for details.
\star This example shall show how to use the libirg.
#include <14/irq/irq.h>
#include <14/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 = 14irq_request(IRQ_NO, isr_handler, 0, 0xff, 0)))
      printf("Requesting IRQ %d failed\n", IRQ_NO);
  printf("Attached to key IRQ %d\nPress keys now, will terminate in %d seconds\n",
         IRQ_NO, seconds);
  14_sleep(seconds * 1000);
  if (l4irq_release(irqdesc))
      printf("Failed to release IRQ\n");
      return 1;
  printf("Bye\n");
  return 0;
```

12.13 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)
^{\circ} This file is part of TUD:0S and distributed under the terms of the \star GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
#include <14/irq/irq.h>
#include <14/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 = 14irq_attach(IRQ_NO)))
   return NULL;
  while (1)
      if (l4irq_wait(irq))
      isr_handler();
 return NULL;
int main (void)
 pthread_t thread;
 if (pthread_create(&thread, NULL, isr_thread, NULL))
    return 1;
 14_sleep_forever();
 return 0:
```

12.14 examples/libs/shmc/prodcons.c

Simple shared memory example.

```
#include <14/sys/thread.h>
// a small helper
static const char some_data[] = "Hi consumer!";
static void *thread_producer(void *d)
  (void)d;
 14shmc_chunk_t p_one;
14shmc_signal_t s_one, s_done;
  14shmc_area_t shmarea;
 // attach this thread to the shm object
CHK(14shmc_attach("testshm", &shmarea));
  // add a chunk
 CHK(14shmc_add_chunk(&shmarea, "one", 1024, &p_one));
  // add a signal
 {\tt CHK(14shmc\_add\_signal(\&shmarea, "prod", \&s\_one));}
 CHK(14shmc_attach_signal_to(&shmarea, "done",
                               pthread_get14cap(pthread_self()), 10000, &s_done));
  // connect chunk and signal
 CHK(14shmc_connect_chunk_signal(&p_one, &s_one));
 printf("PRODUCER: ready\n");
      while (14shmc_chunk_try_to_take(&p_one))
printf("Uh, should not happen!\n"); //14_thread_yield();
      memcpy(14shmc_chunk_ptr(&p_one), some_data, sizeof(some_data));
      CHK(14shmc_chunk_ready_sig(&p_one, sizeof(some_data)));
      printf("PRODUCER: Sent data\n");
     CHK(14shmc_wait_signal(&s_done));
 14_sleep_forever();
 return NULL;
static void *thread_consume(void *d)
  (void)d:
  14shmc_area_t shmarea;
 14shmc_chunk_t p_one;
14shmc_signal_t s_one, s_done;
 // attach to shared memory area
CHK(14shmc_attach("testshm", &shmarea));
  // get chunk 'one'
 CHK(14shmc_get_chunk(&shmarea, "one", &p_one));
  // add a signal
 CHK(14shmc_add_signal(&shmarea, "done", &s_done));
  // attach signal to this thread
 CHK(14shmc_attach_signal_to(&shmarea, "prod",
                                pthread_get14cap(pthread_self()), 10000, &s_one));
  // connect chunk and signal
 CHK(14shmc_connect_chunk_signal(&p_one, &s_one));
  while (1)
      CHK(14shmc_wait_chunk(&p_one));
      printf("CONSUMER: Received from chunk one: s\n",
      (char *)14shmc_chunk_ptr(&p_one));
memset(14shmc_chunk_ptr(&p_one), 0, 14shmc_chunk_size(&p_one));
      CHK(14shmc_chunk_consumed(&p_one));
      CHK(14shmc_trigger(&s_done));
  return NULL;
```

```
int main(void)
{
  pthread_t one, two;

  // create new shared memory area, 8K in size
  if (14shmc_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
  14_sleep_forever();
  return 0;
}
```

12.15 examples/sys/aliens/main.c

This example shows how system call tracing can be done.

```
\star (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.
//#define MEASURE
#include <14/sys/ipc.h>
#include <14/sys/thread.h>
#include <14/sys/factory.h>
#include <14/sys/utcb.h>
#include <14/sys/kdebug.h>
#include <14/util/util.h>
#include <14/util/rdtsc.h>
#include <14/re/env.h>
#include <14/re/c/util/cap_alloc.h>
#include <14/sys/debugger.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
static char alien_thread_stack[8 << 10];</pre>
static 14_cap_idx_t alien;
static void alien_thread(void)
  volatile 14_msgtag_t x;
  while (1) {
    x = 14\_ipc\_call(0x1234 << L4\_CAP\_SHIFT, 14\_utcb(),
     14_msgtag(0, 0, 0, 0), L4_IPC_NEVER);
#ifdef MEASURE
    14_sleep(0);
#else
    14_sleep(1000);
    outstring("An int3 -- you should see this\n");
    outnstring("345", 3);
#endif
  }
}
int main(void)
  14_msgtag_t tag;
#ifdef MEASURE
 14_cpu_time_t s, e;
#endif
```

```
14\_utcb\_t *u = 14\_utcb();
  14_exc_regs_t exc;
  14_umword_t mr0, mr1;
 printf("Alien feature testing\n");
  14_debugger_set_object_name(l4re_env()->main_thread, "alientest");
  /* Start alien thread */
  if (14_is_invalid_cap(alien = 14re_util_cap_alloc()))
    return 1:
  14_touch_rw(alien_thread_stack, sizeof(alien_thread_stack));
  tag = 14_factory_create_thread(14re_env()->factory, alien);
  if (14_error(tag))
    return 1:
  14_debugger_set_object_name(alien, "alienth");
  14_thread_control_start();
  14_thread_control_pager(14re_env()->main_thread);
  14_thread_control_exc_handler(14re_env()->main_thread);
  14_thread_control_bind((14_utcb_t *)14re_env()->first_free_utcb,
     L4RE_THIS_TASK_CAP);
  14_thread_control_alien(1);
  tag = 14_thread_control_commit(alien);
  if (14_error(tag))
   return 2;
 tag = 14_thread_ex_regs(alien,
                          (14_umword_t)alien_thread,
                          (14_umword_t)alien_thread_stack + sizeof(alien_thread_stack),
                          0);
 if (14_error(tag))
   return 3:
  14_sched_param_t sp = 14_sched_param(1, 0);
  tag = 14_scheduler_run_thread(14re_env()->scheduler, alien, &sp);
  if (14_error(tag))
   return 4;
#ifdef MEASURE
 14_calibrate_tsc(14re_kip());
#endif
  /* Pager/Exception loop */
 printf("14_ipc_receive failed");
 memcpy(&exc, 14_utcb_exc(), sizeof(exc));
mr0 = 14_utcb_mr()->mr[0];
mr1 = 14_utcb_mr()->mr[1];
  for (;;)
#ifdef MEASURE
     s = 14_rdtsc();
#endif
      if (14_msgtag_is_exception(tag))
#ifndef MEASURE
         printf("PC=%081x SP=%081x Err=%081x Trap=%1x, %s syscall, SC-Nr: %1x\n",
                14_utcb_exc_pc(&exc), exc.sp, exc.err,
exc.trapno, (exc.err & 4) ? " after" : "before",
exc.err >> 3);
#endif
         printf("Umm, non-handled request (like PF): %lx %lx\n", mr0, mr1);
      memcpy(14_utcb_exc(), &exc, sizeof(exc));
      /* Reply and wait */
      if (14_msgtag_has_error(tag = 14_ipc_call(alien, u, tag,
      L4_IPC_NEVER)))
          printf("14_ipc_call failed\n");
          return 1;
```

```
memcpy(&exc, 14_utcb_exc(), sizeof(exc));
    mr0 = 14_utcb_mr()->mr[0];
    mr1 = 14_utcb_mr()->mr[1];
#ifdef MEASURE
    e = 14_rdtsc();
    printf("time %lld\n", 14_tsc_to_ns(e - s));
#endif
    }
    return 0;
}
```

12.16 examples/sys/ipc/ipc.cfg

Sample configuration file for the IPC example.

```
# vim:se ft=lua:
require("L4");
L4.default_loader:start({}, "rom/ex_ipc1");
```

12.17 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)
 \star This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 \star Please see the COPYING-GPL-2 file for details.
#include <14/sys/ipc.h>
#include <pthread-14.h>
#include <unistd.h>
#include <stdio.h>
static pthread t t2;
/\star Threadl is the initiator thread, i.e. it initiates the IPC calls. In
 \star other words, it takes the client role. It uses L4 IPC mechanisms to send
 \star an integer value to thread2 and received a calculation result back. \star/
static void *thread1_fn(void *arg)
  14_msgtag_t tag;
  int ipc_error;
  unsigned long value = 1;
  (void) arg;
  while (1)
      printf("Sending: %ld\n", value);
      /\star Store the value which we want to have squared in the first message
        \star register of our UTCB. \star/
      14_utcb_mr()->mr[0] = value;
      /\star To an L4 IPC call, i.e. send a message to thread2 and wait for a \star reply from thread2. The '1' in the msgtag denotes that we want to
       \star transfer one word of our message registers (i.e. MR0). No timeout. \star/
      tag = 14_ipc_call(pthread_get14cap(t2), 14_utcb(),
                           14_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 = 14_ipc_error(tag, 14_utcb());
          (ipc_error)
        fprintf(stderr, "thread1: IPC error: %x\n", ipc_error);
      else
        printf("Received: %ld\n", 14_utcb_mr()->mr[0]);
       /* Wait some time and increment our value. */
```

```
sleep(1);
      value++;
 return NULL;
/\star Thread2 is in the server role, i.e. it waits for requests from others and
 \star sends back the calculation results. \star/
static void *thread2_fn(void *arg)
 14_msgtag_t tag;
14_umword_t label;
  int ipc_error;
  (void) arg;
  /\star Wait for requests from any thread. No timeout, i.e. wait forever. \star/
  tag = 14_ipc_wait(14_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 = 14_ipc_error(tag, 14_utcb());
      if (ipc_error)
          fprintf(stderr, "thread2: IPC error: %x\n", ipc_error);
          tag = 14_ipc_wait(14_utcb(), &label, L4_IPC_NEVER);
          continue;
        }
      /\star So, the IPC was ok, now take the value out of message register 0
       \star of the UTCB and store the square of it back to it. \star/
      14_utcb_mr()->mr[0] = 14_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
       \star the message registers (i.e. MR0) \star/
      tag = 14_ipc_reply_and_wait(14_utcb(),
      14_msgtag(0, 1, 0, 0),
                                     &label, L4_IPC_NEVER);
 return NULL;
int main(void)
 //\ \mbox{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");
  // Just run threadl in the main thread
 thread1_fn(NULL);
 return 0:
```

12.18 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 <14/re/c/util/cap_alloc.h>
```

```
#include <14/re/c/namespace.h>
#include <14/sys/utcb.h>
#include <14/sys/irq.h>
#include <14/sys/factory.h>
#include <14/sys/icu.h>
#include <stdio.h>
int main(void)
  int irqno = 1;
  14_cap_idx_t irqcap, icucap;
14_msgtag_t tag;
  icucap = 14re_env_get_cap("icu");
  /\star Get a free capability slot for the ICU capability \star/
  if (14_is_invalid_cap(icucap))
      printf("Did not find an ICU\n");
  /* Get another free capaiblity slot for the corresponding IRQ object*/
if (14_is_invalid_cap(irqcap = 14re_util_cap_alloc()))
    return 1;
  /* Create IRQ object */
  if (14_error(tag = 14_factory_create_irq(14re_global_env->
      factory, irqcap)))
      printf("Could not create IRQ object: %lx\n", 14_error(tag));
  * Bind the recently allocated IRQ object to the IRQ number irqno
   * as provided by the ICU.
  if (14_error(14_icu_bind(icucap, irqno, irqcap)))
      printf("Binding IRQ%d to the ICU failed\n", irqno);
  /* Attach ourselves to the IRQ */
  tag = 14_irq_attach(irqcap, 0xDEAD, 14re_env()->main_thread);
  if ((err = 14_error(tag)))
      printf("Error attaching 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 = 14_irq_receive(irqcap, L4_IPC_NEVER);
if ((err = 14_ipc_error(tag, 14_utcb())))
       printf("Error on IRQ receive: %d\n", err);
      else
           /\star Process the interrupt -- may do a 'break' \star/
          printf("Got IRQ with label 0x%lX\n", label);
    }
  /* We're done, detach from the interrupt. */
  tag = 14_irq_detach(irqcap);
if ((err = 14_error(tag)))
    printf("Error detach from IRQ: %d\n", err);
 return 0;
```

12.19 examples/sys/migrate/thread_migrate.cc

Thread migration example.

```
* (c) 2008-2009 Author(s)
       economic rights: Technische Universität Dresden (Germany)
 \star This file is part of TUD:0S and distributed under the terms of the \star GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
#include <14/sys/scheduler>
#include <14/re/env>
#include <14/re/util/cap_alloc>
#include <pthread-14.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
enum { NR THREADS = 12 };
static L4::Cap<L4::Thread> threads[NR_THREADS];
static 14_umword_t
                         cpu_map, cpu_nrs;
/\star 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)
  14_sched_cpu_set_t cs = 14_sched_cpu_set(0, 0);
  if (14_error(L4Re::Env::env()->scheduler()->info(&cpu_nrs, &cs)) < 0)</pre>
    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;</pre>
/\star Create a couple of threads and store their capabilities in an array \star/
static int create_threads(void)
  unsigned i;
  for (i = 0; i < NR_THREADS; ++i)</pre>
      pthread_t t;
      if (pthread_create(&t, NULL, thread_fn, NULL))
        return 1;
      threads[i] = L4::Cap<L4::Thread>(pthread_get14cap(t));
  printf("Created %d threads.\n", NR_THREADS);
  return 0;
/\star Helper function to get the next CPU \star/
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;
}
/\star Function that shuffles the threads on the available CPUs \star/
```

```
static void shuffle (void)
 unsigned start = 0;
 while (1)
      unsigned t:
      unsigned c = start;
      for (t = 0; t < NR_THREADS; ++t)</pre>
          14_sched_param_t sp = 14_sched_param(20);
          c = get_next_cpu(c);
          sp.affinity = 14_sched_cpu_set(c, 0);
if (14_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;
```

12.20 examples/sys/migrate/thread_migrate.cfg

Sample configuration file for the thread migration example.

12.21 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>,
               economic rights: Technische Universität Dresden (Germany)
 \star This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 \star Please see the COPYING-GPL-2 file for details.
* Single stepping example for the x86-32 architecture.
#include <14/sys/ipc.h>
#include <14/sys/factory.h>
#include <14/sys/thread.h>
#include <14/sys/utcb.h>
#include <14/sys/kdebug.h>
#include <14/util/util.h>
#include <14/re/env.h>
#include <14/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)
 14_msgtag_t tag;
  int ipc_stat = 0;
  14_cap_idx_t th = 14re_util_cap_alloc();
  14_exc_regs_t exc;
 14_umword_t mr0, mr1;
14_utcb_t *u = 14_utcb();
 printf("Singlestep testing\n");
 if (14_is_invalid_cap(th))
   return 1;
 14_touch_rw(thread_stack, sizeof(thread_stack));
14_touch_ro(thread_func, 1);
  tag = 14_factory_create_thread(14re_env()->factory, th);
  if (14_error(tag))
   return 1:
  14_thread_control_start();
  14_thread_control_pager(14re_env()->main_thread);
  14_thread_control_exc_handler(14re_env()->main_thread);
  14_thread_control_bind((14_utcb_t *)14re_env()->first_free_utcb,
                           L4RE_THIS_TASK_CAP);
  14_thread_control_alien(1);
  tag = 14_thread_control_commit(th);
  if (l4_error(tag))
    return 2;
 0);
 if (14_error(tag))
   return 3;
 14_sched_param_t sp = 14_sched_param(1, 0);
tag = 14_scheduler_run_thread(14re_env()->scheduler, th, &sp);
  if (14_error(tag))
   return 4;
  /* Pager/Exception loop */
  printf("14_ipc_receive failed");
      return 5;
  memcpy(&exc, 14_utcb_exc(), sizeof(exc));
 mr0 = 14_utcb_mr()->mr[0];
mr1 = 14_utcb_mr()->mr[1];
```

```
for (;;)
    if (14_msgtag_is_exception(tag))
        printf("PC = %081x Trap = %081x Err = %081x, SP = %081x SC-Nr: %1x\n",
               14_utcb_exc_pc(&exc), exc.trapno, exc.err,
        exc.sp, exc.err >> 3);
if (exc.err >> 3)
            if (!(exc.err & 4))
                tag = 14_msgtag(L4_PROTO_ALLOW_SYSCALL,
                                 L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
                if (ipc_stat)
                  enter_kdebug("Should not be 1");
            else
                tag = 14_msgtag(L4_PROTO_NONE,
                                L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
                if (!ipc_stat)
                  enter_kdebug("Should not be 0");
            ipc_stat = !ipc_stat;
        14_sleep(100);
    else
      printf("Umm, non-handled request: %ld, %08lx %08lx\n",
             14 msgtag label(tag), mr0, mr1);
    memcpy(14_utcb_exc(), &exc, sizeof(exc));
    /\star Reply and wait \star/
    if (14_msgtag_has_error(tag = 14_ipc_call(th, u, tag,
    L4_IPC_NEVER)))
        printf("l4_ipc_call failed\n");
    memcpy(&exc, 14_utcb_exc(), sizeof(exc));
    mr0 = 14 utcb mr() -> mr[0];
   mr1 = 14_utcb_mr()->mr[1];
return 0;
```

12.22 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.
\star Start a thread with an exception reply. This example does only work on \star the x86-32 architecture.
#include <14/sys/thread.h>
#include <14/sys/factory.h>
#include <14/sys/ipc.h>
#include <14/sys/utcb.h>
#include <14/util/util.h>
#include <14/re/env.h>
#include <14/re/c/util/cap_alloc.h>
#include <stdlib.h>
#include <stdio.h>
/\star Stack for the thread to be created. 8kB are enough. \star/
static char thread_stack[8 << 10];</pre>
```

```
/* The thread to be created. For illustration it will print out its
* register set.
*/
static void L4_STICKY(thread_func(14_umword_t *d))
  while (1)
      printf("hey, I'm a thread\n");
      printf("got register values: %ld %ld %ld %ld %ld %ld %ld %ld \n", d[7], d[6], d[5], d[4], d[2], d[1], d[0]);
      14_sleep(800);
/\star 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(
                     \n\t
".global thread
               \n\t
" push %esp
               \n\t"
" call thread_func \n\t"
extern void thread(void);
/* Our main function */
int main(void)
  /* Get a capability slot for our new thread. */
14_cap_idx_t t1 = 14re_util_cap_alloc();
  14_utcb_t *u = 14_utcb();
14_exc_regs_t *e = 14_utcb_exc_u(u);
  14_msgtag_t tag;
  int err;
  extern char _start[], _end[], _sdata[];
  if (14_is_invalid_cap(t1))
    return 1:
  /\star Prevent pagefaults of our new thread because we do not want to
   * implement a pager as well. */
  14_touch_ro(_start, _sdata - _start + 1);
14_touch_rw(_sdata, _end - _sdata);
  /\star Create the thread using our default factory \star/
  tag = 14_factory_create_thread(14re_env()->factory, t1);
  if (14_error(tag))
    return 1:
  /* Setup the thread by setting the pager and task. */
  14_thread_control_start();
  14_thread_control_pager(14re_env()->main_thread);
  14_thread_control_exc_handler(14re_env()->main_thread);
  14_thread_control_bind((14_utcb_t *)14re_env()->first_free_utcb,
                            L4RE_THIS_TASK_CAP);
  tag = 14_thread_control_commit(t1);
  if (14_error(tag))
    return 2;
  /\star Start the thread by finally setting instruction and stack pointer \star/
  tag = 14_thread_ex_regs(t1,
                            (14_umword_t)thread,
                             (14_umword_t)thread_stack + sizeof(thread_stack),
                            L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);
  if (14_error(tag))
    return 3;
  14_sched_param_t sp = 14_sched_param(1, 0);
  tag = 14_scheduler_run_thread(14re_env()->scheduler, t1, &sp);
  if (14_error(tag))
    return 4;
  /* Receive initial exception from just started thread */
tag = 14_ipc_receive(t1, u, L4_IPC_NEVER);
    ((err = 14_ipc_error(tag, u)))
      printf("Umm, ipc error: x\n", err);
      return 1:
  /* We expect an exception IPC */
  if (!14_msgtag_is_exception(tag))
      printf("PF?: %lx %lx (not prepared to handle this) %ld\n",
       14_utcb_mr_u(u)->mr[0], 14_utcb_mr_u(u)->mr[1], 14_msgtag_label(tag));
      return 1:
```

```
}
/* Fill out the complete register set of the new thread */
e->ip = (14_umword_t)thread;
e->sp = (14_umword_t)(thread_stack + sizeof(thread_stack));
e->eax = 1;
e \rightarrow e dx = 3;
e->esi = 6;
e->edi = 7;
e->ebp = 5;
/* Send a complete exception */
tag = 14_msgtag(0, L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
/\star Send reply and start the thread with the defined CPU register set \star/
tag = 14_ipc_send(t1, u, tag, L4_IPC_NEVER);
if ((err = 14_ipc_error(tag, u)))
  printf("Error sending IPC: %x\n", err);
/* Idle around */
while (1)
  14_sleep(10000);
return 0;
```

12.23 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>,
                   Biörn Döbel <doebel@os.inf.tu-dresden.de>
        economic rights: Technische Universität Dresden (Germany)
 \star This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 \star Please see the COPYING-GPL-2 file for details.
#include <14/sys/ipc.h>
#include <14/sys/thread.h>
#include <14/sys/factory.h>
#include <14/sys/utcb.h>
#include <14/re/env.h>
#include <14/re/c/util/cap_alloc.h>
#include <stdio.h>
#include <string.h>
static unsigned char stack2[8 << 10];</pre>
static 14_cap_idx_t thread1_cap, thread2_cap;
static void thread1 (void)
  14_msg_regs_t *mr = 14_utcb_mr();
  14_msgtag_t tag;
  int i, j;
  printf("Thread1 up (%p)\n", 14_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 = 14_msgtag(0, L4_UTCB_GENERIC_DATA_SIZE, 0, 0);
if (14_msgtag_has_error(14_ipc_send(thread2_cap,
       14_utcb(), tag, L4_IPC_NEVER)))
  printf("IPC-send error\n");
static void thread2 (void)
  14_msgtag_t tag;
  14_msg_regs_t mr;
  unsigned i;
  printf("Thread2 up (%p)\n", 14_utcb());
```

```
while (1)
       if (14_msgtag_has_error(tag = 14_ipc_receive(thread1_cap,
       14_utcb(), L4_IPC_NEVER()))
printf("IPC receive error\n");
      memcpy(&mr, 14_utcb_mr(), sizeof(mr));
printf("Thread2 receive (%d): ", 14_msgtag_words(tag));
       for (i = 0; i < 14_msgtag_words(tag); i++)
  printf("%c", (char)mr.mr[i]);</pre>
      printf("\n");
}
int main (void)
  14_msgtag_t tag;
  thread1_cap = 14re_env()->main_thread;
thread2_cap = 14re_util_cap_alloc();
  if (14_is_invalid_cap(thread2_cap))
  tag = 14_factory_create_thread(14re_env()->factory, thread2_cap);
  if (14_error(tag))
   return 1;
  14_thread_control_start();
  14_thread_control_pager(14re_env()->rm);
  14_thread_control_exc_handler(14re_env()->rm);
14_thread_control_bind((14_utcb_t *)14re_env()->first_free_utcb,
                              L4RE_THIS_TASK_CAP);
  tag = 14_thread_control_commit(thread2_cap);
  if (14_error(tag))
    return 2;
  tag = 14_thread_ex_regs(thread2_cap,
                               (14_umword_t)thread2,
                               (14_umword_t) (stack2 + sizeof(stack2)), 0);
  if (l4_error(tag))
    return 3;
  14_sched_param_t sp = 14_sched_param(1, 0);
  tag = 14_scheduler_run_thread(14re_env()->scheduler, thread2_cap, &sp);
  if (14_error(tag))
  thread1();
  return 0:
```

12.24 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.
 \star Please see the COPYING-GPL-2 file for details.
#include <14/sys/ipc.h>
#include <14/sys/vhw.h>
#include <14/util/util.h>
#include <14/util/kip.h>
#include <14/re/env.h>
#include <stdlib.h>
#include <stdio.h>
static void print_entry(struct 14_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);
```

12.25 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);
 }
}
```

12.26 tmpfs/lib/src/fs.cc

Example file system for L4Re::Vfs.

```
#include <sys/ioctl.h>
#include <dirent.h>
#include <cstdio>
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:
                 _ref_cnt;
               *_path;
  char
  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 *1, const char *r) const
  { return strcmp(1, r) < 0; }
 bool operator () (const cxx::String 1, const cxx::String r) const
    int v = strncmp(l.start(), r.start(), cxx::min(l.len(), r.len()));
    return v < 0 || (v == 0 && 1.len() < r.len());</pre>
};
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
                                     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();
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();
  ssize_t preadv(const struct iovec *v, int iovent, 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 iovent, off64_t p) throw()
  if (iovcnt < 0)</pre>
   return -EINVAL;
  ssize_t sum = 0;
  for (int i = 0; i < iovent; ++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 -EIO; // most likely ENOSPC, but can't report that
off64_t Tmpfs_file::size() const throw()
{ return _file->data().size(); }
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;
```

```
}
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;
Tmpfs_dir::get_entry(const char *name, int flags, mode_t mode,
                       Ref_ptr<File> *file) throw()
  Ref_ptr<Node> path;
  if (!*name)
    {
     *file = 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)</pre>
    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);
         return -ENOMEM;
      path = node;
  if (path->is_dir())
    *file = new Tmpfs_dir(cxx::ref_ptr_static_cast<Pers_dir>(path));
    *file = 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 1 = strlen(_getdents_iter->path()) + 1;
       if (1 > sizeof(d->d_name))
        1 = sizeof(d->d_name);
      unsigned n = offsetof (struct dirent64, d_name) + 1; n = (n + sizeof(long) - 1) & \sim(sizeof(long) - 1);
```

```
if (n > sz)
       break;
      d->d_ino = 1;
d->d_off = 0;
      memcpy(d->d_name, _getdents_iter->path(), 1);
      d->d_reclen = n;
      d->d_type = DT_REG;
     ret += n;

sz -= n;

d = (struct dirent64 *)((unsigned long)d + n);
 return ret;
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;
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)</pre>
       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;
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;
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

\sim Auto_ptr	L4Re::Rm, 861, 862
cxx::Auto_ptr, 458	L4Re::Util::Event_buffer_t, 637
\sim Be_file_system	Attach_flags
L4Re::Vfs::Be_file_system, 498	L4Re::Rm, 858
pad0	Attach_flags
cxx::Avl_map, 465	L4Re::Rm, 858
_c	Attr
L4::Cap_base, 553	L4::Thread::Attr, 451
· <u> </u>	Auto_ptr
a	cxx::Auto_ptr, 458
L4Re::Video::Pixel_info, 845, 847	auto refresh
ARM Virtual Registers (UTCB), 311	L4Re::Video::Goos::Info, 685
add	Auxiliary data, 108
L4::Thread::Modify_senders, 818	avail
L4vcpu::State, 896	cxx::List_alloc, 794
add_ku_mem	AvI set
L4::Task, 909	cxx::Avl set, 469, 470
alloc	= ', '
cxx::Base_slab_static, 489	ax
cxx::List alloc, 794	l4_vcpu_regs_t, 754
cxx::Slab, 884	b
•	
cxx::Slab_static, 888	L4Re::Video::Pixel_info, 845, 847
L4Re::Cap_alloc, 542	Base API, 123
L4Re::Mem_alloc, 805	Basic Macros, 129
alloc_fd	L4_EXPORT, 130
L4Re::Vfs::Fs, 659	L4_HIDDEN, 131
allocate	L4_NOTHROW, 130
L4Re::Dataspace, 570	Be_file_system
L4Re::Util::Dataspace_svr, 578	L4Re::Vfs::Be_file_system, 498
amd64 Virtual Registers (UTCB), 313	begin
asm_enter_kdebug	cxx::Avl_set, 472, 473
Kernel Debugger, 181, 183	cxx::Bits::Bst, 523, 524
Atomic Instructions, 327	bind
l4util_add8, 331	L4::lcu, 676
l4util_add8_res, 331	L4::Thread::Attr, 453
l4util_atomic_add, 332	bind_thread
I4util atomic inc, 332	L4::lpc_gate, 702
l4util_cmpxchg, 330	bit
I4util_cmpxchg16, 329	cxx::Bitmap_base, 513
I4util_cmpxchg32, 329	Bit Manipulation, 334
l4util cmpxchg64, 328	l4util bsf, 336
l4util_cmpxchg8, 329	14util bsr, 336
I4util inc8, 331	14util_btc, 336
I4util_inc8_res, 331	14util_btr, 335
l4util_xchg, 331	14util_bts, 335
l4util_xchg16, 330	I4util_clear_bit, 335
l4util_xchg32, 330	l4util_complement_bit, 335
l4util_xchg8, 330	
_ •	
attach	I4util_find_first_zero_bit, 337
L4::Irq, 705	l4util_next_power2, 337

l4util_set_bit, 334	L4::Kobject, 723
l4util_test_bit, 335	cap_alloc
Bitmap	L4Re Capability API, 116
cxx::Bitmap, 510	Cap_base
Bitmap graphics and fonts, 374	L4::Cap base, 548
Bits	cap_cast
cxx::Bitfield, 501	Capabilities, 287
bits_per_pixel	cap_dynamic_cast
L4Re::Video::Pixel_info, 846	Capabilities, 288
Bits type	cap equal
cxx::Bitfield, 500	. — .
bp	L4::Task, 908
I4_vcpu_regs_t, 753	cap_has_child
buf	L4::Task, 908
	cap_reinterpret_cast
L4::lpc::Buf_cp_out, 535	Capabilities, 288
L4Re::Util::Event_buffer_t, 636	Cap_type
Buf_cp_in	L4::Cap_base, 547
L4::lpc::Buf_cp_in, 532	cap_valid
buf_cp_in	L4::Task, 907
IPC Messaging Framework, 54	Capabilities, 284
Buf_cp_out	cap_cast, 287
L4::lpc::Buf_cp_out, 535	cap_dynamic_cast, 288
buf_cp_out	cap_reinterpret_cast, 288
IPC Messaging Framework, 53	L4 BASE FACTORY CAP, 287
Buf_in	L4_BASE_ICU_CAP, 287
L4::lpc::Buf_in, 537	L4_BASE_LOG_CAP, 287
buf_in	L4 BASE PAGER CAP, 287
IPC Messaging Framework, 54	L4_BASE_SCHEDULER_CAP, 287
buffer	
L4Re::Util::Event_t, 639	L4_BASE_TASK_CAP, 287
Buffer Registers (BRs)	L4_BASE_THREAD_CAP, 287
L4_BDR_IO_SHIFT, 296	L4_CAP_MASK, 286
L4_BDR_MEM_SHIFT, 296	L4_CAP_SHIFT, 286
L4_BDR_OBJ_SHIFT, 296	L4_CAP_SIZE, 286
Buffer Registers (BRs), 296	L4_INVALID_CAP, 286
I4_buffer_desc_consts_t, 296	L4_DISABLE_COPY, 285
	L4_KOBJECT, 286
bx	I4_cap_consts_t, 286
I4_vcpu_regs_t, 754	l4_cap_idx_t, 286
bytes_per_pixel	I4_capability_equal, 290
L4Re::Video::Pixel_info, 846, 847	l4_default_caps_t, 286
C++ Exceptions, 41	I4_is_invalid_cap, 289
CPU related functions, 315	I4_is_valid_cap, 289
	Capability allocator, 87
I4util_cpu_capabilities, 315	l4re_util_cap_last, 87
	cast
I4util_cpu_has_cpuid, 315	L4vcpu::Vcpu, 947
Cache Consistency, 173	•
l4_cache_clean_data, 173	chain
I4_cache_coherent, 174	L4::lrq, 706
l4_cache_dma_coherent, 174	chars
I4_cache_flush_data, 173	cxx::Bitmap_base, 513
I4_cache_inv_data, 174	Chunks, 418
call	l4shmc_add_chunk, 418
L4::lpc::lostream, 697	l4shmc_chunk_capacity, 421
Cap	l4shmc_chunk_ptr, 421
L4::Cap, 539	l4shmc_chunk_signal, 421
cap	l4shmc_get_chunk, 419
L4::Cap_base, 548	l4shmc_get_chunk_to, 419
L4::Invalid_capability, 692	I4shmc_iterate_chunk, 419
— · · · · · · · · · · · · · · · · · · ·	, -

L4Re::Util::Dataspace, 569 L4Re::Util::Dataspace, 577 clear_ail cx::Bilmap, 510 clear_bit cx::Bilmap, 510 clear_bit cx::Bilmap, 513 Client/Sarver IPC Framework, 45 cni_clomap_tib_flush L4_tracebuffer_status_1, 748 Color component L4::Com_error, 560 Comfortable Command Line Parsing, 360 parse_cmdine, 360 Console API, 96 L4::Com_error, 560 Console API, 96 Console AP	clear	create_view
L4He::Util::Dataspace_svr, 578 L4vcpu::State, 897 clear all cxx::Bitmap, 510 clear bil cxx::Bitmap, 510 cx::Bitmap, 510 cx::Bi		
L4xcpu:State, 897	·	
Cx	· —	_
Circumstable Circ	•	• •
Clear bit		
Cxx::Birmap base, 513	·	_ · _ · _
Client/Server IPC Framework, 45		•
cnt_iobmap_tb_flush Il_tracebuffer_status_t, 748 Objects_per_slab, 485 Objects_per_slab, 485 Slab_size, 485 Comcomponent L4Re:Video:Color_component, 555 Com_error L4:Com_error, 560 Comfortable Command Line Parsing, 360 parse_comdine, 360 Console API, 96 Consumer, 424, 433 Idshmc_chunk_consumed, 425 Idshmc_chunk_consumed, 425 Idshmc_chunk_size, 427 Idshmc_enable_chunk, 424 Idshmc_enable_signal, 433 Idshmc_wait_any_to, 434 Idshmc_wait_any_to, 434 Idshmc_wait_any_to, 434 Idshmc_wait_any_to, 434 Idshmc_wait_drunk_to, 425 Idshmc_wait_drunk_to, 425 Idshmc_wait_drunk_to, 425 Idshmc_wait_drunk_to, 425 Idshmc_wait_signal_to, 434 Idshmc_wait_signal_to, 434 Idshmc_wait_signal_to, 434 Idshmc_wait_signal_to, 434 Idshmc_wait_signal_to, 436 Continuous L4Re:Mem_alloc, 805 control L4:Thread, 915 copy L4Re::Dataspace_svr, 577 copy in L4Re:Dataspace_svr, 577 copy in L4Re:Dataspace_svr, 577 copy in L4Re:Dataspace, 570 count L4.:Factory, 644 create_buffer L4Re:Video::Goos, 668 create_factory L4::Factory, 647 create_inq L4::Factory, 647 create_ing L4::Factory, 648 create_ing L4::Factory, 649 create_lask L4::Factory, 649 create_lask L4::Factory, 645	· -	-
A tracebuffer_status_t, 748 Color_component L4Re::Video::Color_component, 555 Com_error L4::Com_error, 560 Conflortable Command Line Parsing, 360 parse_cmdline, 360 Console API, 96 Consumer, 424, 433 Ashmc_chunk consumed, 425 Hashmc_enable_chunk, 424 Hashmc_enable_chunk, 424 Hashmc_is_chunk ready, 425 Hashmc_wait_any_to, 434 Hashmc_wait_any_to, 434 Hashmc_wait_any_to, 434 Hashmc_wait_chunk, 10, 425 Hashmc_wait_chunk, 10, 425 Hashmc_wait_signal, 433 Hashmc_wait_any_to, 434 Hashmc_wait_signal, 434 Hashm		
Color_component		• —
L4Re::Video::Color_component, 555 Com error L4::Com_error, 560 Comfortable Command Line Parsing, 360 parse_cmdline, 360 Console API, 96 Console API, 96 Consumer, 424, 433 l4shmc_chunk consumed, 425 l4shmc_enable_chunk, 424 l4shmc_enable_chunk, 424 l4shmc_wait_any, 433 l4shmc_wait_any, 433 l4shmc_wait_any, 433 l4shmc_wait_any, 434 l4shmc_wait_any_try, 434 l4shmc_wait_any_try, 434 l4shmc_wait_signal_to, 436 Continuous L4Re::Mem_alloc, 805 control L4::Thread, 915 copy L4Re::Util::Dataspace_svr, 577 copy_in L4Re::Util::Dataspace_svr, 577 copy_in L4::Factory, 644 create_buffer L4Re::Video::Goos, 668 create_lactory L4::Factory, 647 create_gate L4::Factory, 648 create_lask L4::Factory, 649 create_lask L4::Factory, 649 create_lask L4::Factory, 645 creat		
Com_error L4::Com_error, 560 Comfortable Command Line Parsing, 360 parse cmdline, 360 Console API, 96 Consumer, 424, 433 l4shmc_chunk_consumed, 425 l4shmc_enable_chunk, 424 l4shmc_enable_signal, 433 l4shmc_wait_any, 433 l4shmc_wait_any, 433 l4shmc_wait_any to, 434 l4shmc_wait_any to, 434 l4shmc_wait_any to, 434 l4shmc_wait_signal_to, 436 control L4::Thread, 915 copy L4Re::Utili:Dataspace_svr, 577 copy_in L4Re::Dataspace, 570 count L4::Factory, 644 create_buffer L4Re::Videc::Goos, 668 create_L4::Factory, 644 create_buffer L4Re::Videc::Goos, 668 create_L4::Factory, 644 create_buffer L4::Factory, 647 create_gate L4::Factory, 648 create_lask L4::Factory, 645 create_thread L4::Factory, 645 create_thread cx::Alv_set <le>lem, Compare, Alloc >, 466 cx::Alv_set</le> lem, Compare, Alloc >, 466 cx::Alv_set lem, Compare, Alloc >, 466	_ ·	- · · ·
L4:Com_error, 560	— ·	
Comfortable Command Line Parsing, 360 parse cmdline, 360 Console API, 96 Consumer, 424, 433 4shmc_chunk_consumed, 425 4shmc_chunk_consumed, 425 4shmc_enable_chunk, 424 4shmc_enable_signal, 433 4shmc_enable_signal, 433 4shmc_wait_any, 433 4shmc_wait_any, 433 4shmc_wait_any, 434 4shmc_wait_any, 434 4shmc_wait_chunk, 424 4shmc_wait_signal, 434 4shmc_wait_signal, 434 4shmc_wait_signal try, 436 Continuous L4Re::Util::Dataspace_svr, 577 copy_in L4Re::Util::Dataspace_svr, 577 copy_in L4Re::Dataspace, 570 Count L4::Factory, 644 Create_tactory L4::Factory, 646 Create_tactory L4::Factory, 648 Create_tactory L4::Factory, 648 Create_tactory L4::Factory, 649 Create_tack L4::Factory, 645 Create_tread Cxx::AvVI_set Cxx::AvVI_set L4::Factory, 645 Cxx::AvVI_set Cxx::AvVI_set Cxx::AvVI_set L4::Factory, 645 Cxx::AvVI_set Cx		
parse_cmdline, 360	- · · ·	· —
Console API, 96 Consumer, 424, 433 Idshmc_chunk, consumed, 425 Idshmc_enable_chunk, 424 Idshmc_enable_signal, 433 Idshmc_is_chunk_ready, 425 Idshmc_wait_any, 433 Idshmc_wait_any, 433 Idshmc_wait_any, 434 Idshmc_wait_any, to, 434 Idshmc_wait_any, to, 434 Idshmc_wait_chunk, 424 Idshmc_wait_chunk, 424 Idshmc_wait_chunk, 424 Idshmc_wait_chunk, 425 Idshmc_wait_signal_to, 434 Idshmc_wait_signal_to, 434 Idshmc_wait_signal_to, 434 Idshmc_wait_signal_to, 434 Idshmc_wait_signal_to, 434 Idshmc_wait_signal_try, 436 Continuous L4Re::Mem_alloc, 805 control L4::Thread, 915 copy L4Re::Util::Dataspace_svr, 577 copy_in L4Re::Dataspace_svr, 577 copy_in L4Re::Dataspace_svr, 577 copy_in L4::Factory, 644 create_buffer L4::Factory, 647 create_gate L4::Factory, 648 create_gate L4::Factory, 649 create_tread L4::Factory, 645 create_tread Cxx::Avl_set Avl_set, 469, 470 Ave_bed, 473 Ave_bed, 470 Ave_bed, 473 Ave_bed, 470 Ave_bed, 471 Ave_bed, 470 Ave_bed, 471 Ave_bed, 471 Ave_bed, 472 Ave_bed, 471 Ave_bed, 472 Ave_bed, 472 Ave_bed, 474 Ave_bed, 475 Ave_bed, 470 Ave_bed, 470 Ave_bed, 471 Ave_bed, 472 Ave_bed, 472 Ave_bed, 470 Ave_bed, 471 Ave_bed, 472 Ave_bed, 472 Ave_bed, 472 Ave_bed, 473 Ave_bed, 474 Ave_bed, 475 Ave_bed, 475 Ave_bed, 470 Ave_bed, 4	——————————————————————————————————————	
Hshmc_chunk_consumed, 425	Console API, 96	
Hshmc_chunk_consumed, 425	Consumer, 424, 433	Bits, 501
Hshmc_enable_chunk, 424		Low_mask, 501
Hshmc_enable_signal, 433		
	l4shmc_enable_chunk, 424	Mask, 501
	l4shmc_enable_signal, 433	Msb, 501
	l4shmc_is_chunk_ready, 425	cxx::Bits::Direction
	l4shmc_wait_any, 433	L, 592
	l4shmc_wait_any_to, 434	N, 592
	l4shmc_wait_any_try, 434	R, 592
	l4shmc_wait_chunk, 424	cxx::Auto_ptr
	l4shmc_wait_chunk_to, 425	~Auto_ptr, 458
Idshmc_wait_signal_to, 434	l4shmc_wait_chunk_try, 425	Auto_ptr, 458
Idshmc_wait_signal_try, 436	l4shmc_wait_signal, 434	get, 459
Continuous operator->, 459 L4Re::Mem_alloc, 805 operator=, 458 control Ref_type, 458 L4::Thread, 915 release, 459 copy cxx::Auto_ptr< T >, 456 L4Re::Util::Dataspace_svr, 577 cxx::Avl_map copy_in pad0, 465 L4Re::Dataspace, 570 erase, 465 count find, 464 L4::Kip::Mem_desc, 808, 809 find_node, 463 l4_vhw_descriptor, 759 lower_bound_node, 464 create remove, 464 L4::Factory, 644 cxx::Avl_set create_buffer cxx::Avl_set L4Re::Video::Goos, 668 Avl_set, 469, 470 create_factory begin, 472, 473 L4::Factory, 647 end, 473 create_gate find_node, 471 L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set < Item, Compare, Alloc >, 466	l4shmc_wait_signal_to, 434	operator Priv_type *, 460
L4Re::Mem_alloc, 805 control	l4shmc_wait_signal_try, 436	operator*, 459
control L4::Thread, 915 copy	Continuous	operator->, 459
L4::Thread, 915 copy	L4Re::Mem_alloc, 805	operator=, 458
copy cxx::Auto_ptr< T >, 456 L4Re::Util::Dataspace_svr, 577 cxx::Avl_map copy_in pad0, 465 L4Re::Dataspace, 570 erase, 465 count find, 464 L4::Kip::Mem_desc, 808, 809 find_node, 463 I4_vhw_descriptor, 759 lower_bound_node, 464 create remove, 464 L4::Factory, 644 cxx::Avl_map< Key, Data, Compare, Alloc >, 460 create_buffer cxx::Avl_set L4Re::Video::Goos, 668 Avl_set, 469, 470 create_factory begin, 472, 473 L4::Factory, 647 end, 473 create_gate find_node, 471 L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 remove, 471 create_thread cxx::Avl_set ctm, 474 create_thread	control	Ref_type, 458
L4Re::Util::Dataspace_svr, 577 copy_in	L4::Thread, 915	release, 459
copy_in pad0, 465 L4Re::Dataspace, 570 erase, 465 count find, 464 L4::Kip::Mem_desc, 808, 809 find_node, 463 l4_vhw_descriptor, 759 lower_bound_node, 464 create remove, 464 L4::Factory, 644 cxx::Avl_map Key, Data, Compare, Alloc >, 460 create_buffer cxx::Avl_set Avl_set, 469, 470 create_factory begin, 472, 473 end, 473 create_gate find_node, 471 insert, 470 create_gate find_node, 471 insert, 470 create_irq lower_bound_node, 472 rbegin, 474 create_task remove, 471 remove, 471 L4::Factory, 645 rend, 474 rend, 474 create_thread cxx::Avl_set< Item, Compare, Alloc >, 466	сору	cxx::Auto_ptr< T >, 456
L4Re::Dataspace, 570 erase, 465 count find, 464 L4::Kip::Mem_desc, 808, 809 find_node, 463 I4_vhw_descriptor, 759 lower_bound_node, 464 create remove, 464 L4::Factory, 644 cxx::Avl_map< Key, Data, Compare, Alloc >, 460 create_buffer cxx::Avl_set L4Re::Video::Goos, 668 Avl_set, 469, 470 create_factory begin, 472, 473 L4::Factory, 647 end, 473 create_gate find_node, 471 L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set	L4Re::Util::Dataspace_svr, 577	cxx::Avl_map
count find, 464 L4::Kip::Mem_desc, 808, 809 find_node, 463 l4_vhw_descriptor, 759 lower_bound_node, 464 create remove, 464 L4::Factory, 644 cxx::Avl_map< Key, Data, Compare, Alloc >, 460 create_buffer cxx::Avl_set L4Re::Video::Goos, 668 Avl_set, 469, 470 create_factory begin, 472, 473 L4::Factory, 647 end, 473 create_gate find_node, 471 L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set cxx::Avl_set ltem, Compare, Alloc >, 466	copy_in	pad0, 465
L4::Kip::Mem_desc, 808, 809 find_node, 463 l4_vhw_descriptor, 759 lower_bound_node, 464 create remove, 464 L4::Factory, 644 cxx::Avl_map< Key, Data, Compare, Alloc >, 460 create_buffer cxx::Avl_set L4Re::Video::Goos, 668 Avl_set, 469, 470 create_factory begin, 472, 473 L4::Factory, 647 end, 473 create_gate find_node, 471 L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set< Item, Compare, Alloc >, 466	L4Re::Dataspace, 570	
I4_vhw_descriptor, 759		
create remove, 464 L4::Factory, 644 cxx::Avl_map < Key, Data, Compare, Alloc >, 460 create_buffer cxx::Avl_set L4Re::Video::Goos, 668 Avl_set, 469, 470 create_factory begin, 472, 473 L4::Factory, 647 end, 473 create_gate find_node, 471 L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set < Item, Compare, Alloc >, 466	· — · · · ·	
L4::Factory, 644 cxx::Avl_map< Key, Data, Compare, Alloc >, 460 create_buffer cxx::Avl_set L4Re::Video::Goos, 668 Avl_set, 469, 470 create_factory begin, 472, 473 L4::Factory, 647 end, 473 create_gate find_node, 471 L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set< Item, Compare, Alloc >, 466	I4_vhw_descriptor, 759	
create_buffer cxx:::Avl_set L4Re::Video::Goos, 668 Avl_set, 469, 470 create_factory begin, 472, 473 L4::Factory, 647 end, 473 create_gate find_node, 471 L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set < Item, Compare, Alloc >, 466		
L4Re::Video::Goos, 668 Avl_set, 469, 470 create_factory begin, 472, 473 L4::Factory, 647 end, 473 create_gate find_node, 471 L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set < Item, Compare, Alloc >, 466	•	
create_factory begin, 472, 473 L4::Factory, 647 end, 473 create_gate find_node, 471 L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set < Item, Compare, Alloc >, 466		
L4::Factory, 647 end, 473 create_gate find_node, 471 L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set < Item, Compare, Alloc >, 466		
create_gate find_node, 471 L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set < Item, Compare, Alloc >, 466	- •	_
L4::Factory, 648 insert, 470 create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set < Item, Compare, Alloc >, 466	•	•
create_irq lower_bound_node, 472 L4::Factory, 649 rbegin, 474 create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set < Item, Compare, Alloc >, 466		
L4::Factory, 649 create_task L4::Factory, 645 create_thread rbegin, 474 remove, 471 rend, 474 cxx::Avl_set< Item, Compare, Alloc >, 466		
create_task remove, 471 L4::Factory, 645 rend, 474 create_thread cxx::Avl_set< Item, Compare, Alloc >, 466	- ·	
L4::Factory, 645 rend, 474 create_thread cxx::Avl_set < Item, Compare, Alloc >, 466	-	-
create_thread cxx::Avl_set< Item, Compare, Alloc >, 466		
_ ·	•	•
L4::Factory, 646 cxx::Avi_set < Item, Compare, Alloc >::Node, 827		<u> </u>
	L4::Factory, 646	cxx::Avi_set< item, Compare, Alloc >::Node, 827

cxx::Avl_set::Node	dir, 522, 523
valid, 828	end, 524, 525
cxx::AvI_tree	find, 528
insert, 477	find_node, 526
Iterator, 477	lower_bound_node, 527
remove, 478	rbegin, 525, 526
cxx::AvI_tree < Node, Get_key, Compare >, 475	rend, 525, 526
cxx::Avl_tree_node, 478	cxx::Bits::Bst< Node, Get_key, Compare >, 518
cxx::Base_slab	cxx::Bits::Bst_node, 529
free_objects, 485	cxx::Bits::Direction, 590
total_objects, 485	Direction_e, 592
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc	cxx::List
>, 483	items, 792
cxx::Base_slab_static	push_back, 792
alloc, 489	push_front, 792
free, 489	remove, 792
free_objects, 489	size, 792
total_objects, 489	cxx::List< D, Alloc >, 791
cxx::Base_slab_static< Obj_size, Slab_size, Max_free,	cxx::List< D, Alloc >::Iter, 721
Alloc >, 486	cxx::List_alloc, 793
cxx::Bitfield	alloc, 794
Bits_type, 500	avail, 794
get, 501	free, 794
get_unshifted, 502	List_alloc, 793
Masks, 501	cxx::List_item, 794
Ref, 500	get_next_item, 796
Ref_unshifted, 501	get_prev_item, 796
set, 504	insert_next_item, 796
set_dirty, 502	insert_prev_item, 796
set_unshifted, 504	push_back, 797
set_unshifted_dirty, 503	push_front, 797
Shift_type, 500	remove, 797
Val, 501	remove_me, 796
val, 506	cxx::List_item::Iter, 718
val_dirty, 506	remove_me, 720
Val_unshifted, 501	cxx::List_item::T_iter< T, Poly >, 898
val unshifted, 507	cxx::Lt_functor< Obj >, 802
cxx::Bitfield< T, LSB, MSB >, 498	cxx::New_allocator< _Type >, 826
cxx::Bitfield< T, LSB, MSB >::Value< TT >, 924	cxx::Nothrow, 828
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >, 925	cxx::Pair
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >,	Pair, 839
927	cxx::Pair< First, Second >, 838
cxx::Bitmap	cxx::Pair_first_compare
Bitmap, 510	operator(), 840
•	Pair_first_compare, 840
clear_all, 510	cxx::Pair_first_compare< Cmp, Typ >, 839
cxx::Bitmap < BITS >, 508	cxx::Slab
cxx::Bitmap_base, 510	alloc, 884
bit, 513	free, 884
chars, 513	cxx::Slab< Type, Slab_size, Max_free, Alloc >, 881
clear_bit, 513	cxx::Slab_static
scan_zero, 515	alloc, 888
set_bit, 514	cxx::Slab_static< Type, Slab_size, Max_free, Alloc >,
words, 513	884
cxx::Bitmap_base::Char< BITS >, 553	
cxx::Bitmap_base::Word< BITS >, 955	d_val
cxx::Bits, 442	Elf32_Dyn, 603
cxx::Bits::Bst	Elf64_Dyn, 609
begin, 523, 524	DF_1_CONFALT

ELF binary format, 357	l4re_debug_obj_debug, 61
DF_1_DIRECT	Debugging API, 98
ELF binary format, 356	dec_refcnt
DF_1_DISPRELDNE	L4::Kobject, 723
ELF binary format, 357	Default
DF_1_DISPRELPND	L4Re Protocol identifiers, 114
ELF binary format, 357	delete_buffer
DF_1_ENDFILTEE	L4Re::Video::Goos, 668
ELF binary format, 357	delete_obj
DF_1_GLOBAL	L4::Task, 906
ELF binary format, 356	delete_view
DF_1_GROUP	L4Re::Video::Goos, 669
ELF binary format, 356	descs
DF_1_INTERPOSE	I4_vhw_descriptor, 759
ELF binary format, 356	detach
DF_1_LOADFLTR	L4::lrq, 707
ELF binary format, 356	L4Re::Rm, 862, 864
DF_1_NODEFLIB	L4Re::Util::Event_buffer_t, 637
ELF binary format, 357	Detach_again
DF_1_NODELETE	L4Re::Rm, 858
ELF binary format, 356	Detach_exact
DF_1_NODUMP	L4Re::Rm, 858
ELF binary format, 357	Detach_free
DF 1 NOOPEN	L4Re::Rm, 858
ELF binary format, 356	Detach_keep
DF 1 NOW	L4Re::Rm, 858
ELF binary format, 356	Detach_overlap
DF 1 ORIGIN	L4Re::Rm, 858
ELF binary format, 356	Detach_flags
DF P1 GROUPPERM	L4Re::Rm, 858
ELF binary format, 357	Detach_result
DF P1 LAZYLOAD	L4Re::Rm, 857
ELF binary format, 357	Detached_ds
DT HIPROC	L4Re::Rm, 857
ELF binary format, 355	di
DT LOPROC	14_vcpu_regs_t, 753
ELF binary format, 355	dir
DT NULL	cxx::Bits::Bst, 522, 523
ELF binary format, 355	Direction_e
Data-Space API, 97	cxx::Bits::Direction, 592
data_space	dispatch
L4Re::Vfs::Be_file, 495	L4::Basic_registry, 491
L4Re::Vfs::Be_file, 453 L4Re::Vfs::Regular_file, 853	L4::Server_object, 880
	L4Re::Util::Vcon_svr, 936
Dataspace	L4Re::Util::Video::Goos_svr, 672
L4Re Protocol identifiers, 114	drive_cylinders
Dataspace interface, 58	l4util_mb_drive_t, 785
l4re_ds_allocate, 59	drive_mode
l4re_ds_clear, 58	I4util_mb_drive_t, 784
l4re_ds_copy_in, 59	drive_number
l4re_ds_flags, 59	l4util_mb_drive_t, 784
l4re_ds_info, 59	dump
l4re_ds_phys, 59	L4Re::Video::Color_component, 557
l4re_ds_size, 59	L4Re::Video::Pixel_info, 847
Debug	dx
L4Re Protocol identifiers, 114	l4_vcpu_regs_t, 754
debug	
L4Re::Debug_obj, 581	e_phnum
Debug interface, 61	Elf32_Ehdr, 605

Elf64_Ehdr, 611	PT_GNU_RELRO, 355
e_shnum	PT_GNU_STACK, 355
Elf32_Ehdr, 605	PT_HIOS, 354
Elf64_Ehdr, 611	PT_HIPROC, 354
EI_CLASS	PT_L4_AUX, 355
ELF binary format, 350	PT_L4_KIP, 355
EI_DATA	PT_L4_STACK, 355
ELF binary format, 350	PT_LOOS, 354
EI_OSABI	PT_LOPROC, 354
ELF binary format, 351, 352	SHF_GROUP, 354
EI_PAD	SHF_MASKOS, 354
ELF binary format, 353	SHF_TLS, 354
EI_VERSION	SHT_NUM, 354
ELF binary format, 351	ELFCLASSNONE
ELF binary format, 339	ELF binary format, 350
DF_1_CONFALT, 357	ELFDATA2LSB
DF_1_DIRECT, 356	ELF binary format, 351
DF_1_DISPRELDNE, 357	ELFDATA2MSB
DF_1_DISPRELPND, 357	ELF binary format, 351
DF_1_ENDFILTEE, 357	ELFDATANONE
DF_1_GLOBAL, 356	ELF binary format, 350, 351
DF_1_GROUP, 356	ELFOSABI_AIX
DF_1_INTERPOSE, 356	ELF binary format, 353
DF_1_LOADFLTR, 356	ELFOSABI_FREEBSD
DF_1_NODELETE_356	ELF binary format, 353
DF_1_NODELETE, 356	ELFOSABI_HPUX
DF_1_NODUMP, 357	ELF binary format, 352
DF_1_NOOPEN, 356 DF_1_NOW, 356	ELFOSABI_IRIX ELF binary format, 353
DF_1_NOW, 356 DF_1_ORIGIN, 356	ELFOSABI LINUX
DF_P1_GROUPPERM, 357	ELF binary format, 352
DF_P1_LAZYLOAD, 357	ELFOSABI MODESTO
DT_HIPROC, 355	ELF binary format, 353
DT_INITIOO, 355	ELFOSABI NETBSD
DT_RULL, 355	ELF binary format, 352
EI_CLASS, 350	ELFOSABI OPENBSD
EI_DATA, 350	ELF binary format, 353
EI_OSABI, 351, 352	ELFOSABI SOLARIS
EI PAD, 353	ELF binary format, 352
EI_VERSION, 351	ELFOSABI_SYSV
ELFCLASSNONE, 350	ELF binary format, 352
ELFDATA2LSB, 351	ELFOSABI TRU64
ELFDATA2MSB, 351	ELF binary format, 353
ELFDATANONE, 350, 351	EM ARC
ELFOSABI AIX, 353	ELF binary format, 353
ELFOSABI FREEBSD, 353	Eager map
ELFOSABI HPUX, 352	L4Re::Rm, 858
ELFOSABI IRIX, 353	Elf32_Dyn, 603
ELFOSABI_LINUX, 352	d_val, 603
ELFOSABI MODESTO, 353	Elf32 Ehdr, 604
ELFOSABI_NETBSD, 352	e_phnum, 605
ELFOSABI_OPENBSD, 353	e_shnum, 605
ELFOSABI_SOLARIS, 352	Elf32_Phdr, 605
ELFOSABI_SYSV, 352	Elf32_Shdr, 607
ELFOSABI_TRU64, 353	Elf32_Sym, 608
EM_ARC, 353	Elf64_Dyn, 609
NT_VERSION, 355	d_val, 609
PT_GNU_EH_FRAME, 354	Elf64_Ehdr, 610

e phnum, 611	L4 IPC SND ERR MASK, 219
e_shnum, 611	14 error, 220
Elf64 Phdr, 611	14 jpc error, 219
Elf64 Shdr, 613	14_ipc_error_code, 222
Elf64_Sym, 614	I4 ipc is rcv error, 222
	14_ipc_is_snd_error, 221
end	14_ipc_tcr_error_t, 218
cxx::Avl_set, 473	-·
cxx::Bits::Bst, 524, 525	Event
L4::Kip::Mem_desc, 810	L4Re Protocol identifiers, 114
enter_kdebug	Event API, 107
Kernel Debugger, 181, 182	Event interface, 62
entry_ip	l4re_event_get_axis_info, 64
L4vcpu::Vcpu, 945	l4re_event_get_buffer, 62
entry_sp	l4re_event_get_num_streams, 62
L4vcpu::Vcpu, 945	l4re_event_get_stream_info, 64
env	l4re_event_get_stream_info_for_id, 64
L4Re::Env, 617	Event_buffer_t
erase	L4Re::Event_buffer_t, 633
cxx::Avl_map, 465	ex_regs
Error codes, 189	L4::Thread, 913, 914
L4 EACCESS, 189	exc handler
L4_EADDRNOTAVAIL, 190	L4::Thread::Attr, 452
	Exception registers, 298
L4_EAGAIN, 189	14_utcb_exc, 298
L4_EBADPROTO, 190	I4_utcb_exc_is_pf, 299
L4_EBUSY, 189	14_utcb_exc_pc, 299
L4_EEXIST, 189	14_utcb_exc_pc_set, 299
L4_EINVAL, 189	_ _
L4_EIO, 189	ext_alloc
L4_EIPC_HI, 190	L4vcpu::Vcpu, 947
L4_EIPC_LO, 190	Extended vCPU support, 413
L4_ENAMETOOLONG, 189	l4vcpu_ext_alloc, 413
L4_ENODEV, 189	F above
L4_ENOENT, 189	L4Re::Video::View, 949
L4_ENOMEM, 189	
L4_ENOREPLY, 190	F_auto_refresh
L4_ENOSYS, 190	L4Re::Video::Goos, 667
L4 EOK, 189	F_dyn_allocated
L4_EPERM, 189	L4Re::Video::View, 949
L4_ERANGE, 189	F_dynamic_buffers
L4 ERRNOMAX, 190	L4Re::Video::Goos, 667
l4 error code t, 189	F_dynamic_views
:	L4Re::Video::Goos, 667
Error Handling, 218	F_flags_mask
L4_IPC_ENOT_EXISTENT, 219	L4Re::Video::View, 949
L4_IPC_ERROR_MASK, 219	F_fully_dynamic
L4_IPC_REABORTED, 219	L4Re::Video::View, 949
L4_IPC_RECANCELED, 219	F_l4re_video_goos_auto_refresh
L4_IPC_REMAPFAILED, 219	Video API, 91
L4_IPC_REMSGCUT, 219	F_I4re_video_goos_dynamic_buffers
L4_IPC_RERCVPFTO, 219	Video API, 91
L4_IPC_RESNDPFTO, 219	F_I4re_video_goos_dynamic_views
L4_IPC_RETIMEOUT, 219	Video API, 91
L4_IPC_SEABORTED, 219	F_l4re_video_goos_pointer
L4_IPC_SECANCELED, 219	Video API, 91
L4 IPC SEMAPFAILED, 219	F_l4re_video_view_above
L4 IPC SEMSGCUT, 219	Video API, 91
L4_IPC_SERCVPFTO, 219	F_l4re_video_view_dyn_allocated
L4 IPC SESNDPFTO, 219	Video API, 91
L4_IPC_SETIMEOUT, 219	F_l4re_video_view_flags_mask
LT_II O_OL I IIVILOO I ,	i _i=io_viaco_view_iiays_iiiasn

Video API, 91	fiasco_tbuf_clear, 136
F_I4re_video_view_none	fiasco_tbuf_dump, 136
Video API, 91	fiasco_tbuf_get_status, 134
F_l4re_video_view_set_background	fiasco_tbuf_get_status_phys, 134
Video API, 91	fiasco_tbuf_log, 134
F_l4re_video_view_set_buffer	fiasco_tbuf_log_3val, 135
Video API, 91	fiasco_tbuf_log_binary, 136
F_l4re_video_view_set_buffer_offset	fiasco_watchdog_takeover, 137
Video API, 91	fiasco_watchdog_touch, 137
F_l4re_video_view_set_bytes_per_line	Fiasco real time scheduling extensions, 140
Video API, 91	Fiasco-UX Virtual devices
F_l4re_video_view_set_flags	L4_TYPE_VHW_FRAMEBUFFER, 307
Video API, 91	L4_TYPE_VHW_INPUT, 307
F_l4re_video_view_set_pixel	L4_TTT L_VTW_INT 01, 307 L4 TYPE VHW NET, 307
Video API, 91	L4_TYPE_VHW_NET, 307 L4_TYPE_VHW_NONE, 307
F_l4re_video_view_set_position Video API, 91	Fiasco-UX Virtual devices, 307
	I4_vhw_entry_type, 307
F_none	fiasco_gdt_get_entry_offset
L4Re::Video::View, 949	Fiasco extensions, 138
F_pointer	fiasco_gdt_set
L4Re::Video::Goos, 667	Fiasco extensions, 137
F_set_background	fiasco_ldt_set
L4Re::Video::View, 949	Fiasco extensions, 137
F_set_buffer	fiasco_tbuf_clear
L4Re::Video::View, 949	Fiasco extensions, 136
F_set_buffer_offset	fiasco_tbuf_dump
L4Re::Video::View, 949	Fiasco extensions, 136
F_set_bytes_per_line	fiasco_tbuf_get_status
L4Re::Video::View, 949	Fiasco extensions, 134
F_set_flags	fiasco_tbuf_get_status_phys
L4Re::Video::View, 949	Fiasco extensions, 134
F_set_pixel	fiasco_tbuf_log
L4Re::Video::View, 949	Fiasco extensions, 134
F_set_position	fiasco_tbuf_log_3val
L4Re::Video::View, 949	Fiasco extensions, 135
faccessat	fiasco_tbuf_log_binary
L4Re::Vfs::Directory, 595	Fiasco extensions, 136
Factory, 191	fiasco_watchdog_takeover
<pre>I4_factory_create_factory, 194</pre>	Fiasco extensions, 137
I4_factory_create_gate, 194	fiasco_watchdog_touch
I4_factory_create_irq, 195	Fiasco extensions, 137
l4_factory_create_task, 192	find
I4_factory_create_thread, 193	cxx::Avl_map, 464
I4_factory_create_vm, 196	cxx::Bits::Bst, 528
factory	L4Re::Rm, 864
L4Re::Env, 618, 621	find_node
fchmod	cxx::Avl_map, 463
L4Re::Vfs::Generic_file, 663	cxx::Avl_set, 471
fd	cxx::Bits::Bst, 526
I4_vhw_entry, 761	first
fdatasync	L4::Kip::Mem_desc, 808
L4Re::Vfs::Regular_file, 854	first_free_cap
features	L4Re::Env, 618, 621
I4_icu_info_t, 734	first_free_utcb
Fiasco extensions, 132	L4Re::Env, 619, 622
fiasco_gdt_get_entry_offset, 138	Flags
fiasco_gdt_set, 137	L4Re::Video::Goos, 667
fiasco_ldt_set, 137	L4Re::Video::View, 949

flags	free_objects
I4_exc_regs_t, 731	cxx::Base_slab, 485
I4_msgtag_t, 739	cxx::Base_slab_static, 489
L4Re::Dataspace, 572	fstat64
L4Re::Video::View::Info, 687	L4Re::Vfs::Be_file, 495
l4re_env_cap_entry_t, 771	L4Re::Vfs::Generic_file, 663
Flex pages, 141	fsync
L4_CAP_FPAGE_R, 143	L4Re::Vfs::Regular_file, 854
L4_CAP_FPAGE_RO, 143	ftruncate64
L4_CAP_FPAGE_RW, 143	L4Re::Vfs::Regular_file, 853
L4_FPAGE_ADDR_BITS, 143	Functions for rendering bitmap data in frame buffers, 375
L4_FPAGE_ADDR_SHIFT, 143	gfxbitmap_bmap, 376
L4_FPAGE_BUFFERABLE, 143	gfxbitmap_color_pix_t, 376
L4_FPAGE_CACHE_OPT, 143	gfxbitmap_color_t, 375
L4_FPAGE_CACHEABLE, 143	gfxbitmap_convert_color, 376
L4_FPAGE_RIGHTS_BITS, 143	gfxbitmap_copy, 377
L4_FPAGE_RIGHTS_SHIFT, 142	gfxbitmap_fill, 376
L4_FPAGE_RO, 143	gfxbitmap_set, 377
L4_FPAGE_RW, 143	Functions for rendering bitmap fonts to frame buffers,
L4_FPAGE_SIZE_BITS, 143	378
L4_FPAGE_SIZE_SHIFT, 143	gfxbitmap_font_data, 380
L4_FPAGE_TYPE_BITS, 143	gfxbitmap_font_get, 379
L4_FPAGE_TYPE_SHIFT, 142	gfxbitmap_font_height, 379
L4_FPAGE_UNCACHEABLE, 143	gfxbitmap_font_init, 379 gfxbitmap_font_text, 380
L4_IOPORT_MAX, 144	gfxbitmap_font_text_scale, 380
L4_WHOLE_ADDRESS_SPACE, 143	gfxbitmap_font_text_scale, 360 gfxbitmap_font_width, 379
L4_WHOLE_IOADDRESS_SPACE, 144	Functions to manipulate the local IDT, 318
L4_cap_fpage_rights, 143	Turictions to manipulate the local IDT, 010
I4_fpage, 144	g
l4_fpage_all, 144	L4Re::Video::Pixel_info, 845, 846
I4_fpage_cacheability_opt_t, 143	get
I4_fpage_consts, 142	cxx::Auto_ptr, 459
I4_fpage_contains, 148	cxx::Bitfield, 501
I4_fpage_invalid, 144	L4::lpc::lstream, 714, 715
I4_fpage_max_order, 148	L4Re::Env, 619
I4_fpage_page, 147	L4Re::Video::Color_component, 556
L4_fpage_rights, 143	get_attr
I4_fpage_rights, 146	L4::Vcon, 934
I4_fpage_set_rights, 147	get_buffer
I4_fpage_size, 147	L4Re::Event, 626
I4_fpage_type, 146	get_cap
I4_iofpage, 144 I4_is_fpage_writable, 145	L4Re::Env, 620
l4_obj_fpage, 145	get_cap_alloc
foreach available event	L4Re::Cap_alloc, 543
L4Re::Util::Event_buffer_consumer_t, 630	get_fb
	L4Re::Util::Video::Goos_svr, 671
fpage L4::Cap_base, 550	get_file
free	L4Re::Vfs::Fs, 659
cxx::Base_slab_static, 489	get_infos
cxx::List_alloc, 794	L4::lpc_gate, 702
cxx::Slab, 884	get_lock
L4Re::Cap_alloc, 543	L4Re::Vfs::Regular_file, 854
L4Re::Mem alloc, 806	get_next_item cxx::List_item, 796
free area	get_object_name
L4Re::Rm, 860	L4::Debugger, 587
free fd	get_prev_item
L4Re::Vfs::Fs, 659	cxx::List_item, 796
= 11.0 ¥10 0, 000	OMMEDICATION, 100

get_static_buffer	L4vcpu::Vcpu, 945
L4Re::Video::Goos, 668	IO interface
get_status_flags	L4IO_DEVICE_ANY, 383
L4Re::Vfs::Generic_file, 663	L4IO_DEVICE_INVALID, 383
get_unshifted	L4IO_DEVICE_OTHER, 383
cxx::Bitfield, 502	L4IO_DEVICE_PCI, 383
gfxbitmap_bmap	L4IO_DEVICE_USB, 383
Functions for rendering bitmap data in frame	L4IO_MEM_CACHED, 383
buffers, 376	L4IO_MEM_EAGER_MAP, 383
gfxbitmap_color_pix_t	L4IO_MEM_NONCACHED, 383
Functions for rendering bitmap data in frame	L4IO_MEM_USE_MTRR, 383
buffers, 376	L4IO_MEM_USE_RESERVED_AREA, 383
gfxbitmap_color_t	L4IO_RESOURCE_ANY, 383
Functions for rendering bitmap data in frame	L4IO_RESOURCE_INVALID, 383
buffers, 375	L4IO_RESOURCE_IRQ, 383
gfxbitmap_convert_color	L4IO_RESOURCE_MEM, 383
Functions for rendering bitmap data in frame	L4IO_RESOURCE_PORT, 383
buffers, 376	IPC-Gate API
gfxbitmap_copy Functions for rendering bitmap data in frame	L4_IPC_GATE_BIND_OP, 127
buffers, 377	L4_IPC_GATE_GET_INFO_OP, 127
gfxbitmap fill	IRQs
Functions for rendering bitmap data in frame	L4_IRQ_F_BOTH, 225
buffers, 376	L4_IRQ_F_BOTH_EDGE, 225
gfxbitmap_font_data	L4_IRQ_F_CLEAR_WAKEUP, 225
Functions for rendering bitmap fonts to frame	L4_IRQ_F_EDGE, 225 L4_IRQ_F_LEVEL, 225
buffers, 380	
gfxbitmap_font_get	L4_IRQ_F_LEVEL_HIGH, 225 L4_IRQ_F_LEVEL_LOW, 225
Functions for rendering bitmap fonts to frame	L4_IRQ_F_MASK, 225
buffers, 379	L4_IRQ_F_NEG, 225
gfxbitmap_font_height	L4_IRQ_F_NEG_EDGE, 225
Functions for rendering bitmap fonts to frame	L4_IRQ_F_NONE, 225
buffers, 379	L4_IRQ_F_POS, 225
gfxbitmap_font_init	L4 IRQ F POS EDGE, 225
Functions for rendering bitmap fonts to frame	L4_IRQ_F_SET_WAKEUP, 225
buffers, 379	IA32 Port I/O API, 370
gfxbitmap_font_text	l4util in16, 371
Functions for rendering bitmap fonts to frame	I4util_in32, 371
buffers, 380	I4util_in8, 370
gfxbitmap_font_text_scale	I4util_ins16, 371
Functions for rendering bitmap fonts to frame	I4util_ins32, 372
buffers, 380 gfxbitmap font width	I4util_ins8, 371
Functions for rendering bitmap fonts to frame	l4util_out16, 372
buffers, 379	I4util_out32, 372
gfxbitmap_offset, 664	l4util_out8, 372
gfxbitmap_set	l4util_outs16, 373
Functions for rendering bitmap data in frame	l4util_outs32, 373
buffers, 377	l4util_outs8, 372
global id	IO interface, 382
L4::Debugger, 584	l4io_device_types_t, 383
Goos	l4io_has_resource, 387
L4Re Protocol identifiers, 114	l4io_iomem_flags_t, 383
Goos video API, 119	l4io_lookup_device, 386
	l4io_lookup_resource, 387
has_alpha	l4io_release_iomem, 384
L4Re::Video::Pixel_info, 846	l4io_release_ioport, 386
·	l4io_request_iomem, 384
1	l4io_request_iomem_region, 384

l4io_request_ioport, 386	l4_int16_t, 438
l4io_request_resource_iomem, 387	l4_int32_t, 439
l4io_resource_t, 383	I4_int64_t, 439
l4io_resource_types_t, 383	I4_int8_t, 438
l4io_search_iomem_region, 384	l4_uint16_t, 438
IPC Messaging Framework, 53	l4_uint32_t, 439
buf_cp_in, 54	I4_uint64_t, 439
buf_cp_out, 53	l4_uint8_t, 438
buf_in, 54	interface
msg_ptr, 54	L4::Meta, 814
IPC Streams, 46	Interface for asynchronous ISR handlers with a given I-
operator<<, 49d	RQ capability., 398
operator>>, 46d	l4irq_request_cap, 398
IPC-Gate API, 126	Interface for asynchronous ISR handlers., 394
I4_ipc_gate_bind_thread, 127	l4irq_release, 395
I4_ipc_gate_get_infos, 127	l4irq_request, 394
L4_ipc_gate_ops, 127	Interface using direct functionality., 390, 396
IRQ handling library, 389	l4irq_attach, 390
IRQs, 224	l4irq_attach_cap, 396
I4_irq_attach, 225	l4irq_attach_cap_ft, 396
I4_irq_chain, 226	l4irq_attach_ft, 391
I4_irq_detach, 226	l4irq_attach_thread, 391
L4_irq_mode, 225	l4irq_attach_thread_cap, 397
I4_irq_receive, 228	l4irq_attach_thread_cap_ft, 397
I4_irg_trigger, 227	l4irq_attach_thread_ft, 391
I4_irg_unmask, 229	l4irq_detach, 393
I4_irq_wait, 229	l4irg_unmask, 392
idle_time	l4irq_unmask_and_wait_any, 392
L4::Scheduler, 875	l4irq_wait, 392
In_area	l4irq_wait_any, 392 Internal constants, 403
L4Re::Rm, 858 info	Internal functions, 333
L4::lcu, 678	internal_loop
L4::Icu, 676 L4::Scheduler, 874	L4::Server, 879
L4::Scheduler, 874 L4Re::Dataspace, 573	Interrupt controller, 198
L4Re::Video::Goos, 667	L4_ICU_FLAG_MSI, 199
L4Re::Video::View, 950	I4_icu_bind, 199
init	L4_icu_flags, 199
L4Re::Util::Event_t, 639	I4 icu info, 201
init_infos	I4_icu_info_t, 199
L4Re::Util::Video::Goos svr, 673	I4 icu mask, 202
Initial Environment, 101	I4_icu_msi_info, 201
l4re_env, 102	14_icu_set_mode, 200
l4re_env_get_cap, 103	14 icu unbind, 200
l4re_env_get_cap_e, 103	I4_icu_unmask, 202
l4re_env_get_cap_l, 105	Invalid
I4re_env_t, 102	L4::Cap_base, 547
I4re_kip, 103	Invalid_capability
initial_caps	L4::Invalid_capability, 692
L4Re::Env, 619, 622	ioctl
insert	L4Re::Vfs::Special_file, 895
cxx::Avl_set, 470	lostream
cxx::Avl_tree, 477	L4::lpc::lostream, 696
insert_next_item	irq
cxx::List_item, 796	L4Re::Util::Event_t, 639
insert_prev_item	irq_disable_save
cxx::List_item, 796	L4vcpu::Vcpu, 941
Integer Types, 437	irq_enable
	-

	L4vcpu::Vcpu, 942	
irq_restore L4::Debugger, 585 L4vcpu::Vcpu, 942 kobject_typeid is_irq_entry L4, 445 L4vcpu::Vcpu, 944 L4::Kobject_2t, 726 is_online L4::Kobject_2t, 726 L4::Scheduler, 876 L4::Kobject_t, 728 is_page_fault_entry Kumem allocator utility, 88 L4vcpu::Vcpu, 944 l4re_util_kumem_alloc, 88 is_static Kumem utilities, 118 L4::Cap_base, 549 kumem_alloc is_valid kumem_alloc L4::Cap_base, 549 Kumem utilities, 118 is_virtual L L4::Kip::Mem_desc, 811 L Istream cxx::Bits::Direction, 592 L4::Ipc::Istream, 713 L4, 442 items kobject_typeid, 445 cxx::List, 792 L4::Cap_base Iterator Invalid, 547 cxx::Avl_tree, 477 No_init, 547 kd_display Reply_compound, 446 Kept_ds L4_BASE_FACTORY_CAP Capabilities, 287 Kernel Debugger, 180 L4_BASE_FACTORY_CAP Capabilities, 287		Kernel Debugger, 182, 183
L4vcpu::Vcpu, 942 is_irq_entry	·	
is_irq_entry L4, 445 L4vcpu::Vcpu, 944 L4::Kobject, 724 is_online L4::Kobject_2t, 726 L4::Scheduler, 876 L4::Kobject_t, 728 is_page_fault_entry Kumem allocator utility, 88 L4vcpu::Vcpu, 944 I4re_util_kumem_alloc, 88 is_static Kumem utilitities, 118 L4Re::Util::Dataspace_svr, 578 kumem_alloc, 118 is_valid kumem_alloc L4::Cap_base, 549 Kumem utilities, 118 is_virtual L L4::Kip::Mem_desc, 811 L Istream L4::Ipc::Istream, 713 L4::Ipc::Istream, 713 L4, 442 items kobject_typeid, 445 Cxx::Avl_tree, 477 L4::Ipc_svr kd_display Reply_compound, 446 Kepr_ds L4_BASE_FACTORY_CAP Capabilities, 287 L4_BASE_FACTORY_CAP Capabilities, 287 L4_BASE_ICU_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_SCHEDULER_CAP		
L4vcpu::Vcpu, 944 is_online L4::Scheduler, 876 L4::Scheduler, 876 is_page_fault_entry L4vcpu::Vcpu, 944 is_static L4Re::Util::Dataspace_svr, 578 is_valid L4::Cap_base, 549 is_virtual L4::Kip::Mem_desc, 811 Istream L4::Ipc::Istream, 713 items cxx::List, 792 Iterator cxx::Avl_tree, 477 kG_display Kernel Debugger, 182, 183 Kept_ds L4Re::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 182 kd_display, 182, 183 ko, 182, 183 l4_ebbugger_global_id, 185 l4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_SCHEDULER_CAP		
is_online	· ·	
L4::Scheduler, 876 is_page_fault_entry L4vcpu::Vcpu, 944 is_static L4Re::Util::Dataspace_svr, 578 is_valid L4::Cap_base, 549 is_virtual L4::Kip::Mem_desc, 811 Istream L4::Ipc::Istream, 713 items cxx::List, 792 Iterator cxx::Avl_tree, 477 kd_display Kernel Debugger, 182, 183 Kept_ds L4Re::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd::Sapabase l4::Rbip::Base kumem_alloc, 118 kumem_alloc Kumem utilitities, 118 cxx::Bits::Direction, 592 L4, 442 kubject_typeid, 445 L4::Cap_base lnvalid, 547 No_init, 547 L4::Ipc_svr Reply_compound, 446 Reply_separate, 446 Reply_separate, 446 L4_BASE_FACTORY_CAP Capabilities, 287 L4_BASE_ICU_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_SCHEDULER_CAP	·	
is_page_fault_entry L4vcpu::Vcpu, 944 is_static L4Re::Util::Dataspace_svr, 578 is_valid L4::Cap_base, 549 is_virtual L4::Kip::Mem_desc, 811 L4::Ipc::Istream, 713 items cxx::List, 792 Iterator cxx::Avl_tree, 477 kd_display Kernel Debugger, 182, 183 Kept_ds L4Re::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd_display, 182, 183 kdem allocator utility, 88 l4re_util_kumem_alloc, 88 kumem_alloc, 118 kumem_alloc Kumem utilitites, 118 cxx::Bits::Direction, 592 L4, 442 kobject_typeid, 445 L4, 442 kobject_typeid, 445 L4::Cap_base Invalid, 547 No_init, 547 L4::Ipc_svr Reply_compound, 446 Reply_separate, 446 Reply_separate, 446 L4_BASE_FACTORY_CAP Capabilities, 287 L4_BASE_ICU_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_SCHEDULER_CAP		
L4vcpu::Vcpu, 944 is_static L4Re::Util::Dataspace_svr, 578 is_valid L4::Cap_base, 549 is_virtual L4::Kip::Mem_desc, 811 Istream L4::lpc::Istream, 713 items cxx::List, 792 Iterator cxx::Avl_tree, 477 kd_display Kernel Debugger, 182, 183 Kept_ds L4::Rep::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd_display, 182, 183 kobject_typeid, 445 L4::Ipc_svr Reply_compound, 446 Reply_separate, 446 L4_BASE_ICU_CAP Capabilities, 287 Ked_display, 182, 183 enter_kdebug, 181, 182 kd_display, 182, 183 kget_das L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_SCHEDULER_CAP		
is_static L4Re::Util::Dataspace_svr, 578 is_valid L4::Cap_base, 549 is_virtual L4::Kip::Mem_desc, 811 Istream L4::Ipc::Istream, 713 items cxx::List, 792 Iterator cxx::Avl_tree, 477 kd_display Kernel Debugger, 182, 183 Kept_ds L4::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 182 enter_kdebug, 181, 182 kd_display, 182, 183 kumem_utilitities, 118 kumem_alloc Kumem utilitities, 118 kumem_alloc, 118 kumem_alloc, 118 kumem_alloc Kumem utilitities, 118 kumem_alloc, 118 kumem_alloc Kumem utilitities, 118 kumem_alloc, 118 kumem_alloc Kumem utilitities, 118 cxx::Bits::Direction, 592 L4, 442 kobject_typeid, 445 L4::Cap_base Invalid, 547 No_init, 547 L4::lpc_svr Reply_compound, 446 Reply_separate, 446 Reply_separate, 446 Reply_separate, 446 Reply_separate, 446 Reply_separate, 446 Reply_compound, 446 Reply_capound, 446 Reply_capound, 446 Reply_separate, 446 L4_BASE_FACTORY_CAP Capabilities, 287 kd_display, 182, 183 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_SCHEDULER_CAP		
L4Re::Util::Dataspace_svr, 578 is_valid L4::Cap_base, 549 is_virtual L4::Kip::Mem_desc, 811 Istream L4::Ipc::Istream, 713 items cxx::List, 792 Iterator cxx::Avl_tree, 477 kd_display Kernel Debugger, 182, 183 Kept_ds L4::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd_display, 182, 183 kumem_alloc, 118 kumem_alloc, 181 cxx::Bits::Direction, 592 L4, 442 kobject_typeid, 445 L4::Cap_base Invalid, 547 No_init, 547 L4::lpc_svr Reply_compound, 446 Reply_separate, 446 Reply_separate, 446 Reply_separate, 446 Reply_separate, 287 L4_BASE_ICU_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_SCHEDULER_CAP	·	
is_valid L4::Cap_base, 549 Kumem utilitities, 118 is_virtual L4::Kip::Mem_desc, 811 Istream L4::lpc::Istream, 713 items cxx::List, 792 Iterator cxx::Avl_tree, 477 kd_display Kernel Debugger, 182, 183 Kept_ds L4::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd_display, 182, 183 kobject_typeid, 445 L4::Cap_base Invalid, 547 No_init, 547 L4::Ipc_svr Reply_compound, 446 Reply_separate, 446 Reply_separate, 446 L4_BASE_FACTORY_CAP Capabilities, 287 L4_BASE_ICU_CAP capabilities, 287 L4_BASE_LOG_CAP capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_PAGER_CAP L4_BASE_PAGER_CAP L4_debugger_global_id, 185 L4_BASE_SCHEDULER_CAP		
L4::Cap_base, 549 is_virtual L4::Kip::Mem_desc, 811 Istream L4::Ipc::Istream, 713 items cxx::List, 792 Iterator cxx::Avl_tree, 477 kd_display Kernel Debugger, 182, 183 Kept_ds L4::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd_display, 182, 183 kd_display, 182, 183 kept_ds capabilities, 287 kd_display, 182, 183 capabilities, 287 capa	L4Re::Util::Dataspace_svr, 578	
is_virtual L4::Kip::Mem_desc, 811 Istream L4::Ipc::Istream, 713 items cxx::List, 792 Iterator cxx::Avl_tree, 477 kd_display Kernel Debugger, 182, 183 Kept_ds L4::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd_display, 182, 183 kopt_ds l4_BASE_LOG_CAP capabilities, 287 kd_display, 182, 183 kept_ds l4_BASE_PAGER_CAP capabilities, 287 kd_display, 182, 183 capabilities, 287 kd_display, 182, 183 capabilities, 287 kd_display, 182, 183 capabilities, 287 kd_debugger_global_id, 185 l4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_COL_CAP Capabilities, 287 L4_BASE_CAP Capabilities, 287 L4_BASE_SCHEDULER_CAP	-	
L4::Kip::Mem_desc, 811 Istream	L4::Cap_base, 549	Kumem utiltities, 118
Istream L4::lpc::lstream, 713 items cxx::List, 792 Iterator cxx::Avl_tree, 477 kd_display Kernel Debugger, 182, 183 Kept_ds L4:Re::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd_display, 182, 183 kept_ds l4_BASE_LOG_CAP capabilities, 287 kd_display, 182, 183 kept_ds L4_BASE_LOG_CAP capabilities, 287 L4_BASE_PAGER_CAP capabilities, 287 L4_BASE_PAGER_CAP capabilities, 287 L4_BASE_PAGER_CAP capabilities, 287 L4_BASE_SCHEDULER_CAP	_virtual	
items kobject_typeid, 445 items kobject_typeid, 445 cxx::List, 792 L4::Cap_base Invalid, 547 cxx::Avl_tree, 477 No_init, 547 kd_display Reply_compound, 446 Kernel Debugger, 182, 183 Reply_separate, 446 Kept_ds L4_BASE_FACTORY_CAP L4Re::Rm, 857 Capabilities, 287 Kernel Debugger, 180 L4_BASE_ICU_CAP asm_enter_kdebug, 181, 183 Capabilities, 287 kd_display, 182, 183 L4_BASE_LOG_CAP kd_display, 182, 183 L4_BASE_LOG_CAP kd_debugger_global_id, 185 l4_debugger_kobj_to_id, 185 L4_BASE_SCHEDULER_CAP	L4::Kip::Mem_desc, 811	
items cxx::List, 792 Iterator cxx::Avl_tree, 477 kd_display Kernel Debugger, 182, 183 Kept_ds L4::Cap_base Invalid, 547 No_init, 547 L4::lpc_svr Reply_compound, 446 Reply_separate, 446 Reply_separate, 446 Kept_ds L4_BASE_FACTORY_CAP L4Re::Rm, 857 Capabilities, 287 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 capabilities, 287 enter_kdebug, 181, 182 kd_display, 182, 183 ko, 182, 183 l4_BASE_LOG_CAP Capabilities, 287 L4_BASE_PAGER_CAP L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_SCHEDULER_CAP	ream	
cxx::List, 792 Iterator	L4::lpc::lstream, 713	
Iterator CXX::Avl_tree, 477 Kd_display Kernel Debugger, 182, 183 Kept_ds L4Re::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd_display, 182, 183 ko, 182, 183 l4_debugger_global_id, 185 lt4_BASE_ICU_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_PAGER_CAP L4_BASE_PAGER_CAP L4_BASE_SCHEDULER_CAP	ms	
cxx::Avl_tree, 477 kd_display Kernel Debugger, 182, 183 Kept_ds L4Re::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd_display, 182, 183 ko, 182, 183 l4_debugger_global_id, 185 l4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_PAGER_CAP L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_SCHEDULER_CAP	cxx::List, 792	• —
kd_display Reply_compound, 446 Kernel Debugger, 182, 183 Reply_separate, 446 Kept_ds L4:Re:Rm, 857 Capabilities, 287 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd_display, 182, 183 ko, 182, 183 l4_debugger_global_id, 185 l4:Reply_compound, 446 Reply_separate, 446 L4_BASE_FACTORY_CAP Capabilities, 287 L4_BASE_ICU_CAP Capabilities, 287 L4_BASE_LOG_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_PAGER_CAP L4_BASE_SCHEDULER_CAP	rator	
kd_display	cxx::Avl_tree, 477	- ·
Kernel Debugger, 182, 183 Reply_separate, 446 Kept_ds L4Re::Rm, 857 Capabilities, 287 Kernel Debugger, 180 L4_BASE_ICU_CAP asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 L4_BASE_LOG_CAP kd_display, 182, 183 L4_BASE_LOG_CAP Capabilities, 287 ko, 182, 183 L4_BASE_PAGER_CAP l4_debugger_global_id, 185 L4_BASE_SCHEDULER_CAP		L4::lpc_svr
Kept_dsL4_BASE_FACTORY_CAPL4Re::Rm, 857Capabilities, 287Kernel Debugger, 180L4_BASE_ICU_CAPasm_enter_kdebug, 181, 183Capabilities, 287enter_kdebug, 181, 182L4_BASE_LOG_CAPkd_display, 182, 183Capabilities, 287ko, 182, 183L4_BASE_PAGER_CAPI4_debugger_global_id, 185Capabilities, 287I4_debugger_kobj_to_id, 185L4_BASE_SCHEDULER_CAP	_display	Reply_compound, 446
L4Re::Rm, 857 Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd_display, 182, 183 ko, 182, 183 l4_BASE_PAGER_CAP l4_debugger_global_id, 185 l4_debugger_kobj_to_id, 185 L4_BASE_SCHEDULER_CAP	Kernel Debugger, 182, 183	Reply_separate, 446
Kernel Debugger, 180 asm_enter_kdebug, 181, 183 enter_kdebug, 181, 182 kd_display, 182, 183 ko, 182, 183 l4_BASE_LOG_CAP Capabilities, 287 ko, 182, 183 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_BASE_SCHEDULER_CAP	pt_ds	L4_BASE_FACTORY_CAP
asm_enter_kdebug, 181, 183 Capabilities, 287 enter_kdebug, 181, 182 L4_BASE_LOG_CAP kd_display, 182, 183 Capabilities, 287 ko, 182, 183 L4_BASE_PAGER_CAP l4_debugger_global_id, 185 Capabilities, 287 L4_BASE_SCHEDULER_CAP	L4Re::Rm, 857	Capabilities, 287
enter_kdebug, 181, 182 kd_display, 182, 183 ko, 182, 183 l4_BASE_LOG_CAP Capabilities, 287 L4_BASE_PAGER_CAP Capabilities, 287 L4_debugger_global_id, 185 Capabilities, 287 L4_BASE_SCHEDULER_CAP	rnel Debugger, 180	L4_BASE_ICU_CAP
kd_display, 182, 183 Capabilities, 287 ko, 182, 183 L4_BASE_PAGER_CAP l4_debugger_global_id, 185 Capabilities, 287 l4_debugger_kobj_to_id, 185 L4_BASE_SCHEDULER_CAP	asm_enter_kdebug, 181, 183	Capabilities, 287
ko, 182, 183 L4_BASE_PAGER_CAP l4_debugger_global_id, 185 Capabilities, 287 l4_debugger_kobj_to_id, 185 L4_BASE_SCHEDULER_CAP	enter_kdebug, 181, 182	L4_BASE_LOG_CAP
I4_debugger_global_id, 185Capabilities, 287I4_debugger_kobj_to_id, 185L4_BASE_SCHEDULER_CAP	kd_display, 182, 183	Capabilities, 287
I4_debugger_kobj_to_id, 185	ko, 182, 183	L4_BASE_PAGER_CAP
I4_debugger_kobj_to_id, 185		Capabilities, 287
		L4_BASE_SCHEDULER_CAP
I4_debugger_set_object_name, 184 Capabilities, 287	I4_debugger_set_object_name, 184	Capabilities, 287
l4kd_inchar, 188 L4_BASE_TASK_CAP	l4kd inchar, 188	L4 BASE TASK CAP
outchar, 186 Capabilities, 287		
outdec, 187 L4_BASE_THREAD_CAP	outdec, 187	•
outhex12, 187 Capabilities, 287		
outhex16, 187 L4_BDR_IO_SHIFT		•
		Buffer Registers (BRs), 296
outhex32, 187 L4_BDR_MEM_SHIFT		-
		Buffer Registers (BRs), 296
outnstring, 186 L4_BDR_OBJ_SHIFT		
-		Buffer Registers (BRs), 296
Kernel Interface Page, 232 L4_CAP_FPAGE_R	-	• • • • • • • • • • • • • • • • • • • •
I4_kernel_info_version_offset, 233 Flex pages, 143	-	
I4_kip_clock, 234 L4_CAP_FPAGE_RO		
I4_kip_clock_lw, 234 Flex pages, 143	_ • _	
I4_kip_version, 233 L4_CAP_FPAGE_RW	_ ·	
14_kip_version_string, 233 Flex pages, 143	_ • _	
Kernel Interface Page API, 358 L4_CAP_MASK	_ · -	
l4_CAF_MASK l4util_kip_for_each_feature, 358 Capabilities, 286	-	
		•
_ ,	_ •	•
	_ •	
		•
Kernel Objects, 230 L4_EACCESS	mei Objecis, 200	L4_LACCESS

Error codes, 189	Flex pages, 142
L4_EADDRNOTAVAIL	L4_FPAGE_RO
Error codes, 190	Flex pages, 143
L4_EAGAIN	L4 FPAGE RW
Error codes, 189	Flex pages, 143
L4_EBADPROTO	L4_FPAGE_SIZE_BITS
Error codes, 190	Flex pages, 143
L4 EBUSY	L4_FPAGE_SIZE_SHIFT
Error codes, 189	Flex pages, 143
L4 EEXIST	L4_FPAGE_TYPE_BITS
-	Flex pages, 143
Error codes, 189 L4 EINVAL	
_	L4_FPAGE_TYPE_SHIFT
Error codes, 189	Flex pages, 142
L4_EIO	L4_FPAGE_UNCACHEABLE
Error codes, 189	Flex pages, 143
L4_EIPC_HI	L4_ICU_FLAG_MSI
Error codes, 190	Interrupt controller, 199
L4_EIPC_LO	L4_INVALID_ADDR
Error codes, 190	Memory related, 177
L4_ENAMETOOLONG	L4_INVALID_CAP
Error codes, 189	Capabilities, 286
L4 ENODEV	L4 IOPORT MAX
Error codes, 189	Flex pages, 144
L4 ENOENT	L4_IPC_ENOT_EXISTENT
Error codes, 189	Error Handling, 219
L4 ENOMEM	L4_IPC_ERROR_MASK
_	Error Handling, 219
Error codes, 189	
L4_ENOREPLY	L4_IPC_GATE_BIND_OP
Error codes, 190	IPC-Gate API, 127
L4_ENOSYS	L4_IPC_GATE_GET_INFO_OP
Error codes, 190	IPC-Gate API, 127
L4_EOK	L4_IPC_REABORTED
Error codes, 189	Error Handling, 219
L4 EPERM	L4_IPC_RECANCELED
C4_C1 C1 ((v)	
Error codes, 189	Error Handling, 219
_	
Error codes, 189	Error Handling, 219
Error codes, 189 L4_ERANGE	Error Handling, 219 L4_IPC_REMAPFAILED
Error codes, 189 L4_ERANGE Error codes, 189	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FPAGE_ADDR_BITS	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FP_OTHER_SPACES Flex pages, 143	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED Error Handling, 219
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FPAGE_ADDR_BITS Flex pages, 143 L4_FPAGE_ADDR_SHIFT	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED Error Handling, 219 L4_IPC_SECANCELED
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FP_OTHER_SPACES Flex pages, 143 L4_FPAGE_ADDR_SHIFT Flex pages, 143	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED Error Handling, 219 L4_IPC_SECANCELED Error Handling, 219
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FPAGE_ADDR_BITS Flex pages, 143 L4_FPAGE_BUFFERABLE	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED Error Handling, 219 L4_IPC_SECANCELED Error Handling, 219 L4_IPC_SECANCELED Error Handling, 219 L4_IPC_SEMAPFAILED
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FPAGE_ADDR_BITS Flex pages, 143 L4_FPAGE_BUFFERABLE Flex pages, 143	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED Error Handling, 219 L4_IPC_SECANCELED Error Handling, 219 L4_IPC_SEMAPFAILED Error Handling, 219
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FPAGE_ADDR_BITS Flex pages, 143 L4_FPAGE_BUFFERABLE Flex pages, 143 L4_FPAGE_CACHE_OPT	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED Error Handling, 219 L4_IPC_SECANCELED Error Handling, 219 L4_IPC_SEMAPFAILED Error Handling, 219 L4_IPC_SEMAPFAILED Error Handling, 219 L4_IPC_SEMSGCUT
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FPAGE_ADDR_BITS Flex pages, 143 L4_FPAGE_BUFFERABLE Flex pages, 143 L4_FPAGE_CACHE_OPT Flex pages, 143	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED Error Handling, 219 L4_IPC_SECANCELED Error Handling, 219 L4_IPC_SEMAPFAILED Error Handling, 219 L4_IPC_SEMSGCUT Error Handling, 219
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FPAGE_ADDR_BITS Flex pages, 143 L4_FPAGE_BUFFERABLE Flex pages, 143 L4_FPAGE_CACHE_OPT	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED Error Handling, 219 L4_IPC_SECANCELED Error Handling, 219 L4_IPC_SEMAPFAILED Error Handling, 219 L4_IPC_SEMAPFAILED Error Handling, 219 L4_IPC_SEMSGCUT
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FPAGE_ADDR_BITS Flex pages, 143 L4_FPAGE_BUFFERABLE Flex pages, 143 L4_FPAGE_CACHE_OPT Flex pages, 143	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED Error Handling, 219 L4_IPC_SECANCELED Error Handling, 219 L4_IPC_SEMAPFAILED Error Handling, 219 L4_IPC_SEMSGCUT Error Handling, 219
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FPAGE_ADDR_BITS Flex pages, 143 L4_FPAGE_ADDR_SHIFT Flex pages, 143 L4_FPAGE_BUFFERABLE Flex pages, 143 L4_FPAGE_CACHE_OPT Flex pages, 143 L4_FPAGE_CACHEABLE	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED Error Handling, 219 L4_IPC_SECANCELED Error Handling, 219 L4_IPC_SEMAPFAILED Error Handling, 219 L4_IPC_SEMSGCUT Error Handling, 219 L4_IPC_SEMSGCUT Error Handling, 219 L4_IPC_SERCVPFTO
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FPAGE_ADDR_BITS Flex pages, 143 L4_FPAGE_ADDR_SHIFT Flex pages, 143 L4_FPAGE_BUFFERABLE Flex pages, 143 L4_FPAGE_CACHE_OPT Flex pages, 143 L4_FPAGE_CACHEABLE Flex pages, 143	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED Error Handling, 219 L4_IPC_SECANCELED Error Handling, 219 L4_IPC_SEMAPFAILED Error Handling, 219 L4_IPC_SEMSGCUT Error Handling, 219 L4_IPC_SEMSGCUT Error Handling, 219 L4_IPC_SERCVPFTO Error Handling, 219
Error codes, 189 L4_ERANGE Error codes, 189 L4_ERRNOMAX Error codes, 190 L4_FP_ALL_SPACES Task, 247 L4_FP_DELETE_OBJ Task, 247 L4_FP_OTHER_SPACES Task, 247 L4_FPAGE_ADDR_BITS Flex pages, 143 L4_FPAGE_ADDR_SHIFT Flex pages, 143 L4_FPAGE_BUFFERABLE Flex pages, 143 L4_FPAGE_CACHE_OPT Flex pages, 143 L4_FPAGE_CACHEABLE Flex pages, 143 L4_FPAGE_CACHEABLE Flex pages, 143 L4_FPAGE_CACHEABLE Flex pages, 143 L4_FPAGE_CACHEABLE Flex pages, 143 L4_FPAGE_RIGHTS_BITS	Error Handling, 219 L4_IPC_REMAPFAILED Error Handling, 219 L4_IPC_REMSGCUT Error Handling, 219 L4_IPC_RERCVPFTO Error Handling, 219 L4_IPC_RESNDPFTO Error Handling, 219 L4_IPC_RETIMEOUT Error Handling, 219 L4_IPC_SEABORTED Error Handling, 219 L4_IPC_SECANCELED Error Handling, 219 L4_IPC_SEMAPFAILED Error Handling, 219 L4_IPC_SEMSGCUT Error Handling, 219 L4_IPC_SERCVPFTO Error Handling, 219 L4_IPC_SERCVPFTO Error Handling, 219 L4_IPC_SERCVPFTO Error Handling, 219 L4_IPC_SESNDPFTO

Error Handling, 219	Message Tag, 275
L4_IPC_SND_ERR_MASK	L4_PROTO_EXCEPTION
Error Handling, 219	Message Tag, 276
L4 IRQ F BOTH	L4 PROTO FACTORY
IRQs, 225	Message Tag, 276
L4_IRQ_F_BOTH_EDGE	L4_PROTO_IO_PAGE_FAULT
IRQs, 225	Message Tag, 276
L4_IRQ_F_CLEAR_WAKEUP	L4 PROTO IRQ
IRQs, 225	Message Tag, 275
L4_IRQ_F_EDGE	L4_PROTO_KOBJECT
IRQs, 225	Message Tag, 276
L4_IRQ_F_LEVEL	L4 PROTO LOG
IRQs, 225	Message Tag, 276
L4_IRQ_F_LEVEL_HIGH	L4_PROTO_META
IRQs, 225	Message Tag, 276
L4_IRQ_F_LEVEL_LOW	L4_PROTO_NONE
IRQs, 225	Message Tag, 275
L4_IRQ_F_MASK	L4_PROTO_PAGE_FAULT
IRQs, 225	Message Tag, 275
L4_IRQ_F_NEG	L4_PROTO_PF_EXCEPTION
IRQs, 225	Message Tag, 275
L4_IRQ_F_NEG_EDGE	L4_PROTO_PREEMPTION
IRQs, 225	Message Tag, 275
L4_IRQ_F_NONE	L4 PROTO SCHEDULER
IRQs, 225	Message Tag, 276
L4_IRQ_F_POS	L4_PROTO_SIGMA0
IRQs, 225	Message Tag, 276
L4_IRQ_F_POS_EDGE	L4_PROTO_SYS_EXCEPTION
IRQs, 225	Message Tag, 275
L4_IRQ_F_SET_WAKEUP	L4_PROTO_TASK
IRQs, 225	Message Tag, 276
L4_ITEM_CONT	L4_PROTO_THREAD
Message Items, 150	Message Tag, 276
L4_ITEM_MAP	L4_PROTO_VM
Message Items, 150	Message Tag, 276
L4_MAP_ITEM_GRANT	L4_RCV_ITEM_LOCAL_ID
Message Items, 150	Message Items, 151
L4_MAP_ITEM_MAP	L4_RCV_ITEM_SINGLE_CAP
Message Items, 150	Message Items, 150
L4_MEM_WIDTH_1BYTE	L4_SCHEDULER_IDLE_TIME_OP
Memory operations., 308	Scheduler, 242
L4_MEM_WIDTH_2BYTE	L4_SCHEDULER_INFO_OP
Memory operations., 308	Scheduler, 242
L4_MEM_WIDTH_4BYTE	L4_SCHEDULER_RUN_THREAD_OP
Memory operations., 308	Scheduler, 242
L4 MSGTAG ERROR	L4 SYSF CALL
Message Tag, 276	Object Invocation, 207
L4 MSGTAG FLAGS	L4 SYSF NONE
Message Tag, 276	Object Invocation, 207
L4 MSGTAG PROPAGATE	L4 SYSF OPEN WAIT
Message Tag, 276	Object Invocation, 207
L4_MSGTAG_SCHEDULE	L4_SYSF_RECV
Message Tag, 276	Object Invocation, 207
L4_MSGTAG_TRANSFER_FPU	L4_SYSF_REPLY
Message Tag, 276	Object Invocation, 207
L4_MSGTAG_XCPU	L4_SYSF_REPLY_AND_WAIT
Message Tag, 276	Object Invocation, 207
L4_PROTO_ALLOW_SYSCALL	L4_SYSF_SEND

Object Invocation, 207	Fiasco-UX Virtual devices, 307
L4_SYSF_SEND_AND_WAIT	L4_TYPE_VHW_NET
Object Invocation, 207	Fiasco-UX Virtual devices, 307
L4_SYSF_WAIT	L4_TYPE_VHW_NONE
Object Invocation, 207	Fiasco-UX Virtual devices, 307
L4_THREAD_AMD64_SET_SEGMENT_BASE_OP	L4_UTCB_BUF_REGS_OFFSET
Thread, 257	x86 Virtual Registers (UTCB), 314
L4_THREAD_ARM_TPIDRURO_OP	L4_UTCB_EXCEPTION_REGS_SIZE
Thread, 257	x86 Virtual Registers (UTCB), 314
L4_THREAD_CONTROL_ALIEN	L4_UTCB_GENERIC_BUFFERS_SIZ
Thread, 257	x86 Virtual Registers (UTCB), 314
L4_THREAD_CONTROL_BIND_TASK	L4_UTCB_GENERIC_DATA_SIZE
Thread, 257	x86 Virtual Registers (UTCB), 314
L4_THREAD_CONTROL_MR_IDX_BIND_TASK	L4_UTCB_INHERIT_FPU
Thread, 258	x86 Virtual Registers (UTCB), 314
L4_THREAD_CONTROL_MR_IDX_BIND_UTCB	L4_UTCB_MSG_REGS_OFFSET
Thread, 258	x86 Virtual Registers (UTCB), 314
L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER	L4_UTCB_OFFSET
Thread, 258	x86 Virtual Registers (UTCB), 314
L4_THREAD_CONTROL_MR_IDX_FLAG_VALS	L4_UTCB_THREAD_REGS_OFFSET
Thread, 258	x86 Virtual Registers (UTCB), 314
L4_THREAD_CONTROL_MR_IDX_FLAGS	L4_VCON_ECHO
Thread, 258	Virtual Console, 301
L4_THREAD_CONTROL_MR_IDX_PAGER	L4_VCON_GET_ATTR_OP
Thread, 258	Virtual Console, 302
L4_THREAD_CONTROL_OP	L4_VCON_ICANON
Thread, 257	Virtual Console, 301
L4_THREAD_CONTROL_SET_EXC_HANDLER	L4_VCON_ICRNL
Thread, 257	Virtual Console, 301
L4_THREAD_CONTROL_SET_PAGER	L4_VCON_IGNCR
Thread, 257	Virtual Console, 301
L4_THREAD_CONTROL_UX_NATIVE Thread, 257	L4_VCON_INLCR Virtual Console, 301
L4_THREAD_EX_REGS_CANCEL	L4_VCON_OCRNL
Thread, 258	Virtual Console, 301
L4_THREAD_EX_REGS_OP	L4_VCON_ONLCR
Thread, 257	Virtual Console, 301
L4_THREAD_EX_REGS_TRIGGER_EXCEPTION	L4_VCON_ONLRET
Thread, 258	Virtual Console, 301
L4_THREAD_MODIFY_SENDER_OP	L4_VCON_SET_ATTR_OP
Thread, 257	Virtual Console, 302
L4_THREAD_OPCODE_MASK	L4 VCON WRITE OP
Thread, 257	Virtual Console, 302
L4 THREAD REGISTER DELETE IRQ OP	L4 VCON WRITE SIZE
Thread, 257	Virtual Console, 301
L4_THREAD_STATS_OP	L4_VCPU_F_DEBUG_EXC
Thread, 257	vCPU API, 306
L4_THREAD_SWITCH_OP	L4 VCPU F EXCEPTIONS
Thread, 257	vCPU API, 306
L4_THREAD_VCPU_CONTROL_OP	L4_VCPU_F_FPU_ENABLED
Thread, 257	vCPU API, 306
L4_THREAD_VCPU_RESUME_OP	L4_VCPU_F_IRQ
Thread, 257	vCPU API, 306
L4_THREAD_X86_GDT_OP	L4_VCPU_F_PAGE_FAULTS
Thread, 257	vCPU API, 306
L4_TYPE_VHW_FRAMEBUFFER	L4_VCPU_F_USER_MODE
Fiasco-UX Virtual devices, 307	vCPU API, 306
L4_TYPE_VHW_INPUT	L4_VCPU_OFFSET_EXT_INFOS
_	

vCPU API, 306	Memory descriptors (C version), 238
L4_VCPU_OFFSET_EXT_STATE	I4_mem_type_shared
vCPU API, 306	Memory descriptors (C version), 238
L4_VCPU_SF_IRQ_PENDING	I4_mem_type_undefined
vCPU API, 306	Memory descriptors (C version), 238
L4_VM_VMX_BASIC_REG	L4IO_DEVICE_ANY
VM API for VMX, 163	IO interface, 383
L4_VM_VMX_CR0_FIXED0_REG	L4IO_DEVICE_INVALID
VM API for VMX, 163	IO interface, 383
•	L4IO DEVICE OTHER
L4_VM_VMX_CR0_FIXED1_REG	
VM API for VMX, 163	IO interface, 383
L4_VM_VMX_CR4_FIXED0_REG	L4IO_DEVICE_PCI
VM API for VMX, 163	IO interface, 383
L4_VM_VMX_CR4_FIXED1_REG	L4IO_DEVICE_USB
VM API for VMX, 163	IO interface, 383
L4_VM_VMX_ENTRY_CTLS_DFL1_REG	L4IO_MEM_CACHED
VM API for VMX, 164	IO interface, 383
L4_VM_VMX_EPT_VPID_CAP_REG	L4IO_MEM_EAGER_MAP
VM API for VMX, 163	IO interface, 383
L4_VM_VMX_EXIT_CTLS_DFL1_REG	L4IO_MEM_NONCACHED
VM API for VMX, 164	IO interface, 383
L4_VM_VMX_MISC_REG	L4IO_MEM_USE_MTRR
VM API for VMX, 163	IO interface, 383
L4_VM_VMX_NUM_CAPS_REGS	L4IO_MEM_USE_RESERVED_AREA
VM API for VMX, 163	IO interface, 383
L4_VM_VMX_NUM_DFL1_REGS	L4IO_RESOURCE_ANY
VM API for VMX, 164	IO interface, 383
L4_VM_VMX_PINBASED_CTLS_DFL1_REG	L4IO_RESOURCE_INVALID
VM API for VMX, 164	IO interface, 383
L4_VM_VMX_PROCBASED_CTLS2_REG	L4IO_RESOURCE_IRQ
VM API for VMX, 163	IO interface, 383
L4_VM_VMX_PROCBASED_CTLS_DFL1_REG	L4IO_RESOURCE_MEM
VM API for VMX, 164	IO interface, 383
L4_VM_VMX_TRUE_ENTRY_CTLS_REG	L4IO RESOURCE PORT
VM API for VMX, 163	IO interface, 383
L4_VM_VMX_TRUE_EXIT_CTLS_REG	L4RE_ELF_AUX_T_KIP_ADDR
VM API for VMX, 163	L4Re ELF Auxiliary Information, 100
L4_VM_VMX_TRUE_PINBASED_CTLS_REG	L4RE_ELF_AUX_T_NONE
VM API for VMX, 163	L4Re ELF Auxiliary Information, 100
L4_VM_VMX_TRUE_PROCBASED_CTLS_REG	L4RE ELF AUX T STACK ADDR
VM API for VMX, 163	L4Re ELF Auxiliary Information, 100
L4_VM_VMX_VMCS_CR2	L4RE_ELF_AUX_T_STACK_SIZE
VM API for VMX, 164	L4Re ELF Auxiliary Information, 100
L4_VM_VMX_VMCS_ENUM_REG	L4RE_ELF_AUX_T_VMA
VM API for VMX, 163	L4Re ELF Auxiliary Information, 100
L4_WHOLE_ADDRESS_SPACE	L4RE_RM_ATTACH_FLAGS
Flex pages, 143	Region map interface, 78
• =	L4RE RM EAGER MAP
L4_WHOLE_IOADDRESS_SPACE	
Flex pages, 144	Region map interface, 78
I4_mem_type_archspecific	L4RE_RM_IN_AREA
Memory descriptors (C version), 238	Region map interface, 78
I4_mem_type_bootloader	L4RE_RM_NO_ALIAS
Memory descriptors (C version), 238	Region map interface, 78
I4_mem_type_conventional	L4RE_RM_OVERMAP
Memory descriptors (C version), 238	Region map interface, 78
I4_mem_type_dedicated	L4RE_RM_PAGER
Memory descriptors (C version), 238	Region map interface, 78
I4_mem_type_reserved	L4RE_RM_READ_ONLY

Region map interface, 78	F_pointer, 667
L4RE RM REGION FLAGS	L4Re::Video::View
Region map interface, 78	F above, 949
L4RE RM RESERVED	F_dyn_allocated, 949
Region map interface, 78	F_flags_mask, 949
L4RE RM SEARCH ADDR	F_fully_dynamic, 949
Region map interface, 78	F_none, 949
L4Re ELF Auxiliary Information	F_set_background, 949
L4RE_ELF_AUX_T_KIP_ADDR, 100	F_set_buffer, 949
L4RE_ELF_AUX_T_NONE, 100	F_set_buffer_offset, 949
L4RE_ELF_AUX_T_STACK_ADDR, 100	F_set_bytes_per_line, 949
L4RE_ELF_AUX_T_STACK_SIZE, 100	F_set_flags, 949
L4RE_ELF_AUX_T_VMA, 100	F_set_pixel, 949
L4Re Protocol identifiers	F_set_position, 949
Dataspace, 114	L4SIGMA0_IPCERROR
Debug, 114	Sigma0 API, 400
Default, 114	L4SIGMA0 NOFPAGE
Event, 114	Sigma0 API, 400
Goos, 114	L4SIGMA0 NOTALIGNED
Mem_alloc, 114	Sigma0 API, 400
Namespace, 114	L4SIGMA0 OK
Parent, 114	Sigma0 API, 400
•	•
Rm, 114	L4SIGMA0_SMALLERFPAGE
L4Re::Dataspace	Sigma0 API, 400
Map_ro, 567	L4VCPU_IRQ_STATE_DISABLED
Map_rw, 567	vCPU Support Library, 406
L4Re::Mem_alloc	L4VCPU_IRQ_STATE_ENABLED
Continuous, 805	vCPU Support Library, 406
Pinned, 805	L4::Alloc_list, 449
Super_pages, 805	L4::Base_exception, 481
L4Re::Namespace	L4::Basic_registry, 490
Ro, 823	dispatch, 491
Rw, 823	Value, 491
Strong, 823	L4::Bounds error, 515
L4Re::Rm	L4::Cap
Attach_flags, 858	Cap, 539
	move, 541
Detach_again, 858	
Detach_exact, 858	L4::Cap< T >, 537
Detach_free, 858	L4::Cap_base, 545
Detach_keep, 858	_c, 553
Detach_overlap, 858	cap, 548
Detached_ds, 857	Cap_base, 548
Eager_map, 858	Cap_type, 547
In_area, 858	fpage, 550
Kept_ds, 857	is_valid, 549
Pager, 858	No_init_type, 547
Read_only, 858	snd base, 551
Region_flags, 858	validate, 552
Reserved, 858	L4::Com_error, 557
Search_addr, 858	Com error, 560
	_ · · · ·
Split_ds, 858	L4::Debugger, 581
L4Re::Util::Event_t	get_object_name, 587
Mode_irq, 638	global_id, 584
Mode_polling, 638	kobj_to_id, 585
L4Re::Video::Goos	query_log_name, 586
F_auto_refresh, 667	query_log_typeid, 585
F_dynamic_buffers, 667	set_object_name, 584
F_dynamic_views, 667	switch_log, 586

L4::Element already exists 507	nut 922 924
L4::Element_already_exists, 597 L4::Element_not_found, 600	put, 833, 834 send, 835
L4::Exception_tracer, 640	tag, 834
L4::Factory, 641	L4::lpc::Small_buf, 888
create, 644	L4::lpc_gate, 699
create_factory, 647	bind_thread, 702
create_gate, 648	get_infos, 702
create_irq, 649	L4::lpc_svr, 445
create_task, 645	Reply mode, 446
create_thread, 646	L4::lpc svr::Compound reply, 560
create vm, 650	L4::lpc svr::Default loop hooks, 587
L4::Factory::Lstr, 801	L4::lpc_svr::Default_setup_wait, 588
L4::Factory::Nil, 826	L4::lpc_svr::Default_timeout, 589
L4::Factory::S, 868	L4::lpc_svr::lgnore_errors, 682
operator I4_msgtag_t, 869	L4::lrq, 702
operator<<, 869, 871	attach, 705
S, 869	chain, 706
L4::IOModifier, 692	detach, 707
L4::lcu, 673	receive, 707
bind, 676	trigger, 709
info, 678	unmask, 709
mask, 680	wait, 708
msi_info, 678	L4::Kip::Mem_desc, 806
set_mode, 681	count, 808, 809
unbind, 677	end, 810
unmask, 681	first, 808
L4::lcu::lnfo, 688	is_virtual, 811
L4::Invalid_capability, 689	Mem_desc, 808
cap, 692	set, 811
Invalid_capability, 692	size, 810
L4::lpc::Buf_cp_in	start, 809
Buf_cp_in, 532	sub_type, 811
L4::lpc::Buf_cp_in< T >, 532	type, 811
L4::lpc::Buf_cp_out	L4::Kobject, 722
buf, 535	cap, 723
Buf_cp_out, 535	dec_refcnt, 723
size, 535	kobject_typeid, 724
L4::lpc::Buf_cp_out< T >, 534	L4::Kobject_2t kobject typeid, 726
L4::lpc::Buf_in Buf_in, 537	L4::Kobject 2t< Derived, Base1, Base2, PROTO >,
L4::lpc::Buf in< T >, 536	724
L4::lpc::lostream, 693	L4::Kobject_t
call, 697	kobject_typeid, 728
lostream, 696	L4::Kobject_t< Derived, Base, PROTO >, 726
reply_and_wait, 698	L4::Meta, 812
reset, 696	interface, 814
L4::lpc::lstream, 710	num_interfaces, 814
get, 714, 715	supports, 815
Istream, 713	L4::Out_of_memory, 835
receive, 717	L4::Runtime_error, 866
reset, 714	L4::Scheduler, 871
skip, 715	idle_time, 875
tag, 715, 716	info, 874
wait, 716, 717	is_online, 876
L4::lpc::Msg_ptr	run_thread, 875
Msg_ptr, 819	L4::Server
L4::lpc::Msg_ptr< T >, 818	internal_loop, 879
L4::lpc::Ostream, 830	Server, 879

L4::Server< LOOP_HOOKS >, 877	Memory related, 176
L4::Server_object, 880	L4_NOTHROW
dispatch, 880	Basic Macros, 130
L4::Smart_cap	L4_PAGEMASK
Smart cap, 891	Memory related, 176
L4::Smart_cap< T, SMART >, 889	L4 SUPERPAGEMASK
L4::String, 898	Memory related, 176
L4::Task, 901	L4 SUPERPAGESIZE
add ku mem, 909	Memory related, 176
cap_equal, 908	I4_addr_consts_t
cap_has_child, 908	Memory related, 177
cap_valid, 907	I4_buf_regs_t, 728
delete_obj, 906	I4_buffer_desc_consts_t
_ •	
map, 903	Buffer Registers (BRs), 296
release_cap, 906	I4_busy_wait_ns
unmap, 904	Timestamp Counter, 323
unmap_batch, 905	I4_busy_wait_us
L4::Thread, 910	Timestamp Counter, 324
control, 915	l4_cache_clean_data
ex_regs, 913, 914	Cache Consistency, 173
modify_senders, 918	I4_cache_coherent
register_del_irq, 918	Cache Consistency, 174
stats_time, 916	<pre>I4_cache_dma_coherent</pre>
switch_to, 915	Cache Consistency, 174
vcpu_control, 917	I4_cache_flush_data
vcpu_control_ext, 917	Cache Consistency, 173
vcpu_resume_commit, 916	I4_cache_inv_data
vcpu_resume_start, 916	Cache Consistency, 174
L4::Thread::Attr, 449	I4_calibrate_tsc
Attr, 451	Timestamp Counter, 324
bind, 453	I4_cap_consts_t
exc handler, 452	Capabilities, 286
pager, 451	L4_cap_fpage_rights
ux_host_syscall, 453	Flex pages, 143
L4::Thread::Modify_senders, 817	I4_cap_idx_t
add, 818	Capabilities, 286
L4::Type_info, 920	I4_capability_equal
L4::Unknown_error, 921	Capabilities, 290
L4::Vcon, 928	l4_debugger_global_id
get_attr, 934	Kernel Debugger, 185
read, 933	I4_debugger_kobj_to_id
send, 931	Kernel Debugger, 185
set_attr, 934	I4_debugger_set_object_name
write, 931	Kernel Debugger, 184
L4::Vm, 952	I4_default_caps_t
L4_DISABLE_COPY	Capabilities, 286
Capabilities, 285	I4_error
L4_EXPORT	Error Handling, 220
Basic Macros, 130	I4_error_code_t
L4_HIDDEN	Error codes, 189
Basic Macros, 131	I4_exc_regs_t, 729
L4_IPC_TIMEOUT_0	flags, 731
Timeouts, 154	I4_factory_create_factory
L4 KOBJECT	Factory, 194
Capabilities, 286	l4_factory_create_gate
L4 LOG2 PAGESIZE	Factory, 194
Memory related, 176	I4_factory_create_irq
L4 LOG2 SUPERPAGESIZE	Factory, 195
	. 45.6.7, 100

I4_factory_create_task	l4_int64_t
Factory, 192	Integer Types, 439
I4_factory_create_thread	I4_int8_t
Factory, 193	Integer Types, 438
I4_factory_create_vm	I4_iofpage
Factory, 196	Flex pages, 144
I4 fpage	14_ipc
Flex pages, 144	Object Invocation, 214
l4_fpage_all	I4_ipc_call
Flex pages, 144	Object Invocation, 210
l4_fpage_cacheability_opt_t	I4_ipc_error
Flex pages, 143	Error Handling, 219
•	_
I4_fpage_consts	I4_ipc_error_code
Flex pages, 142	Error Handling, 222
I4_fpage_contains	I4_ipc_gate_bind_thread
Flex pages, 148	IPC-Gate API, 127
I4_fpage_invalid	I4_ipc_gate_get_infos
Flex pages, 144	IPC-Gate API, 127
I4_fpage_max_order	L4_ipc_gate_ops
Flex pages, 148	IPC-Gate API, 127
I4_fpage_page	<pre>I4_ipc_is_rcv_error</pre>
Flex pages, 147	Error Handling, 222
L4_fpage_rights	<pre>I4_ipc_is_snd_error</pre>
Flex pages, 143	Error Handling, 221
I4_fpage_rights	I4_ipc_receive
Flex pages, 146	Object Invocation, 209
I4_fpage_set_rights	I4_ipc_reply_and_wait
Flex pages, 147	Object Invocation, 211
I4_fpage_size	I4_ipc_send
Flex pages, 147	Object Invocation, 207
14_fpage_t, 731	I4_ipc_send_and_wait
	Object Invocation, 212
I4_fpage_type	-
Flex pages, 146	I4_ipc_sleep
I4_get_hz	Object Invocation, 215
Timestamp Counter, 325	I4_ipc_tcr_error_t
I4_icu_bind	Error Handling, 218
Interrupt controller, 199	I4_ipc_timeout
L4_icu_flags	Timeouts, 155
Interrupt controller, 199	I4_ipc_wait
I4_icu_info	Object Invocation, 208
Interrupt controller, 201	l4_irq_attach
I4_icu_info_t, 732	IRQs, 225
features, 734	l4_irq_chain
Interrupt controller, 199	IRQs, 226
I4_icu_mask	I4_irq_detach
Interrupt controller, 202	IRQs, 226
I4_icu_msi_info	L4_irq_mode
Interrupt controller, 201	IRQs, 225
I4_icu_set_mode	I4_irq_receive
Interrupt controller, 200	IRQs, 228
I4_icu_unbind	I4_irq_trigger
Interrupt controller, 200	IRQs, 227
I4_icu_unmask	I4_irq_unmask
	IRQs, 229
Interrupt controller, 202	
l4_int16_t	I4_irq_wait
Integer Types, 438	IRQs, 229
I4_int32_t	I4_is_fpage_writable
Integer Types, 439	Flex pages, 145

I4_is_invalid_cap	I4_msgtag_is_preemption
Capabilities, 289	Message Tag, 280
l4_is_valid_cap	I4_msgtag_is_sigma0
Capabilities, 289	Message Tag, 282
I4_kernel_info_get_mem_desc_end	I4_msgtag_is_sys_exception
Memory descriptors (C version), 239	Message Tag, 281
I4_kernel_info_get_mem_desc_is_virtual	I4_msgtag_items
Memory descriptors (C version), 239	Message Tag, 279
I4_kernel_info_get_mem_desc_start	I4_msgtag_label
Memory descriptors (C version), 239	Message Tag, 277
I4_kernel_info_get_mem_desc_subtype	l4_msgtag_protocol
Memory descriptors (C version), 239	Message Tag, 275
I4_kernel_info_get_mem_desc_type	I4_msgtag_t, 738
Memory descriptors (C version), 239	flags, 739
I4_kernel_info_get_num_mem_descs	Message Tag, 275
Memory descriptors (C version), 238	I4_msgtag_words
I4_kernel_info_mem_desc_t, 734	Message Tag, 278
Memory descriptors (C version), 238	I4_ns_to_tsc
I4_kernel_info_set_mem_desc	Timestamp Counter, 323
Memory descriptors (C version), 238	l4_obj_fpage
I4_kernel_info_t, 735	Flex pages, 145
14 kernel info version offset	I4_rcv_timeout
Kernel Interface Page, 233	Timeouts, 157
I4_kip_clock	I4_rdpmc
Kernel Interface Page, 234	Timestamp Counter, 320
I4_kip_clock_lw	I4_rdpmc_32
Kernel Interface Page, 234	Timestamp Counter, 322
I4_kip_version	I4_rdtsc
Kernel Interface Page, 233	Timestamp Counter, 320
I4_kip_version_string	I4_rdtsc_32
Kernel Interface Page, 233	Timestamp Counter, 320
I4_map_control	l4_round_page
Message Items, 151	Memory related, 178
I4_map_obj_control	I4_round_size
Message Items, 151	Memory related, 178
L4_mem_op_widths	I4_sched_cpu_set
Memory operations., 308	Scheduler, 242
I4_mem_read	I4_sched_cpu_set_t, 740
Memory operations., 309	I4_sched_param_t, 740
I4_mem_type_t	<pre>I4_scheduler_idle_time</pre>
Memory descriptors (C version), 238	Scheduler, 244
I4_mem_write	I4_scheduler_info
Memory operations., 310	Scheduler, 242
I4_msg_item_consts_t	I4_scheduler_is_online
Message Items, 150	Scheduler, 244
I4_msg_regs_t, 737	L4_scheduler_ops
I4_msgtag	Scheduler, 242
Message Tag, 276	I4_scheduler_run_thread
I4_msgtag_flags	Scheduler, 243
Message Tag, 276, 279	I4_snd_fpage_t, 741
I4_msgtag_has_error	I4_snd_timeout
Message Tag, 279	Timeouts, 155
I4_msgtag_is_exception	l4_sndfpage_add
Message Tag, 281	Object Invocation, 216
I4_msgtag_is_io_page_fault	I4_syscall_flags_t
Message Tag, 282	Object Invocation, 206
I4_msgtag_is_page_fault	I4_task_add_ku_mem
Message Tag, 280	Task, 252
	idon, Lot

I4_task_cap_equal	Thread, 262
Task, 252	<pre>I4_thread_vcpu_control_ext</pre>
I4_task_cap_has_child	Thread, 263
Task, 251	<pre>I4_thread_vcpu_resume_commit</pre>
I4_task_cap_valid	Thread, 261
Task, 251	<pre>I4_thread_vcpu_resume_start</pre>
I4_task_delete_obj	Thread, 261
Task, 250	I4_thread_yield
l4_task_map	Thread, 259
Task, 247	I4 timeout
14 task release cap	Timeouts, 155
Task, 250	I4_timeout_abs
I4_task_unmap	Timeouts, 158
Task, 248	I4_timeout_abs_validity
I4_task_unmap_batch	Timeouts, 155
Task, 249	l4 timeout get
I4_thread_arm_set_tpidruro	Timeouts, 158
Thread, 266	l4 timeout is absolute
14 thread control alien	Timeouts, 157
Thread control, 271	I4_timeout_rel
I4_thread_control_bind	Timeouts, 155
Thread_control, 269	14 timeout rel get
I4_thread_control_commit	Timeouts, 157
Thread control, 272	I4_timeout_s, 744
I4_thread_control_exc_handler	Timeouts, 154
Thread control, 269	I4_timeout_t, 744
L4_thread_control_flags	Timeouts, 154
Thread, 257	I4_tracebuffer_status_t, 745
L4_thread_control_mr_indices	cnt_iobmap_tlb_flush, 748
Thread, 257	size0, 748
I4_thread_control_pager	size1, 748
Thread control, 268	tracebuffer0, 748
I4_thread_control_start	tracebuffer1, 748
Thread control, 268	version0, 748
I4_thread_control_ux_host_syscall	version1, 748
Thread control, 272	I4_tracebuffer_status_window_t, 749
l4_thread_ex_regs	I4_trunc_page
Thread, 258	Memory related, 177
L4_thread_ex_regs_flags	I4_trunc_size
Thread, 258	Memory related, 177
I4_thread_ex_regs_ret	I4_tsc_init
Thread, 259	Timestamp Counter, 325
I4_thread_modify_sender_add	I4_tsc_to_ns
Thread, 264	Timestamp Counter, 322
I4_thread_modify_sender_commit	I4_tsc_to_s_and_ns
Thread, 265	Timestamp Counter, 322
I4_thread_modify_sender_start	I4_tsc_to_us
Thread, 264	Timestamp Counter, 322
L4_thread_ops	I4_uint16_t
Thread, 257	Integer Types, 438
l4_thread_register_del_irq	I4_uint32_t
Thread, 263	Integer Types, 439
I4_thread_regs_t, 742	I4 uint64 t
I4_thread_stats_time	Integer Types, 439
Thread, 260	I4 uint8 t
I4_thread_switch	Integer Types, 438
Thread, 260	I4_unmap_flags_t
I4_thread_vcpu_control	Task, 247
mp	, —

I4_utcb_br	count, 759
Virtual Registers (UTCBs), 293	descs, 759
L4_utcb_consts_x86	magic, 759
x86 Virtual Registers (UTCB), 314	version, 759
I4_utcb_exc	I4_vhw_entry, 759
Exception registers, 298	fd, 761
I4_utcb_exc_is_pf	irq_no, 761
Exception registers, 299	mem_size, 761
I4_utcb_exc_pc	mem_start, 761
Exception registers, 299	provider_pid, 761
I4_utcb_exc_pc_set	type, 760
Exception registers, 299	I4_vhw_entry_type
I4_utcb_mr	Fiasco-UX Virtual devices, 307
Virtual Registers (UTCBs), 292	<pre>l4_vm_svm_vmcb_control_area, 762</pre>
I4_utcb_mr64_idx	I4_vm_svm_vmcb_state_save_area, 762
Timeouts, 160	I4_vm_svm_vmcb_state_save_area_seg, 764
I4_utcb_t	I4_vm_svm_vmcb_t, 764
Virtual Registers (UTCBs), 292	I4_vm_tz_state, 766
I4_utcb_tcr	L4_vm_vmx_caps_regs
Virtual Registers (UTCBs), 293	VM API for VMX, 163
I4_vcon_attr_t, 750	I4_vm_vmx_clear
I4_vcon_get_attr	VM API for VMX, 166
Virtual Console, 304	L4_vm_vmx_dfl1_regs
L4_vcon_i_flags	VM API for VMX, 163
Virtual Console, 301	I4_vm_vmx_field_len
L4_vcon_I_flags	VM API for VMX, 165
Virtual Console, 301	I4_vm_vmx_field_order
L4_vcon_o_flags	VM API for VMX, 165
Virtual Console, 301	I4_vm_vmx_get_caps
L4_vcon_ops	VM API for VMX, 164
Virtual Console, 302	I4_vm_vmx_get_caps_default1
I4_vcon_read	VM API for VMX, 164
Virtual Console, 303	I4_vm_vmx_get_cr2_index
I4_vcon_send	VM API for VMX, 167
Virtual Console, 302	I4_vm_vmx_ptr_load
I4_vcon_set_attr	VM API for VMX, 166
Virtual Console, 303	L4_vm_vmx_read
I4_vcon_write	VM API for VMX, 169
Virtual Console, 302	I4_vm_vmx_read_16
L4_vcon_write_consts	VM API for VMX, 168
Virtual Console, 301	I4_vm_vmx_read_32
I4_vcpu_ipc_regs_t, 750	VM API for VMX, 168
I4_vcpu_regs_t, 751	I4_vm_vmx_read_64
ax, 754	VM API for VMX, 169
bp, 753	I4_vm_vmx_read_nat
bx, 754	VM API for VMX, 167
cx, 754	I4_vm_vmx_write
di, 753	VM API for VMX, 172
dx, 754	I4_vm_vmx_write_16
si, 753	VM API for VMX, 171
L4_vcpu_state_flags	I4_vm_vmx_write_32
vCPU API, 306	VM API for VMX, 171
L4_vcpu_state_offset	I4_vm_vmx_write_64
vCPU API, 306	VM API for VMX, 171
I4_vcpu_state_t, 754	I4_vm_vmx_write_nat
L4_vcpu_sticky_flags	VM API for VMX, 170
vCPU API, 306	L4RE_ELF_AUX_ELEM
I4_vhw_descriptor, 757	L4Re ELF Auxiliary Information, 100

Liffa C Interface, 120 Liffa C I Interface, 55 Liffa C I Interface, 120 Liffa C I Interface, 122 Liffa C I I I Interface, 122 Liffa C I I I I Interface, 122 Liffa C I I I I I I I I I I I I I I I I I I	L4Re, 446	free, 806
L4Re Capability API, 116 cap_alloc, 116 L4Re LEF Auxiliary information, 99 L4Re Protocolic distrifiers, 113 Protocols, 113 L4Re bit Centerface, 122 L4Re bit C++ Interface, 57 L4Re:Cap_alloc, 541 alloc, 542 free, 543 get cap_alloc, 543 L4Re:Dataspace, 584 allocate, 570 clar, 589 copy_in, 570 flags, 572 lags, 573 map_568 Map_flags, 567 map_region, 568 phys, 571 size, 572 L4Re:Cap_alloc, 518, 621 first_free_utch, 619, 622 get, 619 get		,
LARE Capability API, 116		•
Cap_alloc, 116		•
LARE LEF Auxiliary Information, 99 LARe Protocol identifiers, 113	• •	• •
LARe Protocol identifiers, 113	• —	
Protocols, 113	-	
L4Re Util C Interface, 122 L4Re::Rm, 855 L4Re::Cap_alloc, 541 alloc, 542 free, 543 get_cap_alloc, 543 L4Re::Cap_alloc, 543 L4Re::Cap_alloc, 543 L4Re::Cap_alloc, 543 L4Re::Cap_alloc, 5443 L4Re::Dataspace, 564 allocate, 570 clear, 589 copy_in, 570 flags, 572 lags, 572 lags, 573 map_568 Map_flags, 567 map_region, 568 phys, 571 size, 572 L4Re::Dataspace:Stats, 897 L4Re::Debug_obj, 579 debug, 581 L4Re::Dataspace:Stats, 897 L4Re::Debug_obj, 579 debug, 581 L4Re::Evn, 615 aro, 617 flactory, 618, 621 first_free_ucb, 619, 622 get_differ_d		
L4Re: Lull C++ Interface, 57 L4Re::Cap_ alloc, 541		
L4Re:Cap_alloc, 541		
alloc, 542		
free, 543 L4Re::Console, 561 L4Re::Dataspace, 564 allocate, 570 clear, 569 copy in, 570 clear, 569 copy in, 573 map, 568 Map_flags, 572 info, 573 map_flags, 573 map_flags, 573 map_region, 568 phys, 571 size, 572 L4Re::Dataspace; Stalts, 897 L4Re::Dataspace; Stalts, 897 L4Re::Dataspace; Stalts, 897 L4Re::Debug_obj, 579 debug_, 581 L4Re::Lill::Cap_alloc_base, 544 L4Re::Lill::Cap_alloc_base, 544 L4Re::Lill::Cap_alloc_base, 544 L4Re::Debug_obj, 579 debug_, 581 L4Re::Debug_obj, 579 debug_, 581 L4Re::Lill::Cap_alloc_base, 578 clear, 577 relase, 577 relase, 577 relase, 577 relase, 577 relase, 577 relase, 577 take, 577 relase, 578 clear,	• —	_ • · · · · · · · · · · · · · · · · · ·
get_cap_alloc, 543 L4Re::Console, 561 L4Re::Dataspace, 564 allocate, 570 clear, 569 copy_in, 570 flags, 572 info, 573 map, 568 Map_flags, 567 map_region, 568 phys, 571 size, 572 L4Re::Debug_obj, 579 debug_obj, 579 debug_obj, 579 debug_obj, 681 L4Re::Env, 615 env, 617 flactory, 618, 621 first_free_ucb, 619, 622 get_cap, 620 initial_caps, 619, 622 log, 618, 621 mem_alloc, 617, 621 parent, 617, 620 rm, 617, 620 rm, 617, 620 get_buffer, 628 L4Re::Event_buffer_t, 633 next, 634 L4Re::Event_buffer_t, 633 next, 634 L4Re::Event_buffer_t, 633 next, 634 L4Re::Event_buffer_t > PAYLOAD >, 631 L4Re::Debug_bufes, 798 print, 801 print, 801 print, 801 L4Re::Debug_bufe, 622 L4Re::Debug_obj, 579 debug_obj, 579 debug_obj, 579 debug_obj, 579 debug_obj, 579 debug_obj, 579 clear, 578 clear, 578 clear, 578 clear, 578 clear, 578 clear, 578 map_fook, 576 map_hook, 5		
L4Re::Dataspace, 564 L4Re::Dataspace, 564 allocate, 570 clear, 569 copy_in, 570 flags, 572 inlo, 573 map, 568 Map_flags, 567 map_region, 568 phys, 571 size, 572 L4Re::Dataspace::Stats, 897 L4Re::Dataspace::Stats, 698 L4Re::Util::Dataspace::Svr, 574 allocate, 578 clear, 578 cl		
L4Re::Dataspace, 564 allocate, 570 clear, 569 copy_in, 570 flags, 572 info, 573 map, 568 May_flags, 567 map_region, 568 phys, 571 issie, 572 L4Re::Dataspace::Stats, 897 L4Re::Debug_obj, 579 debug, 581 L4Re::Debug_obj, 579 debug, 581 L4Re::Debug_obj, 579 L4Re::Debug_obj, 579 debug, 581 L4Re::Debug_obj, 579 copy, 577 factory, 618, 621 first_free_cutcb, 619, 622 get, 619 get_cap, 620 initial_caps, 619, 622 log, 618, 621 main_thread, 618, 621 main_thread, 618, 621 mem_alloc, 617, 620 rm, 617, 620 rm, 617, 620 rm, 617, 620 scheduler, 622 task, 618 ucb_area, 619, 622 L4Re::Event_buffer_t, 633 next, 634 put, 634 L4Re::Event_buffer_t< PAYLOAD >, 631 L4Re::Event_buffer_t< PAYLOAD >, 631 L4Re::Util::Ref_cap_alloc \ COUNTERTYPE >,		
allocate, 570 Region_flags, 858 reserve_area, 858, 859 copy_in, 570 tags, 572 tags, 572 tags, 573 tags_572 tags_573 tags_586 tags_Util::Auto_eap< T >, 453 tags_1859 tags_586 tags_Util::Auto_eap< T >, 453 tags_1859 tags_586 tags_Util::Auto_eap< T >, 453 tags_1859 tags_586 tags_Util::Auto_eap< T >, 455 tags_Util::Counting_cap_alloc< COUNTERTYPE >, 563 tags_Util::Even_Util:	L4Re::Dataspace, 564	
Clear, 569	•	
flags, 572	clear, 569	
flags, 572	copy_in, 570	L4Re::Smart_cap_auto< Unmap_flags >, 892
map, 568	flags, 572	
Map_flags, 567	info, 573	
map_region, 568 phys, 571 size, 572 L4Re::Dataspace::Stats, 897 L4Re::Debug_obj, 579 debug, 581 L4Re::Env, 615 env, 617 factory, 618, 621 first_free_utcb, 619, 622 get, 619 get cap, 620 initial_caps, 619, 622 log, 618, 621 man_thread, 618, 621 man_thread, 618, 621 man_thread, 618, 621 man_alloc, 617, 620 rm, 617, 620 rm, 617, 620 rm, 617, 620 taks, 618 scheduler, 622 taks, 618 scheduler, 622 taks, 619 scheduler, 622 taks, 619 scheduler, 623 get_buffer, 626 L4Re::Event_buffer_t Event_buffer_t Event_buffer_t Event_buffer_t Event_buffer_t Event_buffer_t San Are::Utii::Event_touffer_t PAYLOAD >, 635 L4Re::Utii::Event_touffer_t PAYLOAD >, 635 L4Re::Utii::Event_buffer_t PAYLOAD >, 635 L4Re::Utii::Event_buffer_t Are::Utii::Event_touffer_t PAYLOAD >, 635 L4Re::Utii::Event_buffer_t PAYLOAD >, 635 L4Re::Utii::Event_buffer_t PAYLOAD >, 636 L4Re::Utii::Event_touffer_t PAYLOAD >, 637 L4Re::Utii::Event_touffer_t PAYLOAD >, 637 L4Re::Utii::Event_touffer_t Are::Utii::Event_touffer_t PAYLOAD >, 637 L4Re::Utii::Event_touffer_t Are::Utii::Event_touffer_t Are::Utii::Event_touffer_t Are::Utii::Ref_cap< T >, 848 L4Re::Utii::Ref_cap< T >, 848 L4Re::Utii::Ref_cal_cap< T >, 849 print, 801 L4Re::Utii::Ref_cal_cap< T >, 849 L4Re::Utii::Ref_cal_cap< T >, 849 L4Re::Utii::Ref_cal_count_cap< Unmap_flags >, 891 L4Re::Utii::Smar_count_cap< Unmap_flags >, 893	map, 568	L4Re::Util::Cap_alloc_base, 544
phys, 571 size, 572 L4Re::Dataspace::Stats, 897 L4Re::Debug_obj, 579 debug, 581 cquive first_free_cap, 618, 621 first_free_cap, 618, 621 first_free_utcb, 619, 622 get, 619 get_cap, 620 initial_caps, 619, 622 log, 618, 621 main_thread, 618, 621 main_thread, 618, 621 main_thread, 618, 621 main_thread, 618, 621 parent, 617, 620 rm, 617, 620 rm, 617, 621 scheduler, 622 task, 618 utcb_area, 619, 622 L4Re::Event_buffer, 630 put, 634 L4Re::Util::Event_buffer_t< PAYLOAD >, 631 L4Re::Util::Event_buffer_t< PAYLOAD >, 637 L4Re::Util::Event_buffer_t< pay-to-specific plants L4Re::Util::Event_t< pay-to-	Map_flags, 567	L4Re::Util::Counting_cap_alloc< COUNTERTYPE >,
Size, 572	map_region, 568	563
L4Re::Debug_obj, 579 debug, 581 L4Re::Env, 615 env, 617 factory, 618, 621 first_free_cap, 618, 621 first_free_utcb, 619, 622 get, 619 get_cap, 620 initial_caps, 619, 622 log, 618, 621 mam_lhread, 618, 621 mam_alloc, 617, 621 parent, 617, 620 rm, 617, 621 sak, 618 utcb_area, 619, 622 task, 618 utcb_area, 619, 622 buffer, 639 init, 639 irq, 639 irq, 639 int, 639 int, 639 irq, 639 int, 639 irq, 639 int, 639 irq, 639 int, 639 irq, 639 int, 639	phys, 571	L4Re::Util::Dataspace_svr, 574
L4Re::Debug_obj, 579 debug, 581 L4Re::Env, 615 env, 617 factory, 618, 621 first_free_cap, 618, 621 pgt_cap, 620 initial_caps, 619, 622 log, 618, 621 mem_alloc, 617, 621 parent, 617, 620 rend, 617, 620 rend, 617, 621 scheduler, 622 task, 618 utob_area, 619, 622 task, 618 utob_area, 619, 622 L4Re::Event_buffer_t Event_buffer_t Event_buffer_t Event_buffer_t< R33 next, 634 L4Re::Event_buffer_t< PAYLOAD >, 631 L4Re::Event_buffer_t< PAYLOAD >; 639 print, 801 print, 802 print, 802 print, 803 print, 803 pri	size, 572	allocate, 578
debug, 581 is_static, 578 map, 576 map, 577 map, 576 map, 577 m	L4Re::Dataspace::Stats, 897	clear, 578
L4Re::Env, 615 env, 617 env, 617 factory, 618, 621 first_free_cap, 618, 621 first_free_toth, 619, 622 get, 619 get_cap, 620 initial_caps, 619, 622 log, 618, 621 main_thread, 618, 621 page_shift, 578 ptocked for get_cap, 620 initial_caps, 619, 622 log, 618, 621 main_thread, 618, 621 parent, 617, 620 rm, 617, 620 rm, 617, 621 scheduler, 622 task, 618 utcb_area, 619, 622 L4Re::Event_buffer_tonsumer_t< ptd>PAYLOAD >, 635 L4Re::Util::Event_buffer_tonsumer_t< PAYLOAD >, 635 L4Re::Event_buffer_t initial_caps, 619, 622 L4Re::Event_buffer_t attach, 637 buf, 636 detach, 637 tub_area, 619, 622 L4Re::Event_buffer_t get_buffer, 626 L4Re::Event_buffer_t sint, 639 irq, 639 Mode, 638 put, 634 L4Re::Event_buffer_t< PAYLOAD >, 631 L4Re::Event_buffer_t< PAYLOAD >, 637 L4Re::Event_buffer_t< PAYLOAD >, 637 L4Re::Util::Event_t< PAYLOAD >, 637 L4Re::Util::Ref_cap< T >, 848 print, 801 print, 801 print, 801 L4Re::Util::Smart_cap_auto< Unmap_flags >, 891 L4Re::Util::Smart_cap_auto< Unmap_flags >, 893	L4Re::Debug_obj, 579	copy, 577
env, 617 factory, 618, 621 first_free_cap, 618, 621 phys, 577 first_free_cutcb, 619, 622 get, 619 get_cap, 620 initial_caps, 619, 622 log, 618, 621 main_thread, 618, 621 parent, 617, 621 scheduler, 622 task, 618 utcb_area, 619, 622 t4Re::Event, 623 get_buffer, 626 t4Re::Event_buffer_t Event_buffer_t< PAYLOAD >, 631 t4Re::Event_buffer_t< PAYLOAD >, 637 t4Re::Util::Event_t PAYLOAD >, 637 t4Re::Event_buffer_t< PAYLOAD >, 637 t4Re::Util::Event_t PAYLOAD >, 637 t4Re::Event_buffer_t< PAYLOAD >, 631 t4Re::Event_buffer_t< PAYLOAD >; 637 t4Re::Util::Event_t< PAYLOAD >, 637 t4Re::Util::Event_t< PAYLOAD	debug, 581	is_static, 578
factory, 618, 621 first_free_cap, 618, 621 first_free_cap, 618, 621 first_free_utcb, 619, 622 get, 619 get_cap, 620 log, 618, 621 main_thread, 618, 621 parent, 617, 621 parent, 617, 620 take, 618 utcb_area, 619, 622 take, 618 utcb_area, 619, 622 take::Event_buffer_t Event_buffer_t Event_buffer_t Event_buffer_t, 633 next, 634 put, 634 L4Re::Event_buffer_t< PAYLOAD >, 631 L4Re::Util::Event_buffer_t< PAYLOAD >, 637 L4Re::Util::Smart_cap_auto< Unmap_flags >, 891 L4Re::Util::Smart_count_cap< Unmap_flags >, 893	L4Re::Env, 615	map, 576
first_free_cap, 618, 621 first_free_utcb, 619, 622 get, 619 get_cap, 620 log, 618, 621 main_thread, 618, 621 parent, 617, 620 take, 577 take, 577 take, 577 get_cap, 620 L4Re::Util::Event_buffer_consumer_t foreach_available_event, 630 process, 630 L4Re::Util::Event_buffer_consumer_t 627 L4Re::Util::Event_buffer_t attach, 637 take, 618 cetaphore consumer_t 627 L4Re::Util::Event_buffer_t attach, 637 buf, 636 detach, 637 utcb_area, 619, 622 L4Re::Event, 623 get_buffer, 626 L4Re::Event, 623 get_buffer, 626 L4Re::Event_buffer_t Event_buffer_t Event_buffer_t Event_buffer_t Event_buffer_t Event_buffer_t Event_buffer_t 639 L4Re::Util::Event_t PAYLOAD >, 635 L4Re::Util::Event_t buffer, 639 irq, 639 irq, 639 Mode, 638 L4Re::Util::Event_t PAYLOAD >, 637 L4Re::Util::Event_buffer_t L4Re::Util::Event_t PAYLOAD >, 637 L4Re::Util::Event_buffer_t L4Re::Util::Smart_cap_auto L4Re::Util::Rafe_cap L4Re::Util::Smart_cap_auto L4Re::Util::Smart_cap_auto L4Re::Util::Smart_cap_auto L4Re::Util::Smart_cap_auto L4Re::Util::Smart_cap_auto L4Re::Util::Smart_cap_auto	env, 617	map_hook, 576
first_free_utcb, 619, 622 get, 619 get_cap, 620 take, 577 get_cap, 620 log, 618, 621 main_thread, 618, 621 parent, 617, 620 take, 617 get_cap, 620 take::Utii::Event_buffer_consumer_t foreach_available_event, 630 process, 630 take::Utii::Event_buffer_consumer_t PAYLOAD >, 627 process, 630 take::Utii::Event_buffer_consumer_t PAYLOAD >, 627 parent, 617, 620 take::Utii::Event_buffer_t attach, 637 buf, 636 detach, 637 utcb_area, 619, 622 take, 618 utcb_area, 619, 622 take::Event, 623 take::Event, 623 take::Event_buffer_t get_buffer, 626 take::Event_buffer_t Event_buffer_t Event_buffer_t, 633 next, 634 put, 634 put, 634 take::Event_buffer_t buffer, 639 hode, 638 take::Event_buffer_t parent_t< PAYLOAD >, 637 take::Event_buffer_t	factory, 618, 621	page_shift, 578
get, 619 get_cap, 620 linitial_caps, 619, 622 log, 618, 621 main_thread, 618, 621 parent, 617, 621 parent, 617, 620 lt4Re::Util::Event_buffer_consumer_t foreach_available_event, 630 process, 630 lt4Re::Util::Event_buffer_consumer_t PAYLOAD >, PAYLOAD >	first_free_cap, 618, 621	phys, 577
get_cap, 620 initial_caps, 619, 622 log, 618, 621 main_thread, 618, 621 parent, 617, 620 rm, 617, 620 task, 618 utcb_area, 619, 622 L4Re::Util::Event_buffer_t< PAYLOAD >, 635 L4Re::Event_buffer_t get_buffer_t, 633 next, 634 put, 634 L4Re::Event_buffer_t< PAYLOAD >, 637 L4Re::Util::Event_buffer_t< PAYLOAD >, 637 L4Re::Util::Event_buffer_t< PAYLOAD >, 637 L4Re::Util::Event_buffer_t< PAYLOAD >, 637 L4Re::Util::Event_buffer_t< PAYLOAD >, 635 L4Re::Util::Event_buffer_t< PAYLOAD >, 635 L4Re::Util::Event_t buffer, 629 mode, 638 put, 634 put, 634 L4Re::Event_buffer_t< PAYLOAD >, 631 L4Re::Util::Ref_cap< T >, 848 print, 801 printn, 801 L4Re::Util::Smart_cap_auto< Unmap_flags >, 891 L4Re::Util::Smart_count_cap< Unmap_flags >, 893	first_free_utcb, 619, 622	release, 577
initial_caps, 619, 622	get, 619	
log, 618, 621	get_cap, 620	L4Re::Util::Event_buffer_consumer_t
main_thread, 618, 621 L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, mem_alloc, 617, 621 627 parent, 617, 620 L4Re::Util::Event_buffer_t rm, 617, 621 attach, 637 scheduler, 622 buf, 636 task, 618 detach, 637 utcb_area, 619, 622 L4Re::Util::Event_buffer_t< PAYLOAD >, 635 L4Re::Event, 623 L4Re::Util::Event_t get_buffer, 626 buffer, 639 L4Re::Event_buffer_t init, 639 Event_buffer_t, 633 irq, 639 next, 634 Mode, 638 put, 634 L4Re::Util::Event_t< PAYLOAD >, 637 L4Re::Event_buffer_t< PAYLOAD >::Event, 626 L4Re::Util::Names::Name, 819 L4Re::Util::Ref_cap< T >, 848 print, 801 L4Re::Util::Smart_cap_auto< Unmap_flags >, 891 L4Re::Mem_alloc, 802 L4Re::Util::Smart_count_cap< Unmap_flags >, 893	initial_caps, 619, 622	foreach_available_event, 630
mem_alloc, 617, 621 627 parent, 617, 620 L4Re::Util::Event_buffer_t rm, 617, 621 attach, 637 scheduler, 622 buf, 636 task, 618 detach, 637 utcb_area, 619, 622 L4Re::Util::Event_buffer_t< PAYLOAD >, 635 L4Re::Event, 623 L4Re::Util::Event_t get_buffer, 626 buffer, 639 L4Re::Event_buffer_t init, 639 irq, 639 irq, 639 next, 634 Mode, 638 put, 634 L4Re::Util::Event_t < PAYLOAD >, 637 L4Re::Event_buffer_t < PAYLOAD > ::Event, 626 L4Re::Util::Names::Name, 819 L4Re::Log, 798 L4Re::Util::Ref_cap < T >, 848 print, 801 L4Re::Util::Smart_cap_auto < Unmap_flags >, 891 L4Re::Mem_alloc, 802 L4Re::Util::Smart_count_cap < Unmap_flags >, 893		•
Darent, 617, 620	-	L4Re::Util::Event_buffer_consumer_t< PAYLOAD >,
rm, 617, 621 attach, 637 scheduler, 622 buf, 636 task, 618 detach, 637 utcb_area, 619, 622 L4Re::Util::Event_buffer_t< PAYLOAD >, 635 L4Re::Event, 623 L4Re::Util::Event_t get_buffer, 626 buffer, 639 L4Re::Event_buffer_t init, 639 Event_buffer_t, 633 irq, 639 next, 634 Mode, 638 put, 634 L4Re::Util::Event_t< PAYLOAD >, 637 L4Re::Event_buffer_t< PAYLOAD >::Event, 626 L4Re::Util::Names::Name, 819 L4Re::Log, 798 L4Re::Util::Ref_cap< T >, 848 print, 801 L4Re::Util::Smart_cap_auto< Unmap_flags >, 891 L4Re::Mem_alloc, 802 L4Re::Util::Smart_count_cap< Unmap_flags >, 893	mem_alloc, 617, 621	
scheduler, 622 buf, 636 task, 618 detach, 637 utcb_area, 619, 622 L4Re::Util::Event_buffer_t L4Re::Event, 623 L4Re::Util::Event_t get_buffer, 626 buffer, 639 L4Re::Event_buffer_t init, 639 Event_buffer_t, 633 irq, 639 next, 634 Mode, 638 put, 634 L4Re::Util::Event_t< PAYLOAD >, 637 L4Re::Event_buffer_t< PAYLOAD >::Event, 626 L4Re::Util::Names::Name, 819 L4Re::Log, 798 L4Re::Util::Ref_cap< T >, 848 print, 801 L4Re::Util::Ref_del_cap< T >, 849 printn, 801 L4Re::Util::Smart_cap_auto Unmap_flags >, 891 L4Re::Mem_alloc, 802 L4Re::Util::Smart_count_cap Unmap_flags >, 893	•	
task, 618 detach, 637 utcb_area, 619, 622 L4Re::Util::Event_buffer_t< PAYLOAD >, 635 L4Re::Event, 623 L4Re::Util::Event_t get_buffer, 626 buffer, 639 L4Re::Event_buffer_t init, 639 Event_buffer_t, 633 irq, 639 next, 634 Mode, 638 put, 634 L4Re::Util::Event_t< PAYLOAD >, 637 L4Re::Event_buffer_t< PAYLOAD >::Event, 626 L4Re::Util::Names::Name, 819 L4Re::Log, 798 L4Re::Util::Ref_cap< T >, 848 print, 801 L4Re::Util::Ref_del_cap< T >, 849 L4Re::Util::Smart_cap_auto< Unmap_flags >, 891 L4Re::Util::Smart_count_cap< Unmap_flags >, 893		
utcb_area, 619, 622 L4Re::Util::Event_buffer_t PAYLOAD >, 635 L4Re::Event, 623 L4Re::Util::Event_t get_buffer, 626 buffer, 639 L4Re::Event_buffer_t init, 639 Event_buffer_t, 633 irq, 639 next, 634 Mode, 638 put, 634 L4Re::Util::Event_t < PAYLOAD >, 637 L4Re::Event_buffer_t < PAYLOAD >, 631 L4Re::Util::Item_alloc_base, 718 L4Re::Util::Names::Name, 819 L4Re::Util::Ref_cap < T >, 848 L4Re::Util::Ref_del_cap < T >, 848 L4Re::Util::Ref_del_cap < T >, 849 print, 801 L4Re::Util::Smart_cap_auto < Unmap_flags >, 891 L4Re::Mem_alloc, 802 L4Re::Util::Smart_count_cap < Unmap_flags >, 893		
L4Re::Event, 623		
$\begin{array}{llllllllllllllllllllllllllllllllllll$		
L4Re::Event_buffer_t		-
Event_buffer_t, 633	- -	
next, 634 put, 634 L4Re::Util::Event_t< PAYLOAD >, 637 L4Re::Event_buffer_t< PAYLOAD >, 631 L4Re::Event_buffer_t< PAYLOAD >::Event, 626 L4Re::Util::Names::Name, 819 L4Re::Util::Ref_cap< T >, 848 print, 801 printn, 801 L4Re::Util::Ref_del_cap< T >, 849 L4Re::Util::Smart_cap_auto< Unmap_flags >, 891 L4Re::Mem_alloc, 802 L4Re::Util::Smart_count_cap< Unmap_flags >, 893		
put, 634 L4Re::Util::Event_t< PAYLOAD >, 637 L4Re::Event_buffer_t< PAYLOAD >, 631 L4Re::Util::Item_alloc_base, 718 L4Re::Util::Names::Name, 819 L4Re::Util::Ref_cap< T >, 848 print, 801 printn, 801 L4Re::Util::Ref_del_cap< T >, 849 L4Re::Util::Smart_cap_auto< Unmap_flags >, 891 L4Re::Mem_alloc, 802 L4Re::Util::Smart_count_cap< Unmap_flags >, 893		•
L4Re::Event_buffer_t< PAYLOAD >, 631 L4Re::Event_buffer_t< PAYLOAD >::Event, 626 L4Re::Util::Item_alloc_base, 718 L4Re::Util::Names::Name, 819 L4Re::Util::Ref_cap< T >, 848 print, 801 printn, 801 L4Re::Util::Ref_del_cap< T >, 849 L4Re::Util::Smart_cap_auto< Unmap_flags >, 891 L4Re::Mem_alloc, 802 L4Re::Util::Smart_count_cap< Unmap_flags >, 893		
L4Re::Event_buffer_t< PAYLOAD >::Event, 626 L4Re::Util::Names::Name, 819 L4Re::Log, 798 L4Re::Util::Ref_cap< T >, 848 print, 801 L4Re::Util::Ref_del_cap< T >, 849 printn, 801 L4Re::Util::Smart_cap_auto< Unmap_flags >, 891 L4Re::Mem_alloc, 802 L4Re::Util::Smart_count_cap< Unmap_flags >, 893	•	
	- -	
print, 801 L4Re::Util::Ref_del_cap< T >, 849 printn, 801 L4Re::Util::Smart_cap_auto< Unmap_flags >, 891 L4Re::Mem_alloc, 802 L4Re::Util::Smart_count_cap< Unmap_flags >, 893		
printn, 801 L4Re::Util::Smart_cap_auto< Unmap_flags >, 891 L4Re::Mem_alloc, 802 L4Re::Util::Smart_count_cap< Unmap_flags >, 893		_ ·
L4Re::Mem_alloc, 802 L4Re::Util::Smart_count_cap< Unmap_flags >, 893	•	·
	•	_ ,
alloc, 805 L4He::Util::Vcon_svr		
	alioc, 805	L4Ne::Util::Vcon_svf

dispatch, 936	dump, 557
L4Re::Util::Vcon_svr< SVR >, 935	get, 556
L4Re::Util::Video::Goos_svr, 669	operator==, 556
dispatch, 672	set, 556
get_fb, 671	shift, 556
init_infos, 673	size, <u>556</u>
refresh, 672	L4Re::Video::Goos, 665
screen_info, 671	create_buffer, 668
view_info, 671	create_view, 668
L4Re::Vfs, 447	delete_buffer, 668
L4Re::Vfs::Be_file, 492	delete_view, 669
data_space, 495	Flags, 667
fstat64, 495	get_static_buffer, 668
unlock_all_locks, 495	info, 667
L4Re::Vfs::Be_file_system, 496	view, 669
∼Be file system, 498	L4Re::Video::Goos::Info, 683
Be_file_system, 498	auto refresh, 685
type, 498	L4Re::Video::Pixel info, 843
L4Re::Vfs::Directory, 592	a, 845, 847
faccessat, 595	b, 845, 847
link, 596	bits_per_pixel, 846
mkdir, 595	bytes_per_pixel, 846, 847
rename, 595	dump, 847
,	•
rmdir, 596	g, 845, 846
symlink, 596	has_alpha, 846
unlink, 595	operator==, 847
L4Re::Vfs::File, 651	Pixel_info, 844, 845
L4Re::Vfs::File_system, 653	r, 845, 846
mount, 655	L4Re::Video::View, 948
type, 655	Flags, 949
L4Re::Vfs::Fs, 656	info, 950
alloc_fd, 659	refresh, 950
free_fd, 659	set_info, 950
get_file, 659	set_viewport, 950
mount, 660	stack, 950
set_fd, 659	V_flags, 949
L4Re::Vfs::Generic_file, 660	L4Re::Video::View::Info, 685
fchmod, 663	flags, 687
fstat64, 663	l4io_device_types_t
get_status_flags, 663	IO interface, 383
set_status_flags, 663	l4io_has_resource
unlock_all_locks, 662	IO interface, 387
L4Re::Vfs::Mman, 815	l4io_iomem_flags_t
L4Re::Vfs::Ops, 829	IO interface, 383
L4Re::Vfs::Regular_file, 850	l4io_lookup_device
data_space, 853	IO interface, 386
fdatasync, 854	l4io_lookup_resource
fsync, 854	IO interface, 387
ftruncate64, 853	l4io_release_iomem
get_lock, 854	IO interface, 384
Iseek64, 853	l4io release ioport
ready, 853	IO interface, 386
set_lock, 854	l4io_request_iomem
writev, 853	IO interface, 384
L4Re::Vfs::Special_file, 893	l4io_request_iomem_region
ioctl, 895	IO interface, 384
L4Re::Video::Color_component, 553	l4io_request_ioport
Color_component, 555	IO interface, 386
Joiot_Component, JJJ	io interiace, 300

14io request resource iomem	l4re_ds_size
IO interface, 387	Dataspace interface, 59
l4io_resource_t	l4re_ds_stats_t, 767
IO interface, 383	l4re_elf_aux_mword_t, 768
I4io_resource_types_t	14re elf aux t, 769
IO interface, 383	l4re_elf_aux_vma_t, 769
l4io_search_iomem_region	l4re env
IO interface, 384	Initial Environment, 102
l4irq_attach	l4re_env_cap_entry_t, 770
Interface using direct functionality., 390	flags, 771
l4irq_attach_cap	l4re_env_cap_entry_t, 771
Interface using direct functionality., 396	l4re_env_cap_entry_t, 771
l4irq_attach_cap_ft	l4re_env_get_cap
Interface using direct functionality., 396	Initial Environment, 103
l4irq_attach_ft	l4re_env_get_cap_e
Interface using direct functionality., 391	Initial Environment, 103
l4irq_attach_thread	l4re_env_get_cap_l
-	Initial Environment, 105
Interface using direct functionality., 391	
l4irq_attach_thread_cap	l4re_env_t, 772
Interface using direct functionality., 397	Initial Environment, 102
l4irq_attach_thread_cap_ft	I4re_event_get_axis_info
Interface using direct functionality., 397	Event interface, 64
l4irq_attach_thread_ft	I4re_event_get_buffer
Interface using direct functionality., 391	Event interface, 62
l4irq_detach	I4re_event_get_num_streams
Interface using direct functionality., 393	Event interface, 62
l4irq_release	l4re_event_get_stream_info
Interface for asynchronous ISR handlers., 395	Event interface, 64
l4irq_request	I4re_event_get_stream_info_for_id
Interface for asynchronous ISR handlers., 394	Event interface, 64
l4irq_request_cap	l4re_event_t, 773
Interface for asynchronous ISR handlers with a	l4re_kip
given IRQ capability., 398	Initial Environment, 103
l4irq_unmask	l4re_log_print
Interface using direct functionality., 392	Log interface, 66
l4irq_unmask_and_wait_any	l4re_log_print_srv
Interface using direct functionality., 392	Log interface, 67
l4irq_wait	l4re_log_printn
Interface using direct functionality., 392	Log interface, 67
l4irq_wait_any	l4re_log_printn_srv
Interface using direct functionality., 392	Log interface, 68
l4kd_inchar	l4re_ma_alloc
Kernel Debugger, 188	Memory allocator, 70
I4re_aux_t, 766	l4re_ma_alloc_align
I4re_debug_obj_debug	Memory allocator, 71
Debug interface, 61	l4re_ma_alloc_align_srv
l4re_ds_allocate	Memory allocator, 73
Dataspace interface, 59	l4re_ma_flags
l4re_ds_clear	Memory allocator, 69
Dataspace interface, 58	l4re ma free
I4re_ds_copy_in	Memory allocator, 72
Dataspace interface, 59	l4re_ma_free_srv
l4re_ds_flags	Memory allocator, 73
Dataspace interface, 59	l4re_ns_query_to_srv
l4re_ds_info	Namespace interface, 75
Dataspace interface, 59	l4re_ns_register_flags
l4re_ds_phys	Namespace interface, 75
Dataspace interface, 59	l4re_ns_register_obj_srv

Namespace interface, 75	l4re_video_view_info_flags_t
I4re_rm_attach	Video API, 91
Region map interface, 79	l4re_video_view_info_t, 777
I4re_rm_attach_srv Region map interface, 84	I4re_video_view_refresh Video API, 93
l4re_rm_detach	l4re_video_view_set_info
Region map interface, 80	Video API, 93
l4re_rm_detach_ds	l4re_video_view_set_viewport
Region map interface, 81	Video API, 95
l4re_rm_detach_ds_unmap	l4re_video_view_stack
Region map interface, 82	Video API, 95
l4re rm detach srv	l4re_video_view_t, 779
Region map interface, 85	Video API, 90
l4re_rm_detach_unmap	l4shmc_add_chunk
Region map interface, 81	Chunks, 418
l4re_rm_find	l4shmc_add_signal
Region map interface, 83	Signals, 428
l4re_rm_find_srv	l4shmc_area_overhead
Region map interface, 85	Shared Memory Library, 416
l4re_rm_flags_t	l4shmc_area_size
Region map interface, 78	Shared Memory Library, 416
l4re_rm_free_area	l4shmc_area_size_free
Region map interface, 78	Shared Memory Library, 416
l4re_rm_free_area_srv	l4shmc_attach
Region map interface, 84	Shared Memory Library, 415
l4re_rm_reserve_area	l4shmc_attach_signal
Region map interface, 78	Signals, 429
Mrs. rm. rosovice area and	Mahma attach signal to
Idre_rm_reserve_area_srv	l4shmc_attach_signal_to
Region map interface, 84	Signals, 429
Region map interface, 84 I4re_rm_show_lists	Signals, 429 l4shmc_attach_to
Region map interface, 84 I4re_rm_show_lists Region map interface, 83	Signals, 429 l4shmc_attach_to Shared Memory Library, 415
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic
Region map interface, 84 I4re_rm_show_lists Region map interface, 83 I4re_util_cap_last Capability allocator, 87	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431
Region map interface, 84 I4re_rm_show_lists Region map interface, 83 I4re_util_cap_last Capability allocator, 87 I4re_util_kumem_alloc	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity
Region map interface, 84 I4re_rm_show_lists Region map interface, 83 I4re_util_cap_last Capability allocator, 87 I4re_util_kumem_alloc Kumem allocator utility, 88	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity Chunks, 421
Region map interface, 84 I4re_rm_show_lists Region map interface, 83 I4re_util_cap_last Capability allocator, 87 I4re_util_kumem_alloc Kumem allocator utility, 88 I4re_video_color_component_t, 774	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity
Region map interface, 84 I4re_rm_show_lists Region map interface, 83 I4re_util_cap_last Capability allocator, 87 I4re_util_kumem_alloc Kumem allocator utility, 88	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity Chunks, 421 l4shmc_chunk_consumed
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity Chunks, 421 l4shmc_chunk_consumed Consumer, 425
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity Chunks, 421 l4shmc_chunk_consumed Consumer, 425 l4shmc_chunk_overhead
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view	Signals, 429 I4shmc_attach_to Shared Memory Library, 415 I4shmc_check_magic Signals, 431 I4shmc_chunk_capacity Chunks, 421 I4shmc_chunk_consumed Consumer, 425 I4shmc_chunk_overhead Shared Memory Library, 417
Region map interface, 84 I4re_rm_show_lists Region map interface, 83 I4re_util_cap_last Capability allocator, 87 I4re_util_kumem_alloc Kumem allocator utility, 88 I4re_video_color_component_t, 774 I4re_video_goos_create_buffer Video API, 92 I4re_video_goos_create_view Video API, 92	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity Chunks, 421 l4shmc_chunk_consumed Consumer, 425 l4shmc_chunk_overhead Shared Memory Library, 417 l4shmc_chunk_ptr Chunks, 421 l4shmc_chunk_ready
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_view	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity Chunks, 421 l4shmc_chunk_consumed Consumer, 425 l4shmc_chunk_overhead Shared Memory Library, 417 l4shmc_chunk_ptr Chunks, 421
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_view Video API, 93	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity Chunks, 421 l4shmc_chunk_consumed Consumer, 425 l4shmc_chunk_overhead Shared Memory Library, 417 l4shmc_chunk_ptr Chunks, 421 l4shmc_chunk_ready Producer, 422 l4shmc_chunk_ready_sig
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_buffer Video API, 93 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer	Signals, 429 I4shmc_attach_to Shared Memory Library, 415 I4shmc_check_magic Signals, 431 I4shmc_chunk_capacity Chunks, 421 I4shmc_chunk_consumed Consumer, 425 I4shmc_chunk_overhead Shared Memory Library, 417 I4shmc_chunk_ptr Chunks, 421 I4shmc_chunk_ready Producer, 422 I4shmc_chunk_ready_sig Producer, 423
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_buffer Video API, 93 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer Video API, 92	Signals, 429 I4shmc_attach_to Shared Memory Library, 415 I4shmc_check_magic Signals, 431 I4shmc_chunk_capacity Chunks, 421 I4shmc_chunk_consumed Consumer, 425 I4shmc_chunk_overhead Shared Memory Library, 417 I4shmc_chunk_ptr Chunks, 421 I4shmc_chunk_ready Producer, 422 I4shmc_chunk_ready_sig Producer, 423 I4shmc_chunk_signal
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_buffer Video API, 93 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer Video API, 92 l4re_video_goos_get_view	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity Chunks, 421 l4shmc_chunk_consumed Consumer, 425 l4shmc_chunk_overhead Shared Memory Library, 417 l4shmc_chunk_ptr Chunks, 421 l4shmc_chunk_ready Producer, 422 l4shmc_chunk_ready_sig Producer, 423 l4shmc_chunk_signal Chunks, 421
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer Video API, 92 l4re_video_goos_get_static_buffer Video API, 93 l4re_video_goos_get_view Video API, 93	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity Chunks, 421 l4shmc_chunk_consumed Consumer, 425 l4shmc_chunk_overhead Shared Memory Library, 417 l4shmc_chunk_ptr Chunks, 421 l4shmc_chunk_ready Producer, 422 l4shmc_chunk_ready_sig Producer, 423 l4shmc_chunk_signal Chunks, 421 l4shmc_chunk_size
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer Video API, 92 l4re_video_goos_get_view Video API, 93 l4re_video_goos_get_view Video API, 93 l4re_video_goos_goos_info	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity Chunks, 421 l4shmc_chunk_consumed Consumer, 425 l4shmc_chunk_overhead Shared Memory Library, 417 l4shmc_chunk_ptr Chunks, 421 l4shmc_chunk_ready Producer, 422 l4shmc_chunk_ready_sig Producer, 423 l4shmc_chunk_signal Chunks, 421 l4shmc_chunk_size Consumer, 427
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer Video API, 92 l4re_video_goos_get_view Video API, 93 l4re_video_goos_get_view Video API, 93 l4re_video_goos_get_view Video API, 93 l4re_video_goos_info Video API, 91	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity Chunks, 421 l4shmc_chunk_consumed Consumer, 425 l4shmc_chunk_overhead Shared Memory Library, 417 l4shmc_chunk_ptr Chunks, 421 l4shmc_chunk_ready Producer, 422 l4shmc_chunk_ready_sig Producer, 423 l4shmc_chunk_signal Chunks, 421 l4shmc_chunk_size Consumer, 427 l4shmc_chunk_try_to_take
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer Video API, 92 l4re_video_goos_get_view Video API, 93 l4re_video_goos_get_view Video API, 93 l4re_video_goos_info Video API, 91 l4re_video_goos_info_flags_t	Signals, 429 I4shmc_attach_to Shared Memory Library, 415 I4shmc_check_magic Signals, 431 I4shmc_chunk_capacity Chunks, 421 I4shmc_chunk_consumed Consumer, 425 I4shmc_chunk_overhead Shared Memory Library, 417 I4shmc_chunk_ptr Chunks, 421 I4shmc_chunk_ready Producer, 422 I4shmc_chunk_ready_sig Producer, 423 I4shmc_chunk_signal Chunks, 421 I4shmc_chunk_size Consumer, 427 I4shmc_chunk_try_to_take Producer, 422
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer Video API, 92 l4re_video_goos_get_view Video API, 93 l4re_video_goos_get_view Video API, 93 l4re_video_goos_info Video API, 91 l4re_video_goos_info_flags_t Video API, 90	Signals, 429 I4shmc_attach_to Shared Memory Library, 415 I4shmc_check_magic Signals, 431 I4shmc_chunk_capacity Chunks, 421 I4shmc_chunk_consumed Consumer, 425 I4shmc_chunk_overhead Shared Memory Library, 417 I4shmc_chunk_ptr Chunks, 421 I4shmc_chunk_ready Producer, 422 I4shmc_chunk_ready_sig Producer, 423 I4shmc_chunk_signal Chunks, 421 I4shmc_chunk_size Consumer, 427 I4shmc_chunk_try_to_take Producer, 422 I4shmc_connect_chunk_signal
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer Video API, 93 l4re_video_goos_get_view Video API, 93 l4re_video_goos_info Video API, 91 l4re_video_goos_info Video API, 90 l4re_video_goos_info_t, 775	Signals, 429 I4shmc_attach_to Shared Memory Library, 415 I4shmc_check_magic Signals, 431 I4shmc_chunk_capacity Chunks, 421 I4shmc_chunk_consumed Consumer, 425 I4shmc_chunk_overhead Shared Memory Library, 417 I4shmc_chunk_ptr Chunks, 421 I4shmc_chunk_ready Producer, 422 I4shmc_chunk_ready Producer, 423 I4shmc_chunk_signal Chunks, 421 I4shmc_chunk_size Consumer, 427 I4shmc_chunk_try_to_take Producer, 422 I4shmc_connect_chunk_signal Shared Memory Library, 416
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer Video API, 92 l4re_video_goos_get_view Video API, 93 l4re_video_goos_get_view Video API, 93 l4re_video_goos_info Video API, 91 l4re_video_goos_info Video API, 90 l4re_video_goos_info_t, 775 l4re_video_goos_refresh	Signals, 429 l4shmc_attach_to Shared Memory Library, 415 l4shmc_check_magic Signals, 431 l4shmc_chunk_capacity Chunks, 421 l4shmc_chunk_consumed Consumer, 425 l4shmc_chunk_overhead Shared Memory Library, 417 l4shmc_chunk_ptr Chunks, 421 l4shmc_chunk_ready Producer, 422 l4shmc_chunk_ready_sig Producer, 423 l4shmc_chunk_signal Chunks, 421 l4shmc_chunk_size Consumer, 427 l4shmc_chunk_try_to_take Producer, 422 l4shmc_connect_chunk_signal Shared Memory Library, 416 l4shmc_create
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer Video API, 92 l4re_video_goos_get_view Video API, 93 l4re_video_goos_get_view Video API, 91 l4re_video_goos_info Video API, 90 l4re_video_goos_info_t, 775 l4re_video_goos_refresh Video API, 91	Signals, 429 I4shmc_attach_to Shared Memory Library, 415 I4shmc_check_magic Signals, 431 I4shmc_chunk_capacity Chunks, 421 I4shmc_chunk_consumed Consumer, 425 I4shmc_chunk_overhead Shared Memory Library, 417 I4shmc_chunk_ptr Chunks, 421 I4shmc_chunk_ready Producer, 422 I4shmc_chunk_ready_sig Producer, 423 I4shmc_chunk_signal Chunks, 421 I4shmc_chunk_signal Chunks, 421 I4shmc_chunk_size Consumer, 427 I4shmc_chunk_try_to_take Producer, 422 I4shmc_connect_chunk_signal Shared Memory Library, 416 I4shmc_create Shared Memory Library, 415
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer Video API, 93 l4re_video_goos_get_view Video API, 93 l4re_video_goos_info Video API, 91 l4re_video_goos_info_flags_t Video API, 90 l4re_video_goos_info_t, 775 l4re_video_goos_refresh Video API, 91 l4re_video_goos_refresh Video API, 91 l4re_video_pixel_info_t, 776	Signals, 429 I4shmc_attach_to Shared Memory Library, 415 I4shmc_check_magic Signals, 431 I4shmc_chunk_capacity Chunks, 421 I4shmc_chunk_consumed Consumer, 425 I4shmc_chunk_overhead Shared Memory Library, 417 I4shmc_chunk_ptr Chunks, 421 I4shmc_chunk_ready Producer, 422 I4shmc_chunk_ready_sig Producer, 423 I4shmc_chunk_signal Chunks, 421 I4shmc_chunk_signal Chunks, 421 I4shmc_chunk_size Consumer, 427 I4shmc_chunk_try_to_take Producer, 422 I4shmc_chunk_try_to_take Producer, 422 I4shmc_connect_chunk_signal Shared Memory Library, 416 I4shmc_create Shared Memory Library, 415 I4shmc_enable_chunk
Region map interface, 84 l4re_rm_show_lists Region map interface, 83 l4re_util_cap_last Capability allocator, 87 l4re_util_kumem_alloc Kumem allocator utility, 88 l4re_video_color_component_t, 774 l4re_video_goos_create_buffer Video API, 92 l4re_video_goos_create_view Video API, 92 l4re_video_goos_delete_buffer Video API, 92 l4re_video_goos_delete_view Video API, 93 l4re_video_goos_get_static_buffer Video API, 92 l4re_video_goos_get_view Video API, 93 l4re_video_goos_get_view Video API, 91 l4re_video_goos_info Video API, 90 l4re_video_goos_info_t, 775 l4re_video_goos_refresh Video API, 91	Signals, 429 I4shmc_attach_to Shared Memory Library, 415 I4shmc_check_magic Signals, 431 I4shmc_chunk_capacity Chunks, 421 I4shmc_chunk_consumed Consumer, 425 I4shmc_chunk_overhead Shared Memory Library, 417 I4shmc_chunk_ptr Chunks, 421 I4shmc_chunk_ready Producer, 422 I4shmc_chunk_ready_sig Producer, 423 I4shmc_chunk_signal Chunks, 421 I4shmc_chunk_signal Chunks, 421 I4shmc_chunk_size Consumer, 427 I4shmc_chunk_try_to_take Producer, 422 I4shmc_connect_chunk_signal Shared Memory Library, 416 I4shmc_create Shared Memory Library, 415

Consumer, 433	Atomic Instructions, 332
l4shmc_get_chunk	l4util_atomic_inc
Chunks, 419	Atomic Instructions, 332
l4shmc_get_chunk_to	l4util_bsf
Chunks, 419	Bit Manipulation, 336
l4shmc_get_signal_to	l4util_bsr
Signals, 429	Bit Manipulation, 336
l4shmc_is_chunk_clear	l4util_btc
Producer, 423	Bit Manipulation, 336
l4shmc_is_chunk_ready	l4util_btr
Consumer, 425	Bit Manipulation, 335
l4shmc_iterate_chunk	l4util_bts
Chunks, 419	Bit Manipulation, 335
l4shmc_signal_cap	l4util_clear_bit
Signals, 431	Bit Manipulation, 335
l4shmc_trigger	I4util_cmpxchg
Producer, 432	Atomic Instructions, 330
l4shmc_wait_any	l4util_cmpxchg16
Consumer, 433	Atomic Instructions, 329
l4shmc_wait_any_to	I4util_cmpxchg32
Consumer, 434	Atomic Instructions, 329
l4shmc_wait_any_try	l4util_cmpxchg64
Consumer, 434	Atomic Instructions, 328
l4shmc_wait_chunk	I4util_cmpxchg8
Consumer, 424	Atomic Instructions, 329
l4shmc_wait_chunk_to	l4util_complement_bit
Consumer, 425	Bit Manipulation, 335
l4shmc_wait_chunk_try	I4util_cpu_capabilities
Consumer, 425	CPU related functions, 315
l4shmc_wait_signal	I4util_cpu_capabilities_nocheck
Consumer, 434	CPU related functions, 316
l4shmc_wait_signal_to	I4util_cpu_has_cpuid
Consumer, 434 I4shmc wait signal try	CPU related functions, 315
Consumer, 436	l4util_find_first_set_bit Bit Manipulation, 337
I4sigma0_debug_dump	l4util_find_first_zero_bit
Sigma0 API, 402	Bit Manipulation, 337
I4sigma0_map_anypage	l4util_idt_desc_t, 780
Sigma0_Hap_anypage Sigma0 API, 401	I4util idt header t, 780
I4sigma0_map_errstr	l4util in16
Sigma0_Hap_enstr	IA32 Port I/O API, 371
I4sigma0_map_iomem	l4util in32
Sigma0_Hap_ionerii	IA32 Port I/O API, 371
I4sigma0 map kip	Idutil in8
Sigma0_Nap_Np	IA32 Port I/O API, 370
I4sigma0_map_mem	I4util inc8
Sigma0_Hap_HeH	Atomic Instructions, 331
I4sigma0_map_tbuf	l4util_inc8_res
Sigma0 API, 401	Atomic Instructions, 331
I4sigma0_new_client	I4util ins16
Sigma0 API, 402	IA32 Port I/O API, 371
l4sigma0_return_flags_t	l4util_ins32
Sigma0 API, 400	IA32 Port I/O API, 372
I4util add8	I4util ins8
Atomic Instructions, 331	IA32 Port I/O API, 371
l4util_add8_res	I4util_kip_for_each_feature
Atomic Instructions, 331	Kernel Interface Page API, 358
l4util_atomic_add	l4util_kip_kernel_abi_version
<u></u>	. ram_nip_normor_doi_voroion

Kernel Interface Page API, 359	set, 897
l4util_kip_kernel_has_feature	State, 896
Kernel Interface Page API, 359	L4vcpu::Vcpu, 937
l4util_kip_kernel_is_ux	cast, 947
Kernel Interface Page API, 358	entry_ip, 945
l4util_mb_addr_range_t, 781	entry_sp, 945
I4util_mb_apm_t, 782	ext_alloc, 947
l4util_mb_drive_t, 783	i, 945
drive_cylinders, 785	irq_disable_save, 941
drive_mode, 784	irq_enable, 942
drive_number, 784	irq_restore, 942
l4util_mb_info_t, 785	is_irq_entry, 944
l4util_mb_mod_t, 787	is_page_fault_entry, 944
mod_end, 787	r, 944, 945
mod_start, 787	saved_state, 941, 942
l4util_mb_vbe_ctrl_t, 788	state, 941
l4util_mb_vbe_mode_t, 788	task, 943
l4util_memdesc_vm_high	wait_for_event, 943
Kernel Interface Page API, 359	I4vcpu_ext_alloc
l4util_micros2l4to	Extended vCPU support, 413
Utility Functions, 369	l4vcpu_irq_disable
I4util_next_power2	vCPU Support Library, 406
Bit Manipulation, 337	l4vcpu_irq_disable_save
l4util_out16	vCPU Support Library, 407
IA32 Port I/O API, 372	I4vcpu_irq_enable
l4util_out32	vCPU Support Library, 408
IA32 Port I/O API, 372	I4vcpu_irq_restore
I4util_out8	vCPU Support Library, 409
IA32 Port I/O API, 372	I4vcpu_irq_state_t
l4util_outs16	vCPU Support Library, 406
IA32 Port I/O API, 373	l4vcpu_is_irq_entry
I4util_outs32	vCPU Support Library, 411
IA32 Port I/O API, 373	l4vcpu_is_page_fault_entry
I4util_outs8	vCPU Support Library, 411
IA32 Port I/O API, 372	I4vcpu_print_state
l4util_rand	vCPU Support Library, 410
Random number support, 363	I4vcpu_state
l4util_set_bit	vCPU Support Library, 406
Bit Manipulation, 334	I4vcpu_wait_for_event
l4util_splitlog2_hdl	vCPU Support Library, 410 link
Utility Functions, 367	
	L4Re::Vfs::Directory, 596
Utility Functions, 368	List_alloc
	cxx::List_alloc, 793
Random number support, 363	log
	L4Re::Env, 618, 621
Bit Manipulation, 335	Log interface, 66
Atomic Instructions 221	l4re_log_print, 66
Atomic Instructions, 331	l4re_log_print_srv, 67
Atomic Instructions 220	l4re_log_printn, 67
Atomic Instructions, 330 l4util_xchg32	l4re_log_printn_srv, 68 Logging interface, 109
Atomic Instructions, 330	Low-Level Thread Functions, 365
	Low mask
I4util_xchg8 Atomic Instructions, 330	cxx::Bitfield, 501
	lower_bound_node
L4vcpu::State, 895 add, 896	cxx::Avl_map, 464
clear, 897	cxx::Avl_map, 464 cxx::Avl_set, 472
cicai, 037	0.7AVI_561, 4/2

cxx::Bits::Bst, 527	l4_mem_type_conventional, 238
Lsb	I4_mem_type_dedicated, 238
cxx::Bitfield, 501	14 mem type reserved, 238
Iseek64	
L4Re::Vfs::Regular_file, 853	l4_mem_type_shared, 238
L4nevisnegulai_lile, 655	I4_mem_type_undefined, 238
Machine Restarting Function, 364	Memory descriptors (C version), 237
magic	l4_kernel_info_get_mem_desc_end, 239
I4_vhw_descriptor, 759	l4_kernel_info_get_mem_desc_is_virtual, 239
main_thread	l4_kernel_info_get_mem_desc_start, 239
L4Re::Env, 618, 621	l4_kernel_info_get_mem_desc_subtype, 239
map	l4_kernel_info_get_mem_desc_type, 239
L4::Task, 903	l4_kernel_info_get_num_mem_descs, 238
L4Re::Dataspace, 568	I4_kernel_info_mem_desc_t, 238
L4Re::Util::Dataspace_svr, 576	<pre>l4_kernel_info_set_mem_desc, 238</pre>
Map ro	I4_mem_type_t, 238
L4Re::Dataspace, 567	Memory operations., 308
Map rw	L4_MEM_WIDTH_1BYTE, 308
L4Re::Dataspace, 567	L4_MEM_WIDTH_2BYTE, 308
Map flags	L4_MEM_WIDTH_4BYTE, 308
. – •	L4_mem_op_widths, 308
L4Re::Dataspace, 567	I4_mem_read, 309
map_hook	I4_mem_write, 310
L4Re::Util::Dataspace_svr, 576	Memory related, 175
map_region	L4_INVALID_ADDR, 177
L4Re::Dataspace, 568	L4_LOG2_PAGESIZE, 176
Mask	L4_LOG2_SUPERPAGESIZE, 176
cxx::Bitfield, 501	L4 PAGEMASK, 176
mask	L4_SUPERPAGEMASK, 176
L4::lcu, 680	L4_SUPERPAGESIZE, 176
Masks	I4_addr_consts_t, 177
cxx::Bitfield, 501	l4_round_page, 178
max	I4_round_size, 178
Small C++ Template Library, 44	l4_trunc_page, 177
max_free_slabs	I4_trunc_size, 177
cxx::Base_slab, 485	Message Items, 150
cxx::Base_slab_static, 489	L4_ITEM_CONT, 150
Mem_alloc	L4_ITEM_OONT, 150
L4Re Protocol identifiers, 114	
mem_alloc	L4_MAP_ITEM_GRANT, 150
L4Re::Env, 617, 621	L4_MAP_ITEM_MAP, 150
Mem_alloc_flags	L4_RCV_ITEM_LOCAL_ID, 151
L4Re::Mem_alloc, 805	L4_RCV_ITEM_SINGLE_CAP, 150
Mem_desc	I4_map_control, 151
L4::Kip::Mem_desc, 808	I4_map_obj_control, 151
mem_size	I4_msg_item_consts_t, 150
I4_vhw_entry, 761	Message Registers (MRs), 295
mem_start	Message Tag, 274
I4_vhw_entry, 761	L4_MSGTAG_ERROR, 276
Memory allocator, 69	L4_MSGTAG_FLAGS, 276
l4re_ma_alloc, 70	L4_MSGTAG_PROPAGATE, 276
l4re_ma_alloc_align, 71	L4_MSGTAG_SCHEDULE, 276
l4re_ma_alloc_align_srv, 73	L4_MSGTAG_TRANSFER_FPU, 276
l4re_ma_flags, 69	L4_MSGTAG_XCPU, 276
I4re_ma_free, 72	L4_PROTO_ALLOW_SYSCALL, 275
l4re_ma_free_srv, 73	L4_PROTO_EXCEPTION, 276
Memory allocator API, 110	L4_PROTO_FACTORY, 276
Memory descriptors (C version)	L4_PROTO_IO_PAGE_FAULT, 276
I4_mem_type_archspecific, 238	L4_PROTO_IRQ, 275
I4_mem_type_bootloader, 238	L4_PROTO_KOBJECT, 276

L4 PROTO LOG, 276	ELF binary format, 355
L4 PROTO META, 276	Name-space API, 111
L4 PROTO NONE, 275	Namespace
L4 PROTO PAGE FAULT, 275	L4Re Protocol identifiers, 114
L4 PROTO PF EXCEPTION, 275	Namespace interface, 75
L4 PROTO PREEMPTION, 275	l4re_ns_query_to_srv, 75
L4 PROTO SCHEDULER, 276	l4re_ns_register_flags, 75
L4 PROTO SIGMA0, 276	l4re_ns_register_obj_srv, 75
L4 PROTO SYS EXCEPTION, 275	next
L4 PROTO TASK, 276	L4Re::Event buffer t, 634
L4_PROTO_TASK, 276 L4_PROTO_THREAD, 276	
	No_init
L4_PROTO_VM, 276	L4::Cap_base, 547
I4_msgtag, 276	No_init_type
I4_msgtag_flags, 276, 279	L4::Cap_base, 547
I4_msgtag_has_error, 279	num_interfaces
I4_msgtag_is_exception, 281	L4::Meta, 814
I4_msgtag_is_io_page_fault, 282	Object Invocation, 205
I4_msgtag_is_page_fault, 280	L4 SYSF CALL, 207
I4_msgtag_is_preemption, 280	L4 SYSF NONE, 207
I4_msgtag_is_sigma0, 282	
I4_msgtag_is_sys_exception, 281	L4_SYSF_OPEN_WAIT, 207
I4_msgtag_items, 279	L4_SYSF_RECV, 207
I4_msgtag_label, 277	L4_SYSF_REPLY, 207
I4_msgtag_protocol, 275	L4_SYSF_REPLY_AND_WAIT, 207
I4_msgtag_t, 275	L4_SYSF_SEND, 207
I4_msgtag_words, 278	L4_SYSF_SEND_AND_WAIT, 207
min	L4_SYSF_WAIT, 207
Small C++ Template Library, 44	I4_ipc, 214
mkdir	I4_ipc_call, 210
L4Re::Vfs::Directory, 595	I4_ipc_receive, 209
mod_end	<pre>I4_ipc_reply_and_wait, 211</pre>
 4util_mb_mod_t, 787	I4_ipc_send, 207
mod start	<pre>I4_ipc_send_and_wait, 212</pre>
 4util_mb_mod_t, 787	I4_ipc_sleep, 215
Mode	14_ipc_wait, 208
L4Re::Util::Event t, 638	I4_sndfpage_add, 216
Mode_irq	I4_syscall_flags_t, 206
L4Re::Util::Event_t, 638	object_size
Mode_polling	cxx::Base_slab, 485
L4Re::Util::Event_t, 638	cxx::Base_slab_static, 489
modify_senders	objects_per_slab
L4::Thread, 918	cxx::Base_slab, 485
mount	cxx::Base_slab_static, 489
L4Re::Vfs::File_system, 655	operator I4_msgtag_t
L4Re::Vfs::Fs, 660	L4::Factory::S, 869
move	operator new
L4::Cap, 541	Small C++ Template Library, 44
Msb	operator Priv_type *
	cxx::Auto_ptr, 460
cxx::Bitfield, 501	operator<<
Msg_ptr	IPC Streams, 49d
L4::lpc::Msg_ptr, 819	L4::Factory::S, 869, 871
msg_ptr	operator>>
IPC Messaging Framework, 54	IPC Streams, 46d
msi_info	operator*
L4::lcu, 678	
N	cxx::Auto_ptr, 459
	operator()
cxx::Bits::Direction, 592	cxx::Pair_first_compare, 840
NT_VERSION	operator->

cxx::Auto ptr, 459	Parent API, 112
operator=	parse_cmdline
cxx::Auto_ptr, 458	Comfortable Command Line Parsing, 360
operator==	phys
L4Re::Video::Color_component, 556	L4Re::Dataspace, 571
L4Re::Video::Pixel info, 847	L4Re::Util::Dataspace_svr, 577
outchar	Pinned
Kernel Debugger, 186	L4Re::Mem_alloc, 805
outdec	Pixel_info
Kernel Debugger, 187	L4Re::Video::Pixel_info, 844, 845
outhex12	print
Kernel Debugger, 187	L4Re::Log, 801
outhex16	printn
Kernel Debugger, 187	L4Re::Log, 801
outhex20	Priority related functions, 362
Kernel Debugger, 187	process
outhex32	L4Re::Util::Event_buffer_consumer_t, 630
Kernel Debugger, 187	Producer, 422, 432
outhex8	l4shmc chunk ready, 422
Kernel Debugger, 187	l4shmc_chunk_ready_sig, 423
outnstring	I4shmc_chunk_try_to_take, 422
Kernel Debugger, 186	l4shmc_is_chunk_clear, 423
outstring	l4shmc_trigger, 432
Kernel Debugger, 186	Protocols
Nemer Debugger, 100	
PT_GNU_EH_FRAME	L4Re Protocol identifiers, 113
ELF binary format, 354	provider_pid
PT_GNU_RELRO	I4_vhw_entry, 761
ELF binary format, 355	push_back
PT_GNU_STACK	cxx::List, 792
	cxx::List_item, 797
ELF binary format, 355	push_front
PT_HIOS	cxx::List, 792
ELF binary format, 354	cxx::List_item, 797
PT_HIPROC	put
ELF binary format, 354	L4::lpc::Ostream, 833, 834
PT_L4_AUX	L4Re::Event_buffer_t, 634
ELF binary format, 355	
PT_L4_KIP	query
ELF binary format, 355	L4Re::Namespace, 823
PT_L4_STACK	query log name
ELF binary format, 355	L4::Debugger, 586
PT LOOS	query_log_typeid
ELF binary format, 354	L4::Debugger, 585
PT LOPROC	21Dobaggor, 000
ELF binary format, 354	R
page_shift	cxx::Bits::Direction, 592
L4Re::Util::Dataspace_svr, 578	r
. —	L4Re::Video::Pixel info, 845, 846
Pager	- · · · ·
L4Re::Rm, 858	L4vcpu::Vcpu, 944, 945
pager	Random number support, 363
L4::Thread::Attr, 451	l4util_rand, 363
Pair	l4util_srand, 363
cxx::Pair, 839	rbegin
Pair_first_compare	cxx::AvI_set, 474
cxx::Pair_first_compare, 840	cxx::Bits::Bst, 525, 526
Parent	read
L4Re Protocol identifiers, 114	L4::Vcon, 933
parent	Read_only
L4Re::Env, 617, 620	 L4Re::Rm, 858
	•

readv	cxx::Avl_map, 464
L4Re::Vfs::Regular_file, 853	cxx::Avl_set, 471
Realtime API, 223	cxx::Avl_tree, 478
receive	cxx::List, 792
L4::lpc::lstream, 717	cxx::List_item, 797
L4::Irq, 707	remove_me
Ref	cxx::List_item, 796
cxx::Bitfield, 500	cxx::List_item::Iter, 720
Ref_type	rename
cxx::Auto_ptr, 458	L4Re::Vfs::Directory, 595
Ref_unshifted	rend
cxx::Bitfield, 501	cxx::Avl_set, 474
refresh	cxx::Bits::Bst, 525, 526
L4Re::Util::Video::Goos_svr, 672	Reply_compound
L4Re::Video::View, 950	L4::lpc_svr, 446
Region map API, 115	Reply_separate
Region map interface, 77	L4::lpc_svr, 446
L4RE_RM_ATTACH_FLAGS, 78	reply_and_wait
L4RE_RM_EAGER_MAP, 78	L4::lpc::lostream, 698
L4RE_RM_IN_AREA, 78	Reply_mode
L4RE RM NO ALIAS, 78	L4::lpc_svr, 446
L4RE RM OVERMAP, 78	reserve_area
L4RE RM PAGER, 78	L4Re::Rm, 858, 859
L4RE RM READ ONLY, 78	Reserved
L4RE_RM_REGION_FLAGS, 78	L4Re::Rm, 858
L4RE_RM_RESERVED, 78	reset
L4RE RM SEARCH ADDR, 78	L4::lpc::lostream, 696
l4re rm attach, 79	L4::lpc::lstream, 714
l4re_rm_attach_srv, 84	Rm
l4re_rm_detach, 80	L4Re Protocol identifiers, 114
l4re rm detach ds, 81	rm
l4re_rm_detach_ds_unmap, 82	L4Re::Env, 617, 621
l4re_rm_detach_srv, 85	rmdir
l4re rm detach unmap, 81	L4Re::Vfs::Directory, 596
l4re rm find, 83	Ro
l4re rm find srv, 85	L4Re::Namespace, 823
l4re_rm_flags_t, 78	run_thread
l4re rm free area, 78	L4::Scheduler, 875
l4re rm free area srv, 84	Rw
l4re rm reserve area, 78	L4Re::Namespace, 823
l4re_rm_reserve_area_srv, 84	6
l4re_rm_show_lists, 83	S LAUFactorius 000
Region flags	L4::Factory::S, 869
L4Re::Rm, 858	SHF_GROUP
Region_flags	ELF binary format, 354
L4Re::Rm, 858	SHF_MASKOS
register_del_irq	ELF binary format, 354
L4::Thread, 918	SHF_TLS
Register_flags	ELF binary format, 354
L4Re::Namespace, 823	SHT_NUM
·	ELF binary format, 354
register_obj	saved_state
L4Re::Namespace, 825	L4vcpu::Vcpu, 941, 942
release	scan_zero
cxx::Auto_ptr, 459	cxx::Bitmap_base, 515
L4Re::Util::Dataspace_svr, 577	Scheduler, 241
release_cap	L4_SCHEDULER_IDLE_TIME_OP, 242
L4::Task, 906	L4_SCHEDULER_INFO_OP, 242
remove	L4_SCHEDULER_RUN_THREAD_OP, 242

I4_sched_cpu_set, 242	cxx::Bitfield, 500
l4_scheduler_idle_time, 244	si
l4 scheduler info, 242	I4_vcpu_regs_t, 753
l4_scheduler_is_online, 244	Sigma0 API
L4_scheduler_ops, 242	L4SIGMA0 IPCERROR, 400
l4_scheduler_run_thread, 243	L4SIGMA0_NOFPAGE, 400
scheduler	L4SIGMA0_NOTALIGNED, 400
L4Re::Env, 622	L4SIGMA0 OK, 400
screen_info	L4SIGMA0_SMALLERFPAGE, 400
L4Re::Util::Video::Goos svr, 671	Sigma0 API, 399
Search_addr	l4sigma0_debug_dump, 402
L4Re::Rm, 858	l4sigma0_map_anypage, 401
send	I4sigma0_map_errstr, 402
L4::lpc::Ostream, 835	I4sigma0_map_iomem, 400
L4::Vcon, 931	I4sigma0_map_kip, 400
Server	I4sigma0_map_mem, 400
L4::Server, 879	I4sigma0_map_tbuf, 401
set	l4sigma0_new_client, 402
cxx::Bitfield, 504	l4sigma0_return_flags_t, 400
L4::Kip::Mem_desc, 811	signal
L4Re::Video::Color_component, 556	L4Re::Parent, 843
L4vcpu::State, 897	Signals, 428
set attr	l4shmc add signal, 428
	l4shmc attach signal, 429
set_bit	l4shmc_attach_signal_to, 429
cxx::Bitmap_base, 514	I4shmc_check_magic, 431
set_dirty	l4shmc_get_signal_to, 429
cxx::Bitfield, 502	l4shmc_signal_cap, 431
set_fd	size
L4Re::Vfs::Fs, 659	cxx::List, 792
set_info	L4::lpc::Buf_cp_out, 535
L4Re::Video::View, 950	L4::Kip::Mem_desc, 810
set_lock	L4Re::Dataspace, 572
L4Re::Vfs::Regular_file, 854	L4Re::Video::Color_component, 556
set_mode	size0
L4::lcu, 681	I4_tracebuffer_status_t, 748
set_object_name	· ·
- , -	size1
L4::Debugger, 584	size1 I4_tracebuffer_status_t, 748
L4::Debugger, 584 set_status_flags	
L4::Debugger, 584	I4_tracebuffer_status_t, 748
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485
L4::Debugger, 584 set_status_flags	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489
L4::Debugger, 584 set_status_flags	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport L4Re::Video::View, 950	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44 min, 44
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport L4Re::Video::View, 950 Shared Memory Library, 414	I4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44 min, 44 operator new, 44
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport L4Re::Video::View, 950 Shared Memory Library, 414 l4shmc_area_overhead, 416	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44 min, 44 operator new, 44 Smart_cap
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport L4Re::Video::View, 950 Shared Memory Library, 414 I4shmc_area_overhead, 416 I4shmc_area_size, 416	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44 min, 44 operator new, 44 Smart_cap L4::Smart_cap, 891
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport L4Re::Video::View, 950 Shared Memory Library, 414 l4shmc_area_overhead, 416 l4shmc_area_size, 416 l4shmc_area_size_free, 416	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44 min, 44 operator new, 44 Smart_cap L4::Smart_cap, 891 snd_base
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport L4Re::Video::View, 950 Shared Memory Library, 414 l4shmc_area_overhead, 416 l4shmc_area_size, 416 l4shmc_area_size_free, 416 l4shmc_attach, 415	I4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44 min, 44 operator new, 44 Smart_cap L4::Smart_cap, 891 snd_base L4::Cap_base, 551
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport L4Re::Video::View, 950 Shared Memory Library, 414 I4shmc_area_overhead, 416 I4shmc_area_size, 416 I4shmc_area_size_free, 416 I4shmc_attach, 415 I4shmc_attach_to, 415	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44 min, 44 operator new, 44 Smart_cap L4::Smart_cap, 891 snd_base L4::Cap_base, 551 Split_ds
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport L4Re::Video::View, 950 Shared Memory Library, 414 l4shmc_area_overhead, 416 l4shmc_area_size, 416 l4shmc_area_size_free, 416 l4shmc_attach, 415 l4shmc_attach_to, 415 l4shmc_chunk_overhead, 417	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44 min, 44 operator new, 44 Smart_cap L4::Smart_cap, 891 snd_base L4::Cap_base, 551 Split_ds L4Re::Rm, 858
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport L4Re::Video::View, 950 Shared Memory Library, 414 l4shmc_area_overhead, 416 l4shmc_area_size, 416 l4shmc_area_size_free, 416 l4shmc_attach, 415 l4shmc_attach, 415 l4shmc_connect_chunk_signal, 416	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44 min, 44 operator new, 44 Smart_cap L4::Smart_cap, 891 snd_base L4::Cap_base, 551 Split_ds L4Re::Rm, 858 stack
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport L4Re::Video::View, 950 Shared Memory Library, 414 l4shmc_area_overhead, 416 l4shmc_area_size, 416 l4shmc_area_size_free, 416 l4shmc_attach, 415 l4shmc_attach, 415 l4shmc_chunk_overhead, 417 l4shmc_connect_chunk_signal, 416 l4shmc_create, 415	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44 min, 44 operator new, 44 Smart_cap L4::Smart_cap, 891 snd_base L4::Cap_base, 551 Split_ds L4Re::Rm, 858 stack L4Re::Video::View, 950
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport L4Re::Video::View, 950 Shared Memory Library, 414 l4shmc_area_overhead, 416 l4shmc_area_size, 416 l4shmc_area_size_free, 416 l4shmc_attach, 415 l4shmc_attach, 415 l4shmc_chunk_overhead, 417 l4shmc_connect_chunk_signal, 416 l4shmc_create, 415 shift	I4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44 min, 44 operator new, 44 Smart_cap L4::Smart_cap, 891 snd_base L4::Cap_base, 551 Split_ds L4Re::Rm, 858 stack L4Re::Video::View, 950 start
L4::Debugger, 584 set_status_flags L4Re::Vfs::Generic_file, 663 set_unshifted cxx::Bitfield, 504 set_unshifted_dirty cxx::Bitfield, 503 set_viewport L4Re::Video::View, 950 Shared Memory Library, 414 l4shmc_area_overhead, 416 l4shmc_area_size, 416 l4shmc_area_size_free, 416 l4shmc_attach, 415 l4shmc_attach, 415 l4shmc_chunk_overhead, 417 l4shmc_connect_chunk_signal, 416 l4shmc_create, 415	l4_tracebuffer_status_t, 748 skip L4::lpc::lstream, 715 slab_size cxx::Base_slab, 485 cxx::Base_slab_static, 489 Small C++ Template Library, 43 max, 44 min, 44 operator new, 44 Smart_cap L4::Smart_cap, 891 snd_base L4::Cap_base, 551 Split_ds L4Re::Rm, 858 stack L4Re::Video::View, 950

L4vcpu::State, 896 state	L4_THREAD_CONTROL_SET_EXC_HANDLER, 257
L4vcpu::Vcpu, 941	L4 THREAD CONTROL SET PAGER, 257
stats_time	L4_THREAD_CONTROL_UX_NATIVE, 257
L4::Thread, 916	L4_THREAD_EX_REGS_CANCEL, 258
Strong	L4_THREAD_EX_REGS_OP, 257
L4Re::Namespace, 823	L4_THREAD_EX_REGS_TRIGGER_EXCEPTIO-
sub_type	N, 258
L4::Kip::Mem_desc, 811	L4_THREAD_MODIFY_SENDER_OP, 257
Super_pages	L4 THREAD OPCODE MASK, 257
L4Re::Mem_alloc, 805	L4_THREAD_REGISTER_DELETE_IRQ_OP, 257
supports	L4 THREAD STATS OP, 257
L4::Meta, 815	L4 THREAD SWITCH OP, 257
switch_log	L4_THREAD_VCPU_CONTROL_OP, 257
L4::Debugger, 586	L4_THREAD_VCPU_RESUME_OP, 257
switch_to	L4_THREAD_X86_GDT_OP, 257
L4::Thread, 915	I4_thread_arm_set_tpidruro, 266
symlink	L4_thread_control_flags, 257
L4Re::Vfs::Directory, 596	L4_thread_control_mr_indices, 257
, ,	14_thread_ex_regs, 258
tag	L4_thread_ex_regs_flags, 258
L4::lpc::lstream, 715, 716	l4_thread_ex_regs_ret, 259
L4::lpc::Ostream, 834	l4_thread_modify_sender_add, 264
take	l4_thread_modify_sender_commit, 265
L4Re::Util::Dataspace_svr, 577	14_thread_modify_sender_start, 264
Task, 246	L4_thread_ops, 257
L4_FP_ALL_SPACES, 247	l4_thread_register_del_irq, 263
L4_FP_DELETE_OBJ, 247	l4_thread_stats_time, 260
L4_FP_OTHER_SPACES, 247	I4_thread_switch, 260
I4_task_add_ku_mem, 252	
I4_task_cap_equal, 252	l4_thread_vcpu_control, 262
I4_task_cap_has_child, 251	I4_thread_vcpu_control_ext, 263
I4_task_cap_valid, 251	I4_thread_vcpu_resume_commit, 261
I4_task_delete_obj, 250	l4_thread_vcpu_resume_start, 261
I4_task_map, 247	I4_thread_yield, 259
l4_task_release_cap, 250	Thread control, 267
I4_task_unmap, 248	I4_thread_control_alien, 271
I4_task_unmap_batch, 249	I4_thread_control_bind, 269
l4_unmap_flags_t, 247	I4_thread_control_commit, 272
task	l4_thread_control_exc_handler, 269
L4Re::Env, 618	I4_thread_control_pager, 268
L4vcpu::Vcpu, 943	I4_thread_control_start, 268
Thread, 255	I4_thread_control_ux_host_syscall, 272
L4_THREAD_AMD64_SET_SEGMENT_BASE	Thread Control Registers (TCRs), 297
OP, 257	Timeouts, 153
L4_THREAD_ARM_TPIDRURO_OP, 257	L4_IPC_TIMEOUT_0, 154
L4_THREAD_CONTROL_ALIEN, 257	I4_ipc_timeout, 155
L4_THREAD_CONTROL_BIND_TASK, 257	I4_rcv_timeout, 157
L4_THREAD_CONTROL_MR_IDX_BIND_TASK,	I4_snd_timeout, 155
258	I4_timeout, 155
L4_THREAD_CONTROL_MR_IDX_BIND_UTCB,	I4_timeout_abs, 158
258	I4_timeout_abs_validity, 155
L4_THREAD_CONTROL_MR_IDX_EXC_HANDL-	I4_timeout_get, 158
ER, 258	I4_timeout_is_absolute, 157
L4_THREAD_CONTROL_MR_IDX_FLAG_VALS,	I4_timeout_rel, 155
258	I4_timeout_rel_get, 157
L4_THREAD_CONTROL_MR_IDX_FLAGS, 258	I4_timeout_s, 154
L4_THREAD_CONTROL_MR_IDX_PAGER, 258	I4_timeout_t, 154
L4_THREAD_CONTROL_OP, 257	I4_utcb_mr64_idx, 160

Time and the second of the sec	LA VODIL OFFICET EVT INFOC 000
Timestamp Counter, 319	L4_VCPU_OFFSET_EXT_INFOS, 306
I4_busy_wait_ns, 323	L4_VCPU_OFFSET_EXT_STATE, 306
I4_busy_wait_us, 324	L4_VCPU_SF_IRQ_PENDING, 306
I4_calibrate_tsc, 324	vCPU Support Library
l4_get_hz, 325	L4VCPU_IRQ_STATE_DISABLED, 406
I4_ns_to_tsc, 323	L4VCPU_IRQ_STATE_ENABLED, 406
I4_rdpmc, 320	VM API for VMX
I4_rdpmc_32, 322	L4_VM_VMX_BASIC_REG, 163
I4_rdtsc, 320	L4 VM VMX CR0 FIXED0 REG, 163
l4 rdtsc 32, 320	L4 VM VMX CR0 FIXED1 REG, 163
14 tsc init, 325	L4 VM VMX CR4 FIXED0 REG, 163
I4_tsc_to_ns, 322	L4_VM_VMX_CR4_FIXED1_REG, 163
I4_tsc_to_s_and_ns, 322	L4 VM VMX ENTRY CTLS DFL1 REG, 164
14_tsc_to_us, 322	
total_objects	L4_VM_VMX_EPT_VPID_CAP_REG, 163
-	L4_VM_VMX_EXIT_CTLS_DFL1_REG, 164
cxx::Base_slab, 485	L4_VM_VMX_MISC_REG, 163
cxx::Base_slab_static, 489	L4_VM_VMX_NUM_CAPS_REGS, 163
tracebuffer0	L4_VM_VMX_NUM_DFL1_REGS, 164
I4_tracebuffer_status_t, 748	L4_VM_VMX_PINBASED_CTLS_DFL1_REG, 164
tracebuffer1	L4 VM VMX PROCBASED CTLS2 REG, 163
I4_tracebuffer_status_t, 748	L4_VM_VMX_PROCBASED_CTLS_DFL1_REG,
trigger	164
L4::Irq, 709	L4_VM_VMX_TRUE_ENTRY_CTLS_REG, 163
type	L4_VM_VMX_TRUE_EXIT_CTLS_REG, 163
L4::Kip::Mem_desc, 811	L4_VM_VMX_TRUE_PINBASED_CTLS_REG,
I4_vhw_entry, 760	
L4Re::Vfs::Be_file_system, 498	163
L4Re::Vfs::File_system, 655	L4_VM_VMX_TRUE_PROCBASED_CTLS_REG,
L411evisi iie_systeiii, 000	163
unbind	L4_VM_VMX_VMCS_CR2, 164
L4::lcu, 677	L4_VM_VMX_VMCS_ENUM_REG, 163
unlink	V_flags
	L4Re::Video::View, 949
L4Re::Vfs::Directory, 595	vCPU API, 305
unlock_all_locks	L4_vcpu_state_flags, 306
L4Re::Vfs::Be_file, 495	L4_vcpu_state_offset, 306
L4Re::Vfs::Generic_file, 662	L4 vcpu sticky flags, 306
unmap	vCPU Support Library, 405
L4::Task, 904	• • • • • • • • • • • • • • • • • • • •
unmap_batch	l4vcpu_irq_disable, 406
L4::Task, 905	l4vcpu_irq_disable_save, 407
unmask	I4vcpu_irq_enable, 408
L4::lcu, 681	I4vcpu_irq_restore, 409
L4::lrq, 709	l4vcpu_irq_state_t, 406
utcb_area	I4vcpu_is_irq_entry, 411
L4Re::Env, 619, 622	I4vcpu_is_page_fault_entry, 411
Utility Functions, 366	l4vcpu_print_state, 410
	I4vcpu_state, 406
4util_micros2 4to, 369	I4vcpu wait for event, 410
l4util_splitlog2_hdl, 367	VM API for SVM, 161
l4util_splitlog2_size, 368	VM API for TZ, 312
ux_host_syscall	
L4::Thread::Attr, 453	VM API for VMX, 162
	L4_vm_vmx_caps_regs, 163
vCPU API	I4_vm_vmx_clear, 166
L4_VCPU_F_DEBUG_EXC, 306	L4_vm_vmx_dfl1_regs, 163
L4_VCPU_F_EXCEPTIONS, 306	I4_vm_vmx_field_len, 165
L4_VCPU_F_FPU_ENABLED, 306	I4_vm_vmx_field_order, 165
L4_VCPU_F_IRQ, 306	I4_vm_vmx_get_caps, 164
L4_VCPU_F_PAGE_FAULTS, 306	I4_vm_vmx_get_caps_default1, 164
L4_VCPU_F_USER_MODE, 306	I4_vm_vmx_get_cr2_index, 167
/	′

I4_vm_vmx_ptr_load, 166	l4re_video_goos_create_buffer, 92
L4_vm_vmx_read, 169	l4re_video_goos_create_view, 92
l4_vm_vmx_read_16, 168	l4re_video_goos_delete_buffer, 92
l4_vm_vmx_read_32, 168	l4re_video_goos_delete_view, 93
l4_vm_vmx_read_64, 169	l4re_video_goos_get_static_buffer, 92
I4_vm_vmx_read_nat, 167	l4re_video_goos_get_view, 93
I4_vm_vmx_write, 172	l4re_video_goos_info, 91
I4_vm_vmx_write_16, 171	l4re_video_goos_info_flags_t, 90
I4_vm_vmx_write_32, 171	l4re_video_goos_refresh, 91
I4_vm_vmx_write_64, 171	l4re_video_view_get_info, 93
I4_vm_vmx_write_nat, 170	l4re_video_view_info_flags_t, 91
Val	l4re_video_view_refresh, 93
cxx::Bitfield, 501	l4re_video_view_set_info, 93
val	l4re_video_view_set_viewport, 95
cxx::Bitfield, 506	l4re_video_view_stack, 95
val_dirty	l4re_video_view_t, 90
cxx::Bitfield, 506	view
Val unshifted	L4Re::Video::Goos, 669
cxx::Bitfield, 501	view_info
val_unshifted	L4Re::Util::Video::Goos_svr, 671
cxx::Bitfield, 507	Virtual Console, 300
valid	L4_VCON_ECHO, 301
cxx::Avl set::Node, 828	L4_VCON_GET_ATTR_OP, 302
validate	L4_VCON_ICANON, 301
L4::Cap_base, 552	L4_VCON_ICRNL, 301
Value	L4_VCON_IGNCR, 301
	L4_VCON_INLCR, 301
L4::Basic_registry, 491	L4_VCON_OCRNL, 301
vcpu_control	L4 VCON ONLCR, 301
L4::Thread, 917	L4 VCON ONLRET, 301
vcpu_control_ext	L4_VCON_SET_ATTR_OP, 302
L4::Thread, 917	L4_VCON_WRITE_OP, 302
vcpu_resume_commit	L4_VCON_WRITE_SIZE, 301
L4::Thread, 916	I4 vcon get attr, 304
vcpu_resume_start	L4_vcon_i_flags, 301
L4::Thread, 916	L4 vcon I flags, 301
version	L4_vcon_o_flags, 301
I4_vhw_descriptor, 759	L4_vcon_ops, 302
version0	I4_vcon_read, 303
I4_tracebuffer_status_t, 748	14_vcon_send, 302
version1	I4_vcon_set_attr, 303
l4_tracebuffer_status_t, 748	I4_vcon_write, 302
Video API	L4 vcon write consts, 301
F_l4re_video_goos_auto_refresh, 91	Virtual Machines, 197
F_l4re_video_goos_dynamic_buffers, 91	Virtual Registers (UTCBs), 291
F_l4re_video_goos_dynamic_views, 91	I4_utcb_br, 293
F_l4re_video_goos_pointer, 91	I4_utcb_mr, 292
F_l4re_video_view_above, 91	I4_utcb_t, 292
F_l4re_video_view_dyn_allocated, 91	14_utcb_tcr, 293
F_l4re_video_view_flags_mask, 91	
F_l4re_video_view_none, 91	wait
F_l4re_video_view_set_background, 91	L4::lpc::lstream, 716, 717
F_I4re_video_view_set_buffer, 91	L4::lrq, 708
F_I4re_video_view_set_buffer_offset, 91	wait_for_event
F_l4re_video_view_set_bytes_per_line, 91	L4vcpu::Vcpu, 943
F_l4re_video_view_set_flags, 91	words
F_l4re_video_view_set_pixel, 91	cxx::Bitmap_base, 513
F_l4re_video_view_set_position, 91	write
Video API, 89	L4::Vcon, 931

```
writev
L4Re::Vfs::Regular_file, 853

x86 Virtual Registers (UTCB)
L4_UTCB_BUF_REGS_OFFSET, 314
L4_UTCB_EXCEPTION_REGS_SIZE, 314
L4_UTCB_GENERIC_BUFFERS_SIZE, 314
L4_UTCB_GENERIC_DATA_SIZE, 314
L4_UTCB_INHERIT_FPU, 314
L4_UTCB_MSG_REGS_OFFSET, 314
L4_UTCB_OFFSET, 314
L4_UTCB_THREAD_REGS_OFFSET, 314
x86 Virtual Registers (UTCB), 314
L4_utcb_consts_x86, 314
```