

MPX Thunder Krakens
R6

Generated by Doxygen 1.8.6

Thu Apr 28 2016 05:34:02

Contents

1	Main Page	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	cmcb Struct Reference	7
4.1.1	Detailed Description	7
4.1.2	Field Documentation	7
4.1.2.1	begin_address	7
4.1.2.2	size	7
4.1.2.3	type	7
4.2	context Struct Reference	8
4.2.1	Detailed Description	8
4.2.2	Field Documentation	9
4.2.2.1	cs	9
4.2.2.2	ds	9
4.2.2.3	eax	9
4.2.2.4	ebp	9
4.2.2.5	ebx	9
4.2.2.6	ecx	9
4.2.2.7	edi	9
4.2.2.8	edx	9
4.2.2.9	eflags	9
4.2.2.10	eip	9
4.2.2.11	es	9

4.2.2.12	esi	10
4.2.2.13	esp	10
4.2.2.14	fs	10
4.2.2.15	gs	10
4.3	data_sector Struct Reference	10
4.3.1	Detailed Description	10
4.3.2	Field Documentation	10
4.3.2.1	data	10
4.4	date_time Struct Reference	11
4.4.1	Field Documentation	11
4.4.1.1	day_m	11
4.4.1.2	day_w	11
4.4.1.3	day_y	11
4.4.1.4	hour	11
4.4.1.5	min	11
4.4.1.6	mon	11
4.4.1.7	sec	11
4.4.1.8	year	11
4.5	dir_entry_info Struct Reference	11
4.5.1	Detailed Description	12
4.5.2	Field Documentation	12
4.5.2.1	attributes	12
4.5.2.2	create_date	12
4.5.2.3	create_time	12
4.5.2.4	extension	13
4.5.2.5	file_name	13
4.5.2.6	file_size	13
4.5.2.7	first_log_clu	13
4.5.2.8	ignore1	13
4.5.2.9	last_acc_date	13
4.5.2.10	last_wri_date	13
4.5.2.11	last_wri_time	13
4.5.2.12	reserved	13
4.6	dir_itr Struct Reference	14
4.7	fat_date Struct Reference	14
4.7.1	Detailed Description	14
4.7.2	Field Documentation	14

4.7.2.1	day	14
4.7.2.2	mon	14
4.7.2.3	year	14
4.8	fat_time Struct Reference	15
4.8.1	Detailed Description	15
4.8.2	Field Documentation	15
4.8.2.1	hr	15
4.8.2.2	mi	15
4.8.2.3	se	15
4.9	file_iter Struct Reference	15
4.10	footer Struct Reference	16
4.10.1	Field Documentation	16
4.10.1.1	head	16
4.11	gdt_descriptor Struct Reference	16
4.11.1	Field Documentation	16
4.11.1.1	base	16
4.11.1.2	limit	17
4.12	gdt_entry Struct Reference	17
4.12.1	Field Documentation	17
4.12.1.1	access	17
4.12.1.2	base_high	17
4.12.1.3	base_low	17
4.12.1.4	base_mid	17
4.12.1.5	flags	17
4.12.1.6	limit_low	17
4.13	header Struct Reference	17
4.13.1	Field Documentation	18
4.13.1.1	index_id	18
4.13.1.2	size	18
4.14	heap Struct Reference	18
4.14.1	Field Documentation	18
4.14.1.1	base	18
4.14.1.2	index	18
4.14.1.3	max_size	19
4.14.1.4	min_size	19
4.15	idt_descriptor Struct Reference	19
4.15.1	Field Documentation	19

4.15.1.1	base	19
4.15.1.2	limit	19
4.16	idt_entry Struct Reference	19
4.16.1	Field Documentation	19
4.16.1.1	base_high	19
4.16.1.2	base_low	19
4.16.1.3	flags	19
4.16.1.4	sselect	20
4.16.1.5	zero	20
4.17	img_boot_sector Struct Reference	20
4.17.1	Detailed Description	21
4.17.2	Field Documentation	21
4.17.2.1	boot_sign	21
4.17.2.2	byte_per_sector	21
4.17.2.3	fat_copies_num	21
4.17.2.4	file_sys_type	21
4.17.2.5	head_num	21
4.17.2.6	ignore1	21
4.17.2.7	ignore2	22
4.17.2.8	ignore3	22
4.17.2.9	ignore4	22
4.17.2.10	ignore5	22
4.17.2.11	reserved_sec_num	22
4.17.2.12	root_dir_max_num	22
4.17.2.13	sec_num	22
4.17.2.14	sec_per_fat_num	22
4.17.2.15	sec_per_track	22
4.17.2.16	sector_per_cluster	23
4.17.2.17	total_sec_fat32	23
4.17.2.18	vol_id	23
4.17.2.19	vol_label	23
4.18	img_writer Struct Reference	23
4.19	index_entry Struct Reference	23
4.19.1	Field Documentation	23
4.19.1.1	block	23
4.19.1.2	empty	24
4.19.1.3	size	24

4.20	index_table Struct Reference	24
4.20.1	Field Documentation	24
4.20.1.1	id	24
4.20.1.2	table	24
4.21	Imcb Struct Reference	24
4.21.1	Detailed Description	25
4.21.2	Field Documentation	25
4.21.2.1	size	25
4.21.2.2	type	25
4.22	mcb Struct Reference	25
4.22.1	Detailed Description	26
4.22.2	Field Documentation	26
4.22.2.1	complete_mcb	26
4.22.2.2	limited_mcb	26
4.22.2.3	next	26
4.22.2.4	prev	26
4.23	page_dir Struct Reference	26
4.23.1	Field Documentation	27
4.23.1.1	tables	27
4.23.1.2	tables_phys	27
4.24	page_entry Struct Reference	27
4.24.1	Field Documentation	28
4.24.1.1	accessed	28
4.24.1.2	dirty	28
4.24.1.3	frameaddr	28
4.24.1.4	present	28
4.24.1.5	reserved	28
4.24.1.6	usermode	28
4.24.1.7	writable	28
4.25	page_table Struct Reference	28
4.25.1	Field Documentation	29
4.25.1.1	pages	29
4.26	param Struct Reference	29
4.26.1	Detailed Description	29
4.26.2	Field Documentation	29
4.26.2.1	device_id	29
4.26.2.2	op_code	29

4.27	pcb_queue Struct Reference	29
4.27.1	Detailed Description	30
4.27.2	Field Documentation	30
4.27.2.1	count	30
4.27.2.2	head	30
4.27.2.3	tail	30
4.28	pcb_struct Struct Reference	31
4.28.1	Detailed Description	31
4.28.2	Field Documentation	31
4.28.2.1	class	31
4.28.2.2	is_suspended	32
4.28.2.3	name	32
4.28.2.4	next	32
4.28.2.5	prev	32
4.28.2.6	priority	32
4.28.2.7	running_state	32
4.28.2.8	stack_base	32
4.28.2.9	stack_top	32
5	File Documentation	33
5.1	documentation/mainpage.dox File Reference	33
5.2	include/core/asm.h File Reference	33
5.3	include/core/interrupts.h File Reference	33
5.3.1	Function Documentation	34
5.3.1.1	init_irq	34
5.3.1.2	init_pic	34
5.4	include/core/io.h File Reference	34
5.4.1	Macro Definition Documentation	34
5.4.1.1	inb	34
5.4.1.2	outb	34
5.5	include/core/serial.h File Reference	34
5.5.1	Detailed Description	36
5.5.2	Macro Definition Documentation	36
5.5.2.1	COM1	36
5.5.2.2	COM2	36
5.5.2.3	COM3	36
5.5.2.4	COM4	36

5.5.2.5	USER_INPUT_BUFFER_SIZE	36
5.5.2.6	WithEcho	36
5.5.2.7	WithoutEcho	36
5.5.3	Function Documentation	36
5.5.3.1	get_input_line	36
5.5.3.2	init_serial	36
5.5.3.3	serial_print	36
5.5.3.4	serial_println	37
5.5.3.5	set_serial_in	37
5.5.3.6	set_serial_out	37
5.6	include/core/tables.h File Reference	37
5.6.1	Function Documentation	38
5.6.1.1	gdt_init_entry	38
5.6.1.2	idt_set_gate	38
5.6.1.3	init_gdt	38
5.6.1.4	init_idt	38
5.7	include/mem/heap.h File Reference	38
5.7.1	Macro Definition Documentation	39
5.7.1.1	KHEAP_BASE	39
5.7.1.2	KHEAP_MIN	39
5.7.1.3	KHEAP_SIZE	39
5.7.1.4	TABLE_SIZE	39
5.7.2	Function Documentation	39
5.7.2.1	_kmalloc	39
5.7.2.2	alloc	39
5.7.2.3	init_kheap	39
5.7.2.4	kfree	39
5.7.2.5	kmalloc	39
5.7.2.6	make_heap	39
5.8	include/mem/paging.h File Reference	39
5.8.1	Macro Definition Documentation	40
5.8.1.1	PAGE_SIZE	40
5.8.2	Function Documentation	40
5.8.2.1	clear_bit	40
5.8.2.2	first_free	40
5.8.2.3	get_bit	40
5.8.2.4	get_page	40

5.8.2.5	init_paging	40
5.8.2.6	load_page_dir	40
5.8.2.7	new_frame	40
5.8.2.8	set_bit	40
5.9	include/string.h File Reference	40
5.9.1	Detailed Description	44
5.9.2	Function Documentation	44
5.9.2.1	atoi	44
5.9.2.2	isspace	44
5.9.2.3	memset	44
5.9.2.4	printf	44
5.9.2.5	sprintf	45
5.9.2.6	strcat	45
5.9.2.7	strcmp	45
5.9.2.8	strcpy	45
5.9.2.9	strlen	45
5.9.2.10	strtok	45
5.10	include/system.h File Reference	45
5.10.1	Macro Definition Documentation	46
5.10.1.1	asm	46
5.10.1.2	cli	46
5.10.1.3	GDT_CS_ID	46
5.10.1.4	GDT_DS_ID	46
5.10.1.5	hlt	46
5.10.1.6	iret	46
5.10.1.7	no_warn	46
5.10.1.8	nop	46
5.10.1.9	NULL	46
5.10.1.10	sti	46
5.10.1.11	volatile	46
5.10.2	Typedef Documentation	46
5.10.2.1	size_t	46
5.10.2.2	u16int	46
5.10.2.3	u32int	46
5.10.2.4	u8int	46
5.10.3	Function Documentation	46
5.10.3.1	irq_on	46

5.10.3.2	klogv	46
5.10.3.3	kpanic	46
5.11	modules/cmd_orders.h File Reference	46
5.11.1	Detailed Description	48
5.11.2	Macro Definition Documentation	48
5.11.2.1	FUNCTIONS_BEGIN	48
5.11.2.2	GETDATE	48
5.11.2.3	GETTIME	48
5.11.2.4	HELP	48
5.11.2.5	LOADR3	48
5.11.2.6	MCB_FUNC_END	48
5.11.2.7	MCB_FUNCTIONS_BEGIN	48
5.11.2.8	MPX_FUNC_END	48
5.11.2.9	MPX_FUNCTIONS_BEGIN	48
5.11.2.10	NUM_OF_FUNCTIONS	48
5.11.2.11	PCB_FUNC_END	48
5.11.2.12	PCB_FUNCTIONS_BEGIN	48
5.11.2.13	RESUMEPCB	48
5.11.2.14	SETDATE	48
5.11.2.15	SETPCBPRIO	48
5.11.2.16	SETTIME	48
5.11.2.17	SHOWMCB	48
5.11.2.18	SHOWPCB	49
5.11.2.19	SHUTDOWN	49
5.11.2.20	SUSPDPCB	49
5.11.2.21	VERSION	49
5.11.2.22	WITH_R2_TEMP_CMD	49
5.11.2.23	WITH_R3_TEMP_CMD	49
5.11.2.24	WITH_R5_TEMP_CMD	49
5.12	modules/errno.h File Reference	49
5.12.1	Detailed Description	50
5.12.2	Macro Definition Documentation	50
5.12.2.1	E_EMPTPCB	50
5.12.2.2	E_FILE_NF	50
5.12.2.3	E_FOLDFUL	51
5.12.2.4	E_FREEMEM	51
5.12.2.5	E_INVATTRS	51

5.12.2.6	E_INVPARA	51
5.12.2.7	E_INVSTRF	51
5.12.2.8	E_INVUSRI	51
5.12.2.9	E_NAMEDUP	51
5.12.2.10	E_NAMEINV	51
5.12.2.11	E_NOERROR	51
5.12.2.12	E_NOSPACE	51
5.12.2.13	E_NULL_PTR	51
5.12.2.14	E_PCB_SYS	51
5.12.2.15	E_PROGERR	51
5.12.3	Typedef Documentation	51
5.12.3.1	error_t	51
5.13	modules/mpx_supt.h File Reference	51
5.13.1	Detailed Description	54
5.13.2	Macro Definition Documentation	54
5.13.2.1	EXIT	54
5.13.2.2	IDLE	54
5.13.2.3	MODULE_R1	54
5.13.2.4	MODULE_R2	54
5.13.2.5	MODULE_R3	54
5.13.2.6	MODULE_R4	54
5.13.2.7	MODULE_R5	54
5.13.2.8	READ	54
5.13.2.9	WRITE	54
5.13.3	Function Documentation	54
5.13.3.1	get_op_code	54
5.13.3.2	idle	54
5.13.3.3	mpx_init	54
5.13.3.4	sys_alloc_mem	54
5.13.3.5	sys_free_mem	55
5.13.3.6	sys_req	55
5.13.3.7	sys_set_free	55
5.13.3.8	sys_set_malloc	55
5.14	modules/packing.h File Reference	55
5.14.1	Detailed Description	55
5.14.2	Macro Definition Documentation	56
5.14.2.1	PACKED	56

5.15	modules/r1/r1.h File Reference	56
5.15.1	Detailed Description	57
5.15.2	Enumeration Type Documentation	57
5.15.2.1	comm_type	57
5.15.3	Function Documentation	58
5.15.3.1	command_line_parser	58
5.15.3.2	commhand	58
5.15.3.3	help_usages	58
5.15.3.4	print_help	58
5.16	modules/r1/sys_clock.h File Reference	58
5.16.1	Detailed Description	60
5.16.2	Function Documentation	61
5.16.2.1	get_date	61
5.16.2.2	get_date_main	61
5.16.2.3	get_time	61
5.16.2.4	get_time_main	61
5.16.2.5	set_date	61
5.16.2.6	set_date_main	61
5.16.2.7	set_date_str	61
5.16.2.8	set_time	61
5.16.2.9	set_time_main	61
5.16.2.10	set_time_str	61
5.17	modules/r2/pcb.c File Reference	61
5.17.1	Detailed Description	61
5.17.2	Variable Documentation	62
5.17.2.1	blocked_queue	62
5.17.2.2	ready_queue	62
5.18	modules/r2/pcb.h File Reference	62
5.18.1	Detailed Description	68
5.18.2	Macro Definition Documentation	68
5.18.2.1	COMMHAND_PCB_NAME	68
5.18.2.2	IDLE_PCB_NAME	68
5.18.2.3	SIZE_OF_PCB_NAME	68
5.18.2.4	SIZE_OF_STACK	68
5.18.3	Enumeration Type Documentation	68
5.18.3.1	process_class	68
5.18.4	Function Documentation	69

5.18.4.1	allocate_pcb	69
5.18.4.2	block_pcb	69
5.18.4.3	find_pcb	69
5.18.4.4	free_pcb	69
5.18.4.5	get_running_process	69
5.18.4.6	get_stack_base	69
5.18.4.7	get_stack_top	69
5.18.4.8	insert_pcb	69
5.18.4.9	pcb_init	69
5.18.4.10	remove_pcb	69
5.18.4.11	resume_pcb	69
5.18.4.12	save_running_process	69
5.18.4.13	set_pcb_priority	69
5.18.4.14	setup_pcb	69
5.18.4.15	show_all_processes	69
5.18.4.16	show_blocked_processes	69
5.18.4.17	show_pcb	69
5.18.4.18	show_ready_processes	69
5.18.4.19	shutdown_pcb	69
5.18.4.20	suspend_pcb	69
5.18.4.21	unblock_pcb	69
5.19	modules/r2/pcb_comm.h File Reference	69
5.19.1	Detailed Description	71
5.19.2	Function Documentation	71
5.19.2.1	resume_pcb_main	71
5.19.2.2	set_pcb_priority_main	71
5.19.2.3	show_pcb_main	71
5.19.2.4	suspend_pcb_main	71
5.20	modules/r3/context.h File Reference	72
5.20.1	Detailed Description	73
5.20.2	Function Documentation	74
5.20.2.1	load_process	74
5.20.2.2	load_r3_main	74
5.20.2.3	sys_call	74
5.20.3	Variable Documentation	74
5.20.3.1	cop	74
5.20.3.2	old_context	74

5.21 modules/r5/mcb.c File Reference	74
5.21.1 Detailed Description	74
5.21.2 Enumeration Type Documentation	75
5.21.2.1 mcb_type	75
5.22 modules/r5/mcb.h File Reference	75
5.22.1 Detailed Description	78
5.22.2 Macro Definition Documentation	78
5.22.2.1 MAX_HEAP_SIZE	78
5.22.3 Function Documentation	78
5.22.3.1 init_heap	78
5.22.3.2 is_mcb_empty	78
5.22.3.3 mcb_allocate	78
5.22.3.4 mcb_allocate_mpx	78
5.22.3.5 mcb_allocate_mpx2	78
5.22.3.6 mcb_free_mpx	78
5.22.3.7 show_all_mcb	78
5.22.3.8 show_allocated_mcb	78
5.22.3.9 show_free_mcb	78
5.22.3.10 show_mcb	78
5.22.3.11 show_mcb_main	78
5.22.3.12 shutdown_mcb	78
5.22.4 Variable Documentation	78
5.22.4.1 start_of_memory	78
5.23 modules/r6/ansi.h File Reference	79
5.23.1 Macro Definition Documentation	79
5.23.1.1 B_CYAN	79
5.23.1.2 B_NRM	79
5.23.1.3 T_BOLD	79
5.23.1.4 T_BOLD_OFF	79
5.23.1.5 T_CYAN	79
5.23.1.6 T_DIR	79
5.23.1.7 T_DIR_OFF	79
5.23.1.8 T_ITCS	79
5.23.1.9 T_ITCS_OFF	79
5.23.1.10 T_NRM	79
5.23.1.11 T_RED	79
5.23.1.12 T_RESET	79

5.23.1.13 T_WHT	79
5.24 modules/r6/disk_file_manager.h File Reference	79
5.24.1 Detailed Description	81
5.24.2 Function Documentation	81
5.24.2.1 delete_file	81
5.24.2.2 extract_file	81
5.24.2.3 import_file	81
5.24.2.4 move_file	81
5.24.2.5 type_file	81
5.25 modules/r6/disk_folder_manager.h File Reference	81
5.25.1 Detailed Description	84
5.25.2 Macro Definition Documentation	84
5.25.2.1 FOLDER_STACK_SIZE	84
5.25.3 Function Documentation	84
5.25.3.1 change_dir	84
5.25.3.2 folder_manager_init	84
5.25.3.3 get_entry	84
5.25.3.4 get_entry_by_name	85
5.25.3.5 list_dir_entry_report	85
5.25.3.6 list_dir_entry_short	85
5.25.3.7 list_file_report	85
5.25.3.8 list_files_entry_ext	85
5.25.3.9 list_files_entry_name	85
5.25.3.10 pop_folder	85
5.25.3.11 print_curr_path	85
5.25.3.12 print_dir_entry_info	85
5.25.3.13 print_report_heading	85
5.25.3.14 push_folder	85
5.25.3.15 rename_entry	85
5.26 modules/r6/disk_img_manager.h File Reference	85
5.26.1 Detailed Description	90
5.26.2 Macro Definition Documentation	90
5.26.2.1 ATTR_ARCH_ONLY	90
5.26.2.2 ATTRIBUTE_ARCH	90
5.26.2.3 ATTRIBUTE_HIDD	90
5.26.2.4 ATTRIBUTE_READ	90
5.26.2.5 ATTRIBUTE_SUBD	90

5.26.2.6	ATTRIBUTE_SYST	90
5.26.2.7	ATTRIBUTE_UUS1	90
5.26.2.8	ATTRIBUTE_UUS2	90
5.26.2.9	ATTRIBUTE_VOLL	90
5.26.3	Function Documentation	90
5.26.3.1	calc_free_space	90
5.26.3.2	ch_arr_to_str	90
5.26.3.3	clean_buffers	90
5.26.3.4	fat	90
5.26.3.5	find_unused_fat	90
5.26.3.6	get_data_ptr	90
5.26.3.7	get_fat_date	90
5.26.3.8	get_fat_date_str	91
5.26.3.9	get_fat_time	91
5.26.3.10	get_fat_time_str	91
5.26.3.11	get_fat_val	91
5.26.3.12	load_image_file	91
5.26.3.13	print_boot_sec_info	91
5.26.3.14	seperate_file_name	91
5.26.3.15	set_fat_time	91
5.26.3.16	str_to_ch_arr	91
5.26.3.17	str_to_upper_case	91
5.26.3.18	write_fat	91
5.26.3.19	write_image_file	91
5.26.4	Variable Documentation	91
5.26.4.1	boot_sec	91
5.26.4.2	data_area	91
5.26.4.3	root_dir_entry	91
5.26.4.4	root_dir_file_arr	91
5.27	modules/r6/file_dir_iterator.h File Reference	91
5.27.1	Detailed Description	95
5.27.2	Macro Definition Documentation	96
5.27.2.1	ROOT_DIR_SEC_INDEX	96
5.27.3	Function Documentation	96
5.27.3.1	ditr_begin	96
5.27.3.2	ditr_end	96
5.27.3.3	ditr_get	96

5.27.3.4	ditr_next	96
5.27.3.5	ditr_set_filter	96
5.27.3.6	ditr_set_find_unused	96
5.27.3.7	fitr_begin	96
5.27.3.8	fitr_end	96
5.27.3.9	fitr_get	96
5.27.3.10	fitr_next	96
5.27.3.11	init_dir_itr	96
5.27.3.12	init_file_itr	96
5.27.3.13	init_img_writer	96
5.27.3.14	iw_write	96

Index**97**

Chapter 1

Main Page

Welcome to the Programmer's manual for the Thunder Kracken's MPX Operating system. This document catalogues all of the information one may need to know regarding the use and modification of this Operating system and its contents. Included is a complete API of every method created for the operating system which includes all inputs and outputs as well as a brief summary of the purpose of each method. This will give you a more in depth look at all of the ordinary user commands as well as the internal commands used to perform functions that normal users cannot access. Most likely these commands will be the most important for making new programs on the operating system. This document also lists the documentation for the files in the operating system. This includes all of the variables and methods used in each file. These will help direct you as to where certain functions are defined. For general usage tips, please refer to the user manual. We hope you find working with the Thunder Kracken's MPX Operating System as enjoyable as we do and we thank you for using our product.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

cmcb	Complete Memory Control Block Struct	7
context	Context structure that holds the 15 CPU register values to begin and resume process execution	8
data_sector	Structure containing the sector's data	10
date_time	11
dir_entry_info	Structure containing the directory/file entry's information and data in sector	11
dir_itr	14
fat_date	Structure containing the date information	14
fat_time	Structure containing the time information	15
file_iter	15
footer	16
gdt_descriptor	16
gdt_entry	17
header	17
heap	18
idt_descriptor	19
idt_entry	19
img_boot_sector	Structure containing the Boot Sector's information and data	20
img_writer	23
index_entry	23
index_table	24
lmcb	Limited Memory Control Block Struct	24
mcb	Memory Control Block Struct	25
page_dir	26
page_entry	27
page_table	28

param	A structure to represent interrupt	29
pcb_queue	Queue structure that will store PCBs	29
pcb_struct	Struct that will describe PCB Processes	31

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

include/string.h	
Many usefull functions that used for handling string	40
include/system.h	45
include/core/asm.h	33
include/core/interrupts.h	33
include/core/io.h	34
include/core/serial.h	
Serial - Header	34
include/core/tables.h	37
include/mem/heap.h	38
include/mem/paging.h	39
modules/cmd_orders.h	
This file contains orders & index of all the commands	46
modules/errno.h	
This file contains the type of errors	49
modules/mpx_supt.h	
MPX System Supplementaries	51
modules/packing.h	
Packing classes	55
modules/r1/r1.h	
The command handler and functions associations for Module R1	56
modules/r1/sys_clock.h	
The main file that manipulates and controls the system's clock	58
modules/r2/pcb.c	
The Process Control Block	61
modules/r2/pcb.h	
The Process Control Block	62
modules/r2/pcb_comm.h	
The main functions that manipulate the PCB	69
modules/r3/context.h	
Context Switching	72
modules/r5/mcb.c	
Memory Control Block	74

modules/r5/ mcb.h	
Memory Control Block	75
modules/r6/ ansi.h	79
modules/r6/ disk_file_manager.h	
Disk File Manager	79
modules/r6/ disk_folder_manager.h	
Disk Folder Manager	81
modules/r6/ disk_img_manager.h	
Disk Image Manager	85
modules/r6/ file_dir_iterator.h	
File Directory Iterator	91

Chapter 4

Data Structure Documentation

4.1 cmcb Struct Reference

Complete Memory Control Block Struct.

Data Fields

- enum `mcb_type` `type`
Type indicating free or allocated.
- void * `begin_address`
Beginning address.
- `u32int` `size`
Indicates size of block in bytes.

4.1.1 Detailed Description

Complete Memory Control Block Struct.

These members are private to prevent potential manipulation so that the MPX system can remain functioning correctly.

4.1.2 Field Documentation

4.1.2.1 void* cmcb::begin_address

Beginning address.

4.1.2.2 u32int cmcb::size

Indicates size of block in bytes.

4.1.2.3 enum mcb_type cmcb::type

Type indicating free or allocated.

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

- [modules/r5/mcb.c](#)

4.2 context Struct Reference

Context structure that holds the 15 CPU register values to begin and resume process execution.

```
#include <context.h>
```

Data Fields

- [u32int gs](#)
Segment register.
- [u32int fs](#)
Segment register.
- [u32int es](#)
Segment register.
- [u32int ds](#)
Segment register.
- [u32int edi](#)
General-purpose register.
- [u32int esi](#)
General-purpose register.
- [u32int ebp](#)
General-purpose register.
- [u32int esp](#)
General-purpose register.
- [u32int ebx](#)
General-purpose register.
- [u32int edx](#)
General-purpose register.
- [u32int ecx](#)
General-purpose register.
- [u32int eax](#)
General-purpose register.
- [u32int eip](#)
Status and control register.
- [u32int cs](#)
Status and control register.
- [u32int eflags](#)
Status and control register.

4.2.1 Detailed Description

Context structure that holds the 15 CPU register values to begin and resume process execution.

4.2.2 Field Documentation

4.2.2.1 u32int context::cs

Status and control register.

4.2.2.2 u32int context::ds

Segment register.

4.2.2.3 u32int context::eax

General-purpose register.

4.2.2.4 u32int context::ebp

General-purpose register.

4.2.2.5 u32int context::ebx

General-purpose register.

4.2.2.6 u32int context::ecx

General-purpose register.

4.2.2.7 u32int context::edi

General-purpose register.

4.2.2.8 u32int context::edx

General-purpose register.

4.2.2.9 u32int context::eflags

Status and control register.

4.2.2.10 u32int context::eip

Status and control register.

4.2.2.11 u32int context::es

Segment register.

4.2.2.12 `u32int context::esi`

General-purpose register.

4.2.2.13 `u32int context::esp`

General-purpose register.

4.2.2.14 `u32int context::fs`

Segment register.

4.2.2.15 `u32int context::gs`

Segment register.

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

- [modules/r3/context.h](#)

4.3 `data_sector` Struct Reference

Structure containing the sector's data.

```
#include <disk_img_manager.h>
```

Data Fields

- `uint8_t data` [512]
The sector's raw data.

4.3.1 Detailed Description

Structure containing the sector's data.

•

4.3.2 Field Documentation

4.3.2.1 `uint8_t data_sector::data`[512]

The sector's raw data.

Number of bytes 512.

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

- [modules/r6/disk_img_manager.h](#)

4.4 date_time Struct Reference

```
#include <system.h>
```

Data Fields

- int [sec](#)
- int [min](#)
- int [hour](#)
- int [day_w](#)
- int [day_m](#)
- int [day_y](#)
- int [mon](#)
- int [year](#)

4.4.1 Field Documentation

4.4.1.1 int date_time::day_m

4.4.1.2 int date_time::day_w

4.4.1.3 int date_time::day_y

4.4.1.4 int date_time::hour

4.4.1.5 int date_time::min

4.4.1.6 int date_time::mon

4.4.1.7 int date_time::sec

4.4.1.8 int date_time::year

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

- include/[system.h](#)

4.5 dir_entry_info Struct Reference

Structure containing the directory/file entry's information and data in sector.

```
#include <disk_img_manager.h>
```

Data Fields

- uint8_t [file_name](#) [8]
File name in ASCII Characters.
- uint8_t [extension](#) [3]

File extension in ASCII Characters.

- uint8_t [attributes](#)

File attributes.

- uint16_t [reserved](#)

Reserved.

- uint16_t [create_time](#)

Time created file.

- uint16_t [create_date](#)

Date created file.

- uint16_t [last_acc_date](#)

Date last accessed file.

- uint16_t [ignore1](#)

Ignore file data.

- uint16_t [last_wri_time](#)

Last modified file time.

- uint16_t [last_wri_date](#)

Last modified file date.

- uint16_t [first_log_clu](#)

First logical cluster specifies where the file or subdirectory begins.

- uint32_t [file_size](#)

File size in bytes.

4.5.1 Detailed Description

Structure containing the directory/file entry's information and data in sector.

4.5.2 Field Documentation

4.5.2.1 uint8_t dir_entry_info::attributes

File attributes.

Number of bytes 1 and starting at byte location 11.

4.5.2.2 uint16_t dir_entry_info::create_date

Date created file.

Number of bytes 2 and starting at byte location 16.

4.5.2.3 uint16_t dir_entry_info::create_time

Time created file.

Number of bytes 2 and starting at byte location 14.

4.5.2.4 uint8_t dir_entry_info::extension[3]

File extension in ASCII Characters.

Number of bytes 3 and starting at byte location 8.

4.5.2.5 uint8_t dir_entry_info::file_name[8]

File name in ASCII Characters.

Number of bytes 8 and starting at byte location 0.

4.5.2.6 uint32_t dir_entry_info::file_size

File size in bytes.

Number of bytes 4 and starting at byte location 28.

4.5.2.7 uint16_t dir_entry_info::first_log_clu

First logical cluster specifies where the file or subdirectory begins.

Number of bytes 2 and starting at byte location 26.

4.5.2.8 uint16_t dir_entry_info::ignore1

Ignore file data.

Number of bytes 2 and starting at byte location 20.

4.5.2.9 uint16_t dir_entry_info::last_acc_date

Date last accessed file.

Number of bytes 2 and starting at byte location 18.

4.5.2.10 uint16_t dir_entry_info::last_wri_date

Last modified file date.

Number of bytes 2 and starting at byte location 24.

4.5.2.11 uint16_t dir_entry_info::last_wri_time

Last modified file time.

Number of bytes 2 and starting at byte location 22.

4.5.2.12 uint16_t dir_entry_info::reserved

Reserved.

Number of bytes 2 and starting at byte location 12.

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

- [modules/r6/disk_img_manager.h](#)

4.6 `dir_itr` Struct Reference

```
#include <file_dir_iterator.h>
```

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

- [modules/r6/file_dir_iterator.h](#)

4.7 `fat_date` Struct Reference

Structure containing the date information.

```
#include <disk_img_manager.h>
```

Data Fields

- `uint8_t day`
Day.
- `uint8_t mon`
Month.
- `uint16_t year`
Year.

4.7.1 Detailed Description

Structure containing the date information.

4.7.2 Field Documentation

4.7.2.1 `uint8_t fat_date::day`

Day.

4.7.2.2 `uint8_t fat_date::mon`

Month.

4.7.2.3 `uint16_t fat_date::year`

Year.

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

- [modules/r6/disk_img_manager.h](#)

4.8 fat_time Struct Reference

Structure containing the time information.

```
#include <disk_img_manager.h>
```

Data Fields

- `uint8_t se`
Seconds.
- `uint8_t mi`
Minutes.
- `uint8_t hr`
Hours.

4.8.1 Detailed Description

Structure containing the time information.

4.8.2 Field Documentation

4.8.2.1 `uint8_t fat_time::hr`

Hours.

4.8.2.2 `uint8_t fat_time::mi`

Minutes.

4.8.2.3 `uint8_t fat_time::se`

Seconds.

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

- [modules/r6/disk_img_manager.h](#)

4.9 file_iter Struct Reference

```
#include <file_dir_iterator.h>
```

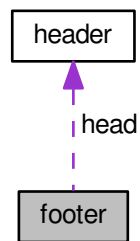
The documentation for this struct was generated from the following file:

- [modules/r6/file_dir_iterator.h](#)

4.10 footer Struct Reference

```
#include <heap.h>
```

Collaboration diagram for footer:



Data Fields

- [header head](#)

4.10.1 Field Documentation

4.10.1.1 header footer::head

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

- [include/mem/heap.h](#)

4.11 gdt_descriptor Struct Reference

```
#include <tables.h>
```

Data Fields

- [u16int limit](#)
- [u32int base](#)

4.11.1 Field Documentation

4.11.1.1 u32int gdt_descriptor::base

4.11.1.2 u16int gdt_descriptor::limit

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

- include/core/[tables.h](#)

4.12 gdt_entry Struct Reference

```
#include <tables.h>
```

Data Fields

- [u16int limit_low](#)
- [u16int base_low](#)
- [u8int base_mid](#)
- [u8int access](#)
- [u8int flags](#)
- [u8int base_high](#)

4.12.1 Field Documentation

4.12.1.1 u8int gdt_entry::access

4.12.1.2 u8int gdt_entry::base_high

4.12.1.3 u16int gdt_entry::base_low

4.12.1.4 u8int gdt_entry::base_mid

4.12.1.5 u8int gdt_entry::flags

4.12.1.6 u16int gdt_entry::limit_low

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

- include/core/[tables.h](#)

4.13 header Struct Reference

```
#include <heap.h>
```

Data Fields

- int [size](#)
- int [index_id](#)

4.13.1 Field Documentation

4.13.1.1 `int header::index_id`

4.13.1.2 `int header::size`

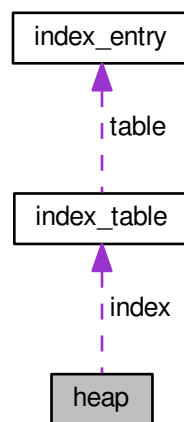
The documentation for this struct was generated from the following file:

- `include/mem/heap.h`

4.14 heap Struct Reference

```
#include <heap.h>
```

Collaboration diagram for heap:



Data Fields

- `index_table` `index`
- `u32int` `base`
- `u32int` `max_size`
- `u32int` `min_size`

4.14.1 Field Documentation

4.14.1.1 `u32int heap::base`

4.14.1.2 `index_table heap::index`

4.14.1.3 u32int heap::max_size

4.14.1.4 u32int heap::min_size

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

- include/mem/[heap.h](#)

4.15 idt_descriptor Struct Reference

```
#include <tables.h>
```

Data Fields

- [u16int limit](#)
- [u32int base](#)

4.15.1 Field Documentation

4.15.1.1 u32int idt_descriptor::base

4.15.1.2 u16int idt_descriptor::limit

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

- include/core/[tables.h](#)

4.16 idt_entry Struct Reference

```
#include <tables.h>
```

Data Fields

- [u16int base_low](#)
- [u16int sselect](#)
- [u8int zero](#)
- [u8int flags](#)
- [u16int base_high](#)

4.16.1 Field Documentation

4.16.1.1 u16int idt_entry::base_high

4.16.1.2 u16int idt_entry::base_low

4.16.1.3 u8int idt_entry::flags

4.16.1.4 `u16int idt_entry::sselect`

4.16.1.5 `u8int idt_entry::zero`

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

- `include/core/tables.h`

4.17 `img_boot_sector` Struct Reference

Structure containing the Boot Sector's information and data.

```
#include <disk_img_manager.h>
```

Data Fields

- `uint8_t ignore1 [11]`
Ignore file data.
- `uint16_t byte_per_sector`
Bytes per sector.
- `uint8_t sector_per_cluster`
Sectors per cluster.
- `uint16_t reserved_sec_num`
Number of reserved sectors.
- `uint8_t fat_copies_num`
Number of FAT copies.
- `uint16_t root_dir_max_num`
Max number of root directory entries.
- `uint16_t sec_num`
Total number of sectors in the File System.
- `uint8_t ignore2`
Ignore file data.
- `uint16_t sec_per_fat_num`
Number of Sectors per FAT.
- `uint16_t sec_per_track`
Sectors per track.
- `uint16_t head_num`
Number of heads.
- `uint32_t ignore3`
Ignore file data.
- `uint32_t total_sec_fat32`
Total sector count for FAT32.
- `uint16_t ignore4`
Ignore file data.
- `uint8_t boot_sign`
Boot signature.
- `uint32_t vol_id`

- Volume ID.*
 - uint8_t [vol_label](#) [11]
Volume label (or name of File System) as an ASCII string.
 - uint8_t [file_sys_type](#) [8]
File System Type as an ASCII string.
 - uint8_t [ignore5](#) [450]
Ignore file data.

4.17.1 Detailed Description

Structure containing the Boot Sector's information and data.

4.17.2 Field Documentation

4.17.2.1 uint8_t img_boot_sector::boot_sign

Boot signature.

Number of bytes 1 and starting at byte location 38.

4.17.2.2 uint16_t img_boot_sector::byte_per_sector

Bytes per sector.

Number of bytes 2 and starting at byte location 11.

4.17.2.3 uint8_t img_boot_sector::fat_copies_num

Number of FAT copies.

Number of bytes 1 and starting at byte location 16.

4.17.2.4 uint8_t img_boot_sector::file_sys_type[8]

File System Type as an ASCII string.

Number of bytes 8 and starting at byte location 54.

4.17.2.5 uint16_t img_boot_sector::head_num

Number of heads.

Number of bytes 2 and starting at byte location 26.

4.17.2.6 uint8_t img_boot_sector::ignore1[11]

Ignore file data.

Number of bytes 11 and starting at byte location 0.

4.17.2.7 `uint8_t img_boot_sector::ignore2`

Ignore file data.

Number of bytes 1 and starting at byte location 21.

4.17.2.8 `uint32_t img_boot_sector::ignore3`

Ignore file data.

Number of bytes 4 and starting at byte location 28.

4.17.2.9 `uint16_t img_boot_sector::ignore4`

Ignore file data.

Number of bytes 2 and starting at byte location 36.

4.17.2.10 `uint8_t img_boot_sector::ignore5[450]`

Ignore file data.

Number of bytes 450 and starting at byte location 62.

4.17.2.11 `uint16_t img_boot_sector::reserved_sec_num`

Number of reserved sectors.

Number of bytes 2 and starting at byte location 14.

4.17.2.12 `uint16_t img_boot_sector::root_dir_max_num`

Max number of root directory entries.

Number of bytes 2 and starting at byte location 17.

4.17.2.13 `uint16_t img_boot_sector::sec_num`

Total number of sectors in the File System.

Number of bytes 2 and starting at byte location 19.

4.17.2.14 `uint16_t img_boot_sector::sec_per_fat_num`

Number of Sectors per FAT.

Number of bytes 2 and starting at byte location 22.

4.17.2.15 `uint16_t img_boot_sector::sec_per_track`

Sectors per track.

Number of bytes 2 and starting at byte location 24.

4.17.2.16 uint8_t img_boot_sector::sector_per_cluster

Sectors per cluster.

Number of bytes 1 and starting at byte location 13.

4.17.2.17 uint32_t img_boot_sector::total_sec_fat32

Total sector count for FAT32.

Number of bytes 4 and starting at byte location 32.

4.17.2.18 uint32_t img_boot_sector::vol_id

Volume ID.

Number of bytes 4 and starting at byte location 39.

4.17.2.19 uint8_t img_boot_sector::vol_label[11]

Volume label (or name of File System) as an ASCII string.

Number of bytes 11 and starting at byte location 43.

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

- [modules/r6/disk_img_manager.h](#)

4.18 img_writer Struct Reference

```
#include <file_dir_iterator.h>
```

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

- [modules/r6/file_dir_iterator.h](#)

4.19 index_entry Struct Reference

```
#include <heap.h>
```

Data Fields

- [int size](#)
- [int empty](#)
- [u32int block](#)

4.19.1 Field Documentation

4.19.1.1 u32int index_entry::block

4.19.1.2 `int index_entry::empty`

4.19.1.3 `int index_entry::size`

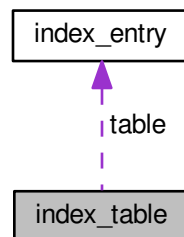
The documentation for this struct was generated from the following file:

- `include/mem/heap.h`

4.20 `index_table` Struct Reference

```
#include <heap.h>
```

Collaboration diagram for `index_table`:



Data Fields

- `index_entry table [TABLE_SIZE]`
- `int id`

4.20.1 Field Documentation

4.20.1.1 `int index_table::id`

4.20.1.2 `index_entry index_table::table[TABLE_SIZE]`

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

- `include/mem/heap.h`

4.21 `lmcb` Struct Reference

Limited Memory Control Block Struct.

Data Fields

- enum [mcb_type](#) type
Type indicating free or allocated.
- [u32int](#) size
Indicates size of block in bytes.

4.21.1 Detailed Description

Limited Memory Control Block Struct.

These members are private to prevent potential manipulation so that the MPX system can remain functioning correctly.

4.21.2 Field Documentation

4.21.2.1 [u32int](#) `lmcb::size`

Indicates size of block in bytes.

4.21.2.2 [enum mcb_type](#) `lmcb::type`

Type indicating free or allocated.

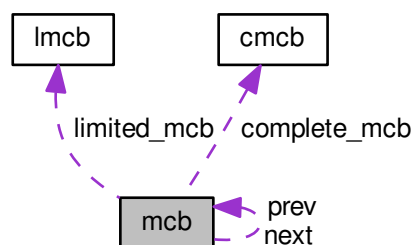
The documentation for this struct was generated from the following file:

- `modules/r5/mcb.c`

4.22 mcb Struct Reference

Memory Control Block Struct.

Collaboration diagram for mcb:



Data Fields

- struct `cmcb` * `complete_mcb`
Complete Memory Control Block.
- struct `lmcb` * `limited_mcb`
Limited Memory Control Block.
- struct `mcb` * `prev`
The previous adjacent Memory Control Block.
- struct `mcb` * `next`
The next adjacent Memory Control Block.

4.22.1 Detailed Description

Memory Control Block Struct.

These members are private to prevent potential manipulation so that the MPX system can remain functioning correctly.

4.22.2 Field Documentation

4.22.2.1 struct `cmcb`* `mcb::complete_mcb`

Complete Memory Control Block.

4.22.2.2 struct `lmcb`* `mcb::limited_mcb`

Limited Memory Control Block.

4.22.2.3 struct `mcb` * `mcb::next`

The next adjacent Memory Control Block.

4.22.2.4 struct `mcb`* `mcb::prev`

The previous adjacent Memory Control Block.

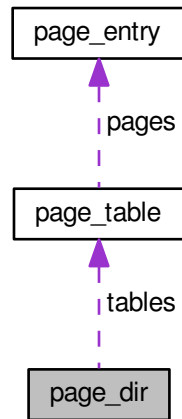
The documentation for this struct was generated from the following file:

- `modules/r5/mcb.c`

4.23 `page_dir` Struct Reference

```
#include <paging.h>
```

Collaboration diagram for page_dir:



Data Fields

- `page_table * tables` [1024]
- `u32int tables_phys` [1024]

4.23.1 Field Documentation

4.23.1.1 `page_table* page_dir::tables[1024]`

4.23.1.2 `u32int page_dir::tables_phys[1024]`

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

- `include/mem/paging.h`

4.24 page_entry Struct Reference

```
#include <paging.h>
```

Data Fields

- `u32int present`: 1
- `u32int writeable`: 1
- `u32int usermode`: 1
- `u32int accessed`: 1
- `u32int dirty`: 1

- [u32int reserved](#): 7
- [u32int frameaddr](#): 20

4.24.1 Field Documentation

4.24.1.1 [u32int page_entry::accessed](#)

4.24.1.2 [u32int page_entry::dirty](#)

4.24.1.3 [u32int page_entry::frameaddr](#)

4.24.1.4 [u32int page_entry::present](#)

4.24.1.5 [u32int page_entry::reserved](#)

4.24.1.6 [u32int page_entry::usermode](#)

4.24.1.7 [u32int page_entry::writeable](#)

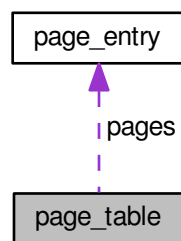
The documentation for this struct was generated from the following file:

- [include/mem/paging.h](#)

4.25 `page_table` Struct Reference

```
#include <paging.h>
```

Collaboration diagram for `page_table`:



Data Fields

- [page_entry pages](#) [1024]

4.25.1 Field Documentation

4.25.1.1 `page_entry` `page_table::pages[1024]`

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

- `include/mem/paging.h`

4.26 param Struct Reference

A structure to represent interrupt.

```
#include <mpx_supt.h>
```

Data Fields

- `int op_code`
interrupt's operation
- `int device_id`
interrupt's device

4.26.1 Detailed Description

A structure to represent interrupt.

4.26.2 Field Documentation

4.26.2.1 `int param::device_id`

interrupt's device

4.26.2.2 `int param::op_code`

interrupt's operation

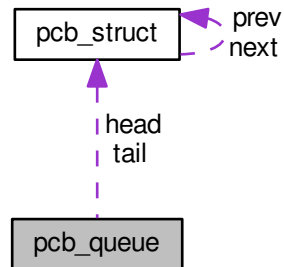
The documentation for this struct was generated from the following file:

- `modules/mpx_supt.h`

4.27 pcb_queue Struct Reference

Queue structure that will store PCBs.

Collaboration diagram for `pcb_queue`:



Data Fields

- `int count`
The length of the queue.
- `struct pcb_struct * head`
Pointer to the start/head of the queue.
- `struct pcb_struct * tail`
Pointer to the end/tail of the queue.

4.27.1 Detailed Description

Queue structure that will store PCBs.

These members are private to prevent potential manipulation so that the MPX system can remain functioning correctly.

4.27.2 Field Documentation

4.27.2.1 `int pcb_queue::count`

The length of the queue.

4.27.2.2 `struct pcb_struct* pcb_queue::head`

Pointer to the start/head of the queue.

4.27.2.3 `struct pcb_struct* pcb_queue::tail`

Pointer to the end/tail of the queue.

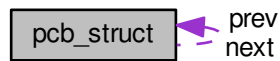
The documentation for this struct was generated from the following file:

- `modules/r2/pcb.c`

4.28 pcb_struct Struct Reference

Struct that will describe PCB Processes.

Collaboration diagram for pcb_struct:



Data Fields

- char [name](#) [[SIZE_OF_PCB_NAME](#)]
PCB's name.
- enum [process_class](#) [class](#)
PCB's class is an application or system process.
- unsigned char [priority](#)
PCB's priority an integer between 0 and 9.
- enum [process_state](#) [running_state](#)
PCB's states are ready, running, or blocked.
- enum [process_suspended](#) [is_suspended](#)
PCB process is either suspended or not suspended.
- unsigned char * [stack_top](#)
Pointer to top of the stack.
- unsigned char * [stack_base](#)
Pointer to base of the stack.
- struct [pcb_struct](#) * [prev](#)
Pointer to the previous PCB in the queue.
- struct [pcb_struct](#) * [next](#)
Pointer to the next PCB in the queue.

4.28.1 Detailed Description

Struct that will describe PCB Processes.

These members are private to prevent potential manipulation so that the MPX system can remain functioning correctly.

4.28.2 Field Documentation

4.28.2.1 enum [process_class](#) [pcb_struct::class](#)

PCB's class is an application or system process.

4.28.2.2 enum process_suspended pcb_struct::is_suspended

PCB process is either suspended or not suspended.

4.28.2.3 char pcb_struct::name[SIZE_OF_PCB_NAME]

PCB's name.

4.28.2.4 struct pcb_struct* pcb_struct::next

Pointer to the next PCB in the queue.

4.28.2.5 struct pcb_struct* pcb_struct::prev

Pointer to the previous PCB in the queue.

4.28.2.6 unsigned char pcb_struct::priority

PCB's priority an integer between 0 and 9.

Processes with higher priority values execute before lower priority processes.

4.28.2.7 enum process_state pcb_struct::running_state

PCB's states are ready, running, or blocked.

4.28.2.8 unsigned char* pcb_struct::stack_base

Pointer to base of the stack.

4.28.2.9 unsigned char* pcb_struct::stack_top

Pointer to top of the stack.

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

- [modules/r2/pcb.c](#)

Chapter 5

File Documentation

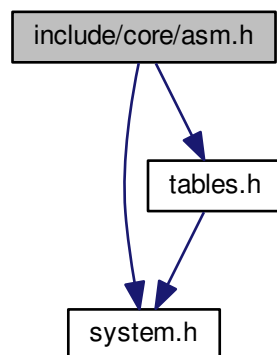
5.1 documentation/mainpage.dox File Reference

5.2 include/core/asm.h File Reference

```
#include <system.h>
```

```
#include <tables.h>
```

Include dependency graph for asm.h:



5.3 include/core/interrupts.h File Reference

Functions

- void `init_irq` (void)
- void `init_pic` (void)

5.3.1 Function Documentation

5.3.1.1 void init_irq (void)

5.3.1.2 void init_pic (void)

5.4 include/core/io.h File Reference

Macros

- #define outb(port, data) asm volatile ("outb %%al,%%dx" : : "a" (data), "d" (port))
- #define inb(port)

5.4.1 Macro Definition Documentation

5.4.1.1 #define inb(port)

Value:

```
((
    unsigned char r; \
    asm volatile ("inb %%dx,%%al": "=a" (r): "d" (port)); \
    r; \
))
```

5.4.1.2 #define outb(port, data) asm volatile ("outb %%al,%%dx" : : "a" (data), "d" (port))

5.5 include/core/serial.h File Reference

Serial - Header.

Macros

- #define COM1 0x3f8
- #define COM2 0x2f8
- #define COM3 0x3e8
- #define COM4 0x2e8
- #define WithoutEcho 0
- #define WithEcho 1
- #define USER_INPUT_BUFFER_SIZE 100

Functions

- int init_serial (int device)
- int serial_println (const char *msg)
- int serial_print (const char *msg)
- int set_serial_out (int device)
- int set_serial_in (int device)

get_input_line

Get user's input from keyboard.

Parameters

buffer	<i>The pointer to the buffer where store the user's input.</i>
buffer_size	<i>The size of that buffer.</i>
bWithEcho	<i>With echo or not</i>

*Returns**VOID*

- void [get_input_line](#) (char *buffer, const int bWithEcho)

5.5.1 Detailed Description

Serial - Header.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.5.2 Macro Definition Documentation

5.5.2.1 #define COM1 0x3f8

5.5.2.2 #define COM2 0x2f8

5.5.2.3 #define COM3 0x3e8

5.5.2.4 #define COM4 0x2e8

5.5.2.5 #define USER_INPUT_BUFFER_SIZE 100

5.5.2.6 #define WithEcho 1

5.5.2.7 #define WithoutEcho 0

5.5.3 Function Documentation

5.5.3.1 void [get_input_line](#) (char * *buffer*, const int *bWithEcho*)

5.5.3.2 int [init_serial](#) (int *device*)

5.5.3.3 int [serial_print](#) (const char * *msg*)

5.5.3.4 `int serial_println (const char * msg)`

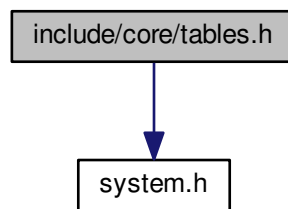
5.5.3.5 `int set_serial_in (int device)`

5.5.3.6 `int set_serial_out (int device)`

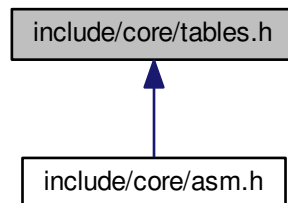
5.6 include/core/tables.h File Reference

```
#include "system.h"
```

Include dependency graph for tables.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [idt_entry](#)
- struct [idt_descriptor](#)
- struct [gdt_descriptor](#)
- struct [gdt_entry](#)

Functions

- void `idt_set_gate` (`u8int` *idx*, `u32int` *base*, `u16int` *sel*, `u8int` *flags*)
- void `gdt_init_entry` (`int` *idx*, `u32int` *base*, `u32int` *limit*, `u8int` *access*, `u8int` *flags*)
- void `init_idt` ()
- void `init_gdt` ()

5.6.1 Function Documentation

5.6.1.1 void `gdt_init_entry` (`int` *idx*, `u32int` *base*, `u32int` *limit*, `u8int` *access*, `u8int` *flags*)

5.6.1.2 void `idt_set_gate` (`u8int` *idx*, `u32int` *base*, `u16int` *sel*, `u8int` *flags*)

5.6.1.3 void `init_gdt` ()

5.6.1.4 void `init_idt` ()

5.7 include/mem/heap.h File Reference

Data Structures

- struct `header`
- struct `footer`
- struct `index_entry`
- struct `index_table`
- struct `heap`

Macros

- `#define` `TABLE_SIZE` 0x1000
- `#define` `KHEAP_BASE` 0xD000000
- `#define` `KHEAP_MIN` 0x10000
- `#define` `KHEAP_SIZE` 0x1000000

Functions

- `u32int` `_kmalloc` (`u32int` *size*, `int` *align*, `u32int` **phys_addr*)
- `u32int` `kmalloc` (`u32int` *size*)
- `u32int` `kfree` ()
- void `init_kheap` ()
- `u32int` `alloc` (`u32int` *size*, `heap` **hp*, `int` *align*)
- `heap` *** `make_heap` (`u32int` *base*, `u32int` *max*, `u32int` *min*)

5.7.1 Macro Definition Documentation

5.7.1.1 `#define KHEAP_BASE 0xD000000`

5.7.1.2 `#define KHEAP_MIN 0x10000`

5.7.1.3 `#define KHEAP_SIZE 0x1000000`

5.7.1.4 `#define TABLE_SIZE 0x1000`

5.7.2 Function Documentation

5.7.2.1 `u32int _kmalloc (u32int size, int align, u32int * phys_addr)`

5.7.2.2 `u32int alloc (u32int size, heap * hp, int align)`

5.7.2.3 `void init_kheap ()`

5.7.2.4 `u32int kfree ()`

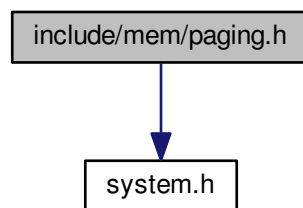
5.7.2.5 `u32int kmalloc (u32int size)`

5.7.2.6 `heap* make_heap (u32int base, u32int max, u32int min)`

5.8 include/mem/paging.h File Reference

```
#include <system.h>
```

Include dependency graph for paging.h:



Data Structures

- struct [page_entry](#)
- struct [page_table](#)
- struct [page_dir](#)

Macros

- #define `PAGE_SIZE` 0x1000

Functions

- void `set_bit` (u32int addr)
- void `clear_bit` (u32int addr)
- u32int `get_bit` (u32int addr)
- u32int `first_free` ()
- void `init_paging` ()
- void `load_page_dir` (page_dir *new_page_dir)
- page_entry * `get_page` (u32int addr, page_dir *dir, int make_table)
- void `new_frame` (page_entry *page)

5.8.1 Macro Definition Documentation

5.8.1.1 #define `PAGE_SIZE` 0x1000

5.8.2 Function Documentation

5.8.2.1 void `clear_bit` (u32int *addr*)

5.8.2.2 u32int `first_free` ()

5.8.2.3 u32int `get_bit` (u32int *addr*)

5.8.2.4 page_entry* `get_page` (u32int *addr*, page_dir * *dir*, int *make_table*)

5.8.2.5 void `init_paging` ()

5.8.2.6 void `load_page_dir` (page_dir * *new_page_dir*)

5.8.2.7 void `new_frame` (page_entry * *page*)

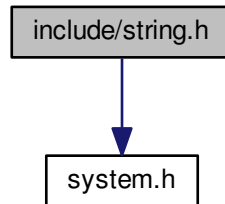
5.8.2.8 void `set_bit` (u32int *addr*)

5.9 include/string.h File Reference

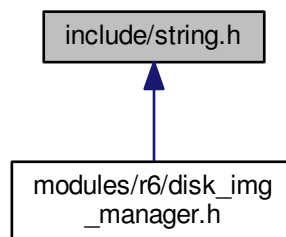
Many usefull functions that used for handling string.

```
#include <system.h>
```

Include dependency graph for string.h:



This graph shows which files directly or indirectly include this file:



Functions

isspace.

Identifies if its space

Parameters

A	constant character
---	--------------------

Returns

1 if it is space, otherwise return 0.

- int `isspace` (const char *c)

memset.

Sets region of memory

Parameters

s	destination
c	byte to write
n	count

Returns

the pointer to the memory space.

- void * [memset](#) (void *s, int c, [size_t](#) n)

strcpy.

Copies one string to another.

Parameters

s1	Destination string
s2	Source string

Returns

pointer to the destination String

- char * [strcpy](#) (char *s1, const char *s2)

strcat.

Concatenate the contents of one string onto another.

Parameters

s1	Destination string
s2	Source string

Returns

pointer to destination String

- char * [strcat](#) (char *s1, const char *s2)

strlen.

Returns the length of a string.

Parameters

s	String input.
---	---------------

Returns

count Length of the String

- int [strlen](#) (const char *s)

strcmp.

String comparison.

Parameters

s1	First string to use for the compare.
s2	Second string to use for the compare.

Returns

whether they are the same or not.

- int [strcmp](#) (const char *s1, const char *s2)

strtok.

Split string into tokens.

Parameters

s1	String
s2	Delimiter

Returns

the pointer to the token.

- char * [strtok](#) (char *s1, const char *s2)

atoi.

Convert an ASCII string to an integer.

Parameters

s	String.
---	---------

Returns

The converted integer.

- int [atoi](#) (const char *s)

sprintf.

Generate a formatted string.

%[-x]c output a character, '-' - align right, x - the output width

%[-x]s output a string, '-' - align right, x - the output width

%[{-,+}x]d output a character, '-' - align right, '+' - align right and display '+' sign, x - the output width

%[-x]X (capital 'X') output a hexadecimal number, '-' - align right, x - the output width

note: Output width will be ignored if width is smaller than actual length.

Parameters

str	- Output string.
format	- The format of the string.
...	- All of the additional parameters.

Returns

vsprintf(str, format, ap) - Return the string with its format and pointer.

- int [sprintf](#) (char *str, const char *format,...)

printf.

Print out a formatted string.

%[-x]c output a character, '-' - align right, x - the output width

%[-x]s output a string, '-' - align right, x - the output width

%[{-,+}x]d output a character, '-' - align right, '+' - align right and display '+' sign, x - the output width

%[-x]X (capital 'X') output a hexadecimal number, '-' - align right, x - the output width

note: Output width will be ignored if width is smaller than actual length.

Parameters

str	- Output string.
format	- The format of the string.
...	- All of the additional parameters.

Returns

vsprintf(str, format, ap) - Return the string with its format and pointer.

- int [printf](#) (const char *format,...)

5.9.1 Detailed Description

Many usefull functions that used for handling string.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.9.2 Function Documentation

5.9.2.1 int atoi (const char * s)

5.9.2.2 int isspace (const char * c)

5.9.2.3 void* memset (void * s, int c, size_t n)

5.9.2.4 int printf (const char * format, ...)

5.9.2.5 `int sprintf (char * str, const char * format, ...)`

5.9.2.6 `char* strcat (char * s1, const char * s2)`

5.9.2.7 `int strcmp (const char * s1, const char * s2)`

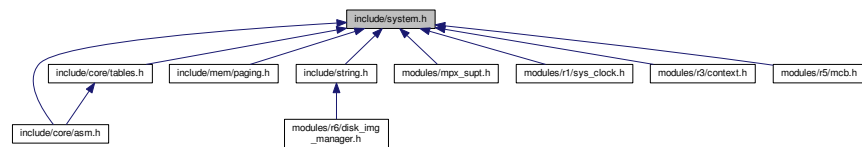
5.9.2.8 `char* strcpy (char * s1, const char * s2)`

5.9.2.9 `int strlen (const char * s)`

5.9.2.10 `char* strtok (char * s1, const char * s2)`

5.10 include/system.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct [date_time](#)

Macros

- `#define` [NULL](#) 0
- `#define` [no_warn](#)(p) if (p) while (1) break
- `#define` [asm](#) __asm__
- `#define` [volatile](#) __volatile__
- `#define` [sti](#)() [asm](#) [volatile](#) ("sti::")
- `#define` [cli](#)() [asm](#) [volatile](#) ("cli::")
- `#define` [nop](#)() [asm](#) [volatile](#) ("nop::")
- `#define` [hlt](#)() [asm](#) [volatile](#) ("hlt::")
- `#define` [iret](#)() [asm](#) [volatile](#) ("iret::")
- `#define` [GDT_CS_ID](#) 0x01
- `#define` [GDT_DS_ID](#) 0x02

Typedefs

- typedef unsigned int [size_t](#)
- typedef unsigned char [u8int](#)
- typedef unsigned short [u16int](#)
- typedef unsigned long [u32int](#)

Functions

- static int `irq_on` ()
- void `klogv` (const char *msg)
- void `kpanic` (const char *msg)

5.10.1 Macro Definition Documentation

5.10.1.1 `#define asm __asm__`

5.10.1.2 `#define cli() asm volatile ("cli::")`

5.10.1.3 `#define GDT_CS_ID 0x01`

5.10.1.4 `#define GDT_DS_ID 0x02`

5.10.1.5 `#define hlt() asm volatile ("hlt::")`

5.10.1.6 `#define iret() asm volatile ("iret::")`

5.10.1.7 `#define no_warn(p) if (p) while (1) break`

5.10.1.8 `#define nop() asm volatile ("nop::")`

5.10.1.9 `#define NULL 0`

5.10.1.10 `#define sti() asm volatile ("sti::")`

5.10.1.11 `#define volatile __volatile__`

5.10.2 Typedef Documentation

5.10.2.1 `typedef unsigned int size_t`

5.10.2.2 `typedef unsigned short u16int`

5.10.2.3 `typedef unsigned long u32int`

5.10.2.4 `typedef unsigned char u8int`

5.10.3 Function Documentation

5.10.3.1 `static int irq_on() [inline],[static]`

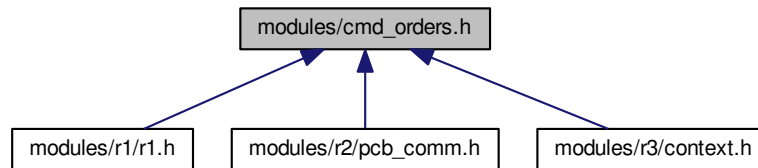
5.10.3.2 `void klogv (const char * msg)`

5.10.3.3 `void kpanic (const char * msg)`

5.11 modules/cmd_orders.h File Reference

This file contains orders & index of all the commands.

This graph shows which files directly or indirectly include this file:



Macros

- #define `WITH_R2_TEMP_CMD` 0
- #define `WITH_R3_TEMP_CMD` 0
- #define `WITH_R5_TEMP_CMD` 0
- #define `FUNCTIONS_BEGIN` 0

- #define `HELP` 0
- #define `MPX_FUNCTIONS_BEGIN` 1

- #define `VERSION` `MPX_FUNCTIONS_BEGIN+0`
- #define `GETTIME` `MPX_FUNCTIONS_BEGIN+1`
- #define `SETTIME` `MPX_FUNCTIONS_BEGIN+2`
- #define `GETDATE` `MPX_FUNCTIONS_BEGIN+3`
- #define `SETDATE` `MPX_FUNCTIONS_BEGIN+4`
- #define `SHUTDOWN` `MPX_FUNCTIONS_BEGIN+5`
- #define `LOADR3` `MPX_FUNCTIONS_BEGIN+6`
- #define `MPX_FUNC_END` `MPX_FUNCTIONS_BEGIN+6`
- #define `PCB_FUNCTIONS_BEGIN` `MPX_FUNC_END+1`

- #define `SUSPDCB` `PCB_FUNCTIONS_BEGIN+0`
- #define `RESUMEPCB` `PCB_FUNCTIONS_BEGIN+1`
- #define `SETPCBPRIO` `PCB_FUNCTIONS_BEGIN+2`
- #define `SHOWPCB` `PCB_FUNCTIONS_BEGIN+3`
- #define `PCB_FUNC_END` `PCB_FUNCTIONS_BEGIN+3`
- #define `MCB_FUNCTIONS_BEGIN` `PCB_FUNC_END+1`

- #define `SHOWMCB` `MCB_FUNCTIONS_BEGIN+0`
- #define `MCB_FUNC_END` `MCB_FUNCTIONS_BEGIN+0`
- #define `NUM_OF_FUNCTIONS` `MCB_FUNC_END+1`

5.11.1 Detailed Description

This file contains orders & index of all the commands.

Author

Thunder Krakens

Date

February 7nd, 2016

Version

R5

5.11.2 Macro Definition Documentation

5.11.2.1 `#define FUNCTIONS_BEGIN 0`

5.11.2.2 `#define GETDATE MPX_FUNCTIONS_BEGIN+3`

5.11.2.3 `#define GETTIME MPX_FUNCTIONS_BEGIN+1`

5.11.2.4 `#define HELP 0`

5.11.2.5 `#define LOADR3 MPX_FUNCTIONS_BEGIN+6`

5.11.2.6 `#define MCB_FUNC_END MCB_FUNCTIONS_BEGIN+0`

5.11.2.7 `#define MCB_FUNCTIONS_BEGIN PCB_FUNC_END+1`

5.11.2.8 `#define MPX_FUNC_END MPX_FUNCTIONS_BEGIN+6`

5.11.2.9 `#define MPX_FUNCTIONS_BEGIN 1`

5.11.2.10 `#define NUM_OF_FUNCTIONS MCB_FUNC_END+1`

5.11.2.11 `#define PCB_FUNC_END PCB_FUNCTIONS_BEGIN+3`

5.11.2.12 `#define PCB_FUNCTIONS_BEGIN MPX_FUNC_END+1`

5.11.2.13 `#define RESUMEPCB PCB_FUNCTIONS_BEGIN+1`

5.11.2.14 `#define SETDATE MPX_FUNCTIONS_BEGIN+4`

5.11.2.15 `#define SETPCBPRIOR PCB_FUNCTIONS_BEGIN+2`

5.11.2.16 `#define SETTIME MPX_FUNCTIONS_BEGIN+2`

5.11.2.17 `#define SHOWMCB MCB_FUNCTIONS_BEGIN+0`

5.11.2.18 `#define SHOWPCB PCB_FUNCTIONS_BEGIN+3`

5.11.2.19 `#define SHUTDOWN MPX_FUNCTIONS_BEGIN+5`

5.11.2.20 `#define SUSPDPCB PCB_FUNCTIONS_BEGIN+0`

5.11.2.21 `#define VERSION MPX_FUNCTIONS_BEGIN+0`

5.11.2.22 `#define WITH_R2_TEMP_CMD 0`

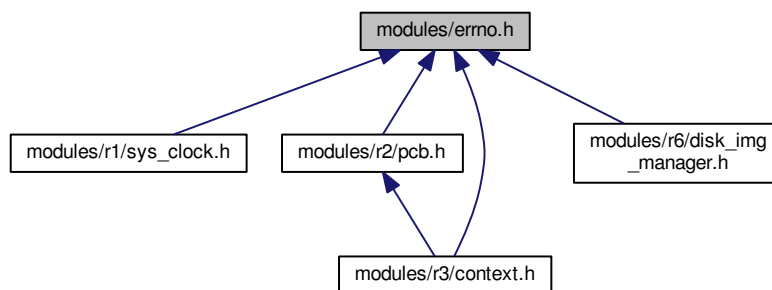
5.11.2.23 `#define WITH_R3_TEMP_CMD 0`

5.11.2.24 `#define WITH_R5_TEMP_CMD 0`

5.12 modules/errno.h File Reference

This file contains the type of errors.

This graph shows which files directly or indirectly include this file:



Macros

- `#define E_NOERROR 0`
- `#define E_INVPARA 1`
- `#define E_INVSTRF 2`
- `#define E_INVUSRI 3`
- `#define E_FREEMEM 4`

Error we cannot actually free the memory space since the student_free had not been implemented before R5.

- `#define E_NULL_PTR 5`

A NULL Pointer Error.

- `#define E_EMPTYPCB 6`

The pcb queue is empty.

- `#define E_PCB_SYS 7`

- `#define E_FILE_NF 8`

The file was not found.

- `#define E_NAMEDUP 9`
The specific file name is already exist.
- `#define E_NAMEINV 10`
The specific file name contains invalid character.
- `#define E_NOSPACE 11`
Not enough space to store the file.
- `#define E_INVATTRS 12`
The file's attributes are invalid.
- `#define E_FOLDFUL 13`
The specific directory is full.
- `#define E_PROGERR 99`

Typedefs

error_t.

The datatype that holds the error code.

- `typedef unsigned int error_t`

5.12.1 Detailed Description

This file contains the type of errors.

Author

Thunder Krakens

Date

February 7nd, 2016

Version

R2

The error can be from invalid paramter passed to a function, or invalid input format.

5.12.2 Macro Definition Documentation

5.12.2.1 `#define E_EMPTPCB 6`

The pcb queue is empty.

5.12.2.2 `#define E_FILE_NF 8`

The file was not found.

5.12.2.3 #define E_FOLDFUL 13

The specific directory is full.

5.12.2.4 #define E_FREEMEM 4

Error we cannot actually free the memory space since the student_free had not been implemented before R5.

5.12.2.5 #define E_INVATTRS 12

The file's attributes are invalid.

5.12.2.6 #define E_INVPARA 1

5.12.2.7 #define E_INVSTRF 2

5.12.2.8 #define E_INVUSRI 3

5.12.2.9 #define E_NAMEDUP 9

The specific file name is already exist.

5.12.2.10 #define E_NAMEINV 10

The specific file name contains invalid character.

5.12.2.11 #define E_NOERROR 0

5.12.2.12 #define E_NOSPACE 11

Not enough space to store the file.

5.12.2.13 #define E_NULL_PTR 5

A NULL Pointer Error.

5.12.2.14 #define E_PCB_SYS 7

5.12.2.15 #define E_PROGERR 99

5.12.3 Typedef Documentation

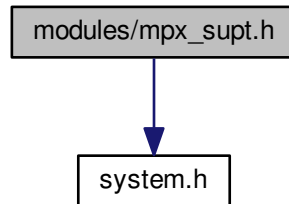
5.12.3.1 typedef unsigned int error_t

5.13 modules/mpx_supt.h File Reference

MPX System Supplementaries.

```
#include <system.h>
```

Include dependency graph for mpx_supt.h:



Data Structures

- struct [param](#)
A structure to represent interrupt.

Macros

- `#define EXIT 0`
- `#define IDLE 1`
- `#define READ 2`
- `#define WRITE 3`
- `#define MODULE_R1 0`
- `#define MODULE_R2 1`
- `#define MODULE_R3 2`
- `#define MODULE_R4 4`
- `#define MODULE_R5 8`

Functions

sys_req

Generate interrupt 60H

Parameters

int	<i>op_code (IDLE)</i>
-----	-----------------------

- int [sys_req](#) (int op_code)

mpx_init

Initialize MPX support software

Parameters

int	<i>cur_mod</i> (symbolic constants <i>MODULE_R1</i> , <i>MODULE_R2</i> , etc
-----	--

- void [mpx_init](#) (int *cur_mod*)

set_malloc

Sets the memory allocation function for sys_alloc_mem

Parameters

Function	<i>pointer</i>
----------	----------------

- void [sys_set_malloc](#) (u32int(*func)(u32int))

set_free

Sets the memory free function for sys_free_mem

Parameters

s1- destination,s2- source	
----------------------------------	--

- void [sys_set_free](#) (int(*func)(void *))

sys_alloc_mem

Allocates a block of memory (similar to malloc)

Parameters

Number	<i>of bytes to allocate</i>
--------	-----------------------------

- void * [sys_alloc_mem](#) (u32int *size*)

sys_free_mem

Frees memory

Parameters

Pointer	<i>to block of memory to free</i>
---------	-----------------------------------

- int [sys_free_mem](#) (void **ptr*)

idle

The idle process

Parameters

None	
------	--

- void [idle](#) ()

get_op_code

Returns the interrupt's operation code

Parameters

None	
------	--

- int [get_op_code](#) ()

5.13.1 Detailed Description

MPX System Supplementaries.

Author

Thunder Krakens

Date

March 18, 2016

Version

R3

5.13.2 Macro Definition Documentation

5.13.2.1 `#define EXIT 0`

5.13.2.2 `#define IDLE 1`

5.13.2.3 `#define MODULE_R1 0`

5.13.2.4 `#define MODULE_R2 1`

5.13.2.5 `#define MODULE_R3 2`

5.13.2.6 `#define MODULE_R4 4`

5.13.2.7 `#define MODULE_R5 8`

5.13.2.8 `#define READ 2`

5.13.2.9 `#define WRITE 3`

5.13.3 Function Documentation

5.13.3.1 `int get_op_code ()`

5.13.3.2 `void idle ()`

5.13.3.3 `void mpx_init (int cur_mod)`

5.13.3.4 `void* sys_alloc_mem (u32int size)`

5.13.3.5 `int sys_free_mem (void * ptr)`

5.13.3.6 `int sys_req (int op_code)`

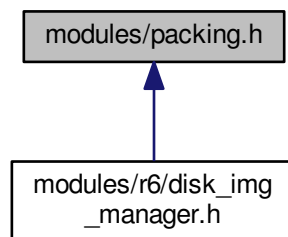
5.13.3.7 `void sys_set_free (int(*) (void *) func)`

5.13.3.8 `void sys_set_malloc (u32int(*) (u32int) func)`

5.14 modules/packing.h File Reference

Packing classes.

This graph shows which files directly or indirectly include this file:



Macros

- `#define PACKED(class_to_pack) __pragma(pack(push, 1)) class_to_pack __pragma(pack(pop))`

5.14.1 Detailed Description

Packing classes.

Author

Thunder Krakens

Date

April 28th, 2016

Version

R6

5.14.2 Macro Definition Documentation

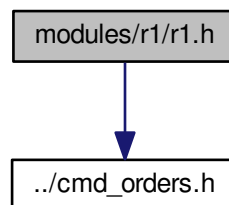
5.14.2.1 `#define PACKED(class_to_pack) __pragma(pack(push, 1)) class_to_pack __pragma(pack(pop))`

5.15 modules/r1/r1.h File Reference

The command handler and functions associations for Module R1.

```
#include "../cmd_orders.h"
```

Include dependency graph for r1.h:



Enumerations

- enum `comm_type` { `mpx`, `pcb`, `mcb`, `help` }
Command types.

Functions

`commhand`

Accepts and handles commands from the user.

- void `commhand` ()

`command_line_parser`

Splits the complete command line into tokens by space, single quote, or double quote.

Parameters

<code>CmdStr</code>	<i>The complete input command.</i>
<code>argc</code>	<i>The number of tokens found.</i>
<code>argv</code>	<i>The array of tokens.</i>
<code>MaxArgNum</code>	<i>The maximum number of tokens that array can hold.</i>

MaxStrLen	<i>The maximum length of each token that string can hold.</i>
-----------	---

- void [command_line_parser](#) (const char *CmdStr, int *argc, char **argv, const int MaxArgNum, const int MaxStrLen)

print_help

Prints the help message of a certain function that specified by the index number

Parameters

function_index	<i>The index number of that function.</i>
----------------	---

- void [print_help](#) (const int function_index)

help_usages

Displays all the usage case of the specified command type.

Parameters

comm_type	<i>The command type.</i>
-----------	--------------------------

Returns

When finished execution returns 0.

- int [help_usages](#) (enum [comm_type](#) type)

5.15.1 Detailed Description

The command handler and functions associations for Module R1.

Author

Thunder Krakens

Date

March 17, 2016

Version

R3 & R4

5.15.2 Enumeration Type Documentation**5.15.2.1 enum comm_type**

Command types.

Enumerator

mpx MPX System command.

pcb Process Control Block command.

mcb Memory Control Block command.

help Help command.

5.15.3 Function Documentation

5.15.3.1 `void command_line_parser (const char * CmdStr, int * argc, char ** argv, const int MaxArgNum, const int MaxStrLen)`

5.15.3.2 `void commhand ()`

5.15.3.3 `int help_usages (enum comm_type type)`

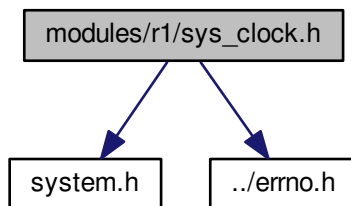
5.15.3.4 `void print_help (const int function_index)`

5.16 modules/r1/sys_clock.h File Reference

The main file that manipulates and controls the system's clock.

```
#include <system.h>
#include "../errno.h"
```

Include dependency graph for sys_clock.h:



Functions

set_time_main

The main set time argument handler for set time function.

Parameters

<code>argc</code>	<i>The number of tokens found.</i>
<code>argv</code>	<i>The array of tokens.</i>

Returns

0 when finished with execution.

- int [set_time_main](#) (int `argc`, char **`argv`)

get_time_main

The main get time argument handler for the get time function.

Parameters

argc	The number of tokens found.
argv	The array of tokens.

Returns

0 when finished with execution.

- int [get_time_main](#) (int argc, char **argv)

set_time_str

Sets the time for the system by string.

Parameters

timeStr	The string type of current Time.
---------	----------------------------------

Returns

The appropriate error code. See [errno.h](#) for details.

- [error_t set_time_str](#) (const char *timeStr)

get_time

Retrieves the system's current time and date.

Parameters

dateTimeValues	The value of current time and date
----------------	------------------------------------

- void [get_time](#) ([date_time](#) *dateTimeValues)

set_time

Sets the time for the system by [date_time](#) structure.

Parameters

dateTimeValues	The structure that holds the time values.
----------------	---

Returns

The appropriate error number. See [errno.h](#) for details.

- [error_t set_time](#) (const [date_time](#) *dateTimeValues)

set_date_main

The set date argument handler for the set date function.

Parameters

argc	The number of tokens.
argv	The array of tokens.

Returns

0 when finished execution.

- int [set_date_main](#) (int argc, char **argv)

get_date_main

The get date argument handler for the get date function.

Parameters

argc	The number of tokens.
argv	The array of tokens.

Returns

0 when finished with execution.

- int [get_date_main](#) (int argc, char **argv)

get_date

Retrieves system's current date.

Parameters

dateTimeValues	The structure that holds the value of current date.
----------------	---

- void [get_date](#) ([date_time](#) *dateTimeValues)

set_date_str

Sets the date for the system by string.

Parameters

str	The string type of current date.
-----	----------------------------------

Returns

0 when finished with execution.

- int [set_date_str](#) (const char *str)

set_date.

Sets the date of the system.

Parameters

dateTimeValues	The structure that holds the value of date
----------------	--

Returns

The appropriate error number. See [errno.h](#) for details.

- [error_t](#) [set_date](#) (const [date_time](#) *dateTimeValues)

5.16.1 Detailed Description

The main file that manipulates and controls the system's clock.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.16.2 Function Documentation

5.16.2.1 void `get_date` (`date_time` * *dateTimeValues*)

5.16.2.2 int `get_date_main` (int *argc*, char ** *argv*)

5.16.2.3 void `get_time` (`date_time` * *dateTimeValues*)

5.16.2.4 int `get_time_main` (int *argc*, char ** *argv*)

5.16.2.5 `error_t` `set_date` (const `date_time` * *dateTimeValues*)

5.16.2.6 int `set_date_main` (int *argc*, char ** *argv*)

5.16.2.7 int `set_date_str` (const char * *str*)

5.16.2.8 `error_t` `set_time` (const `date_time` * *dateTimeValues*)

5.16.2.9 int `set_time_main` (int *argc*, char ** *argv*)

5.16.2.10 `error_t` `set_time_str` (const char * *timeStr*)

5.17 modules/r2/pcb.c File Reference

The Process Control Block.

Data Structures

- struct `pcb_struct`
Struct that will describe PCB Processes.
- struct `pcb_queue`
Queue structure that will store PCBs.

Variables

- static struct `pcb_queue` `ready_queue`
PCBs stored in priority order with highest priority at head.
- static struct `pcb_queue` `blocked_queue`
PCBs stored in FIFO order.

5.17.1 Detailed Description

The Process Control Block.

Author

Thunder Krakens

Date

March 18th, 2016

Version

R3

5.17.2 Variable Documentation

5.17.2.1 `struct pcb_queue blocked_queue` `[static]`

PCBs stored in FIFO order.

5.17.2.2 `struct pcb_queue ready_queue` `[static]`

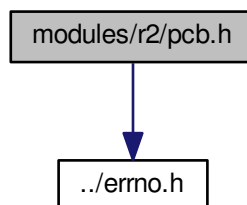
PCBs stored in priority order with highest priority at head.

5.18 `modules/r2/pcb.h` File Reference

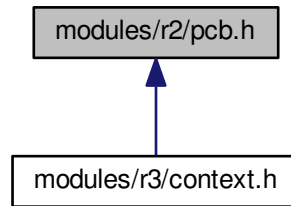
The Process Control Block.

```
#include "../errno.h"
```

Include dependency graph for `pcb.h`:



This graph shows which files directly or indirectly include this file:



Macros

- #define [SIZE_OF_STACK](#) 1024
The default size of the stack for the PCB.
- #define [SIZE_OF_PCB_NAME](#) 10
The max length of the PCB name string.
- #define [COMMHAND_PCB_NAME](#) "commhand"
The name of the command handler PCB.
- #define [IDLE_PCB_NAME](#) "idle"
The name of the idle PCB.

Enumerations

- enum [process_class](#) { [pcb_class_app](#), [pcb_class_sys](#) }
PCB process class types.

Functions

pcb_init

Initializes the PCB queues

- void [pcb_init](#) ()

allocate_pcb

allocate a space for the PCB structure.

Returns

The pointer that point to the PCB structure.

- struct [pcb_struct](#) * [allocate_pcb](#) ()

free_pcb

Frees all memory associated with given PCB, including the PCB itself, the stack, etc, with [sys_free_mem\(\)](#)

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

The appropriate error code. See [errno.h](#) for details. Possible error code to be returned: `E_NOERROR` No error.
`E_INVPARA` The PCB probably had not been removed from queue before free it.

- [error_t](#) `free_pcb` ([struct pcb_struct](#) *pcb_ptr)

setup_pcb

allocate a space for the PCB structure, setup the properties of the PCB.

NOTE: pName must less than 10 character, pClass should be either "application" or "system", and pPriority must within the range of [0, 9].

Parameters

pName	Process Name (length < 10).
pClass	Process class (system or application).
pPriority	Process priority (0 ~ 9).

Returns

The pointer that point to the PCB structure.
NULL if error occured.

- [struct pcb_struct](#) * `setup_pcb` (const char *pName, const enum [process_class](#) pClass, const unsigned char pPriority)

find_pcb

Will search all queues for a process named pName

Parameters

pName	The char pointer to the desired searched name
-------	---

Returns

The PCB pointer.
NULL if PCB is not found

- [struct pcb_struct](#) * `find_pcb` (const char *pName)

insert_pcb

Inserts PCB into the appropriate queue.

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

The appropriate error code. See [errno.h](#) for details. Possible error code to be returned: `E_NOERROR` No error.
`E_NULL_PTR` Null pointer error. `E_INVPARA` The given PCB has running status or abnormal data members.

- [error_t](#) `insert_pcb` ([struct pcb_struct](#) *pcb_ptr)

remove_pcb

Removes PCB from the queue it is currently in.

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

The appropriate error code. See [errno.h](#) for details. Possible error code to be returned: `E_NOERROR` No error. `E_NULL_PTR` Null pointer error. `E_INVPARA` The given PCB has abnormal data members.

- [error_t remove_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

suspend_pcb

Suspends the specific PCB.

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

The appropriate error code. See [errno.h](#) for details. Possible error code to be returned: `E_NOERROR` No error. `E_NULL_PTR` Null pointer error.

- [error_t suspend_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

resume_pcb

Resumes the specific PCB.

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

The appropriate error code. See [errno.h](#) for details. Possible error code to be returned: `E_NOERROR` No error. `E_NULL_PTR` Null pointer error.

- [error_t resume_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

set_pcb_priority

Sets the priority of the selected PCB

Parameters

pcb_ptr	The PCB pointer.
pPriority	The assigned priority

Returns

The appropriate error code. See [errno.h](#) for details. Possible error code to be returned: `E_NOERROR` No error. `E_NULL_PTR` Null pointer error. `E_INVPARA` The pPriority is out of range. Or, the given PCB has abnormal data members (By "remove_pcb" or "insert_pcb").

- [error_t set_pcb_priority](#) (struct [pcb_struct](#) *pcb_ptr, const unsigned char pPriority)

show_pcb

Displays the name, class, state, suspend status, and priority of a PCB.

Parameters

pName	The PCB pointer.
-------	------------------

Returns

The appropriate error code. See [errno.h](#) for details. Possible error code to be returned: `E_NOERROR` No error.
`E_NULL_PTR` Null pointer error.

- [error_t show_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

show_all_processes

Displays all of the processes and their attributes.

- void [show_all_processes](#) ()

show_ready_processes

Displays all of the ready processes and their attributes.

- void [show_ready_processes](#) ()

show_blocked_processes

Displays all blocked processes and their attributes

- void [show_blocked_processes](#) ()

block_pcb

puts the given pcb into the blocked state and places it into the correct queue

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

The appropriate error code. See [errno.h](#) for details. Possible error code to be returned: `E_NOERROR` No error.
`E_NULL_PTR` Null pointer error. `E_INVPARA` The given PCB has abnormal data members (By "remove_pcb" or "insert_pcb").

- [error_t block_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

unblock_pcb

puts the given pcb into the unblocked state and places it into the correct queue

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

The appropriate error code. See [errno.h](#) for details. Possible error code to be returned: `E_NOERROR` No error.
`E_NULL_PTR` Null pointer error. `E_INVPARA` The given PCB has abnormal data members (By "remove_pcb" or "insert_pcb").

- [error_t unblock_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

get_running_process

gets a unsuspended and unblocked process from the front of the queue, and sets it to running state.

Returns

NULL if there is no process available, otherwise, the pointer that point to the PCB structure.

- struct [pcb_struct](#) * [get_running_process](#) ()

save_running_process

sets the running process to ready state, and inserts it to the ready queue.

Parameters

pcb_ptr	The pointer to the PCB.
new_stack_top	The pointer to the new stack top.

Returns

The appropriate error code. See [errno.h](#) for details. Possible error code to be returned: *E_NOERROR* No error. *E_NULL_PTR* Null pointer error. *E_INVPARA* The given PCB has abnormal data members (By "insert_pcb").

- [error_t](#) [save_running_process](#) (struct [pcb_struct](#) *pcb_ptr, struct [context](#) *new_stack_top)

get_stack_top

gets the pointer to the stack top of the specific PCB.

Parameters

pcb_ptr	The pointer to the PCB.
---------	-------------------------

Returns

The pointer that point to the stack top of the specific PCB.
NULL if the pcb_ptr is NULL.

- unsigned char * [get_stack_top](#) (struct [pcb_struct](#) *pcb_ptr)

get_stack_base

gets the pointer to the stack base of the specific PCB.

Parameters

pcb_ptr	The pointer to the PCB.
---------	-------------------------

Returns

The pointer that point to the stack base of the specific PCB.
NULL if the pcb_ptr is NULL.

- unsigned char * [get_stack_base](#) (struct [pcb_struct](#) *pcb_ptr)

shutdown_pcb

called when system is going to shutdown, removes all PCBs, free all PCBs.

- void [shutdown_pcb](#) ()

5.18.1 Detailed Description

The Process Control Block.

Author

Thunder Krakens

Date

February 7th, 2016

Version

R3

5.18.2 Macro Definition Documentation

5.18.2.1 `#define COMMHAND_PCB_NAME "commhand"`

The name of the command handler PCB.

5.18.2.2 `#define IDLE_PCB_NAME "idle"`

The name of the idle PCB.

5.18.2.3 `#define SIZE_OF_PCB_NAME 10`

The max length of the PCB name string.

5.18.2.4 `#define SIZE_OF_STACK 1024`

The default size of the stack for the PCB.

5.18.3 Enumeration Type Documentation

5.18.3.1 `enum process_class`

PCB process class types.

Enumerator

pcb_class_app Process is an application process.

pcb_class_sys Process is a system process.

5.18.4 Function Documentation

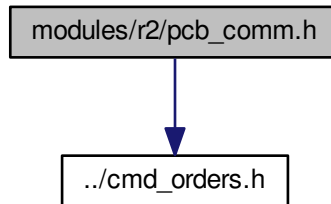
- 5.18.4.1 `struct pcb_struct* allocate_pcb ()`
- 5.18.4.2 `error_t block_pcb (struct pcb_struct * pcb_ptr)`
- 5.18.4.3 `struct pcb_struct* find_pcb (const char * pName)`
- 5.18.4.4 `error_t free_pcb (struct pcb_struct * pcb_ptr)`
- 5.18.4.5 `struct pcb_struct* get_running_process ()`
- 5.18.4.6 `unsigned char* get_stack_base (struct pcb_struct * pcb_ptr)`
- 5.18.4.7 `unsigned char* get_stack_top (struct pcb_struct * pcb_ptr)`
- 5.18.4.8 `error_t insert_pcb (struct pcb_struct * pcb_ptr)`
- 5.18.4.9 `void pcb_init ()`
- 5.18.4.10 `error_t remove_pcb (struct pcb_struct * pcb_ptr)`
- 5.18.4.11 `error_t resume_pcb (struct pcb_struct * pcb_ptr)`
- 5.18.4.12 `error_t save_running_process (struct pcb_struct * pcb_ptr, struct context * new_stack_top)`
- 5.18.4.13 `error_t set_pcb_priority (struct pcb_struct * pcb_ptr, const unsigned char pPriority)`
- 5.18.4.14 `struct pcb_struct* setup_pcb (const char * pName, const enum process_class pClass, const unsigned char pPriority)`
- 5.18.4.15 `void show_all_processes ()`
- 5.18.4.16 `void show_blocked_processes ()`
- 5.18.4.17 `error_t show_pcb (struct pcb_struct * pcb_ptr)`
- 5.18.4.18 `void show_ready_processes ()`
- 5.18.4.19 `void shutdown_pcb ()`
- 5.18.4.20 `error_t suspend_pcb (struct pcb_struct * pcb_ptr)`
- 5.18.4.21 `error_t unblock_pcb (struct pcb_struct * pcb_ptr)`

5.19 modules/r2/pcb_comm.h File Reference

The main functions that manipulate the PCB.

```
#include "../cmd_orders.h"
```

Include dependency graph for pcb_comm.h:



Functions

suspend_pcb_main

The main argument handler for the suspend PCB command.

Accepted formats: `pcb suspend <name>` `pcb suspend -help`

Parameters

argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

Returns

0 when finished execution.

- int [suspend_pcb_main](#) (int argc, char **argv)

resume_pcb_main

The main argument handler for the resume PCB command.

Accepted formats: `pcb resume <name>` `pcb resume -help`

Parameters

argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

Returns

0 when finished execution.

- int [resume_pcb_main](#) (int argc, char **argv)

set_pcb_priority_main

The main argument handler for the set PCB priority command.

Accepted formats: `pcb setpriority <name> <priority>` `pcb setpriority -help`

Parameters

argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

Returns

0 when finished execution.

- int [set_pcb_priority_main](#) (int argc, char **argv)

show_pcb_main

The main argument handler for the show PCB commands.

Accepted formats: pcb show [name] pcb show -all pcb show -ready pcb show -blocked pcb show -help

Parameters

argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

Returns

0 when finished execution.

- int [show_pcb_main](#) (int argc, char **argv)

5.19.1 Detailed Description

The main functions that manipulate the PCB.

Author

Thunder Krakens

Date

February 7th, 2016

Version

R2

5.19.2 Function Documentation

5.19.2.1 int [resume_pcb_main](#) (int *argc*, char ** *argv*)

5.19.2.2 int [set_pcb_priority_main](#) (int *argc*, char ** *argv*)

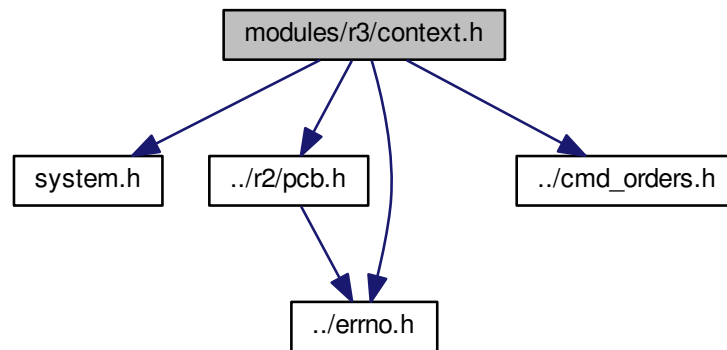
5.19.2.3 int [show_pcb_main](#) (int *argc*, char ** *argv*)

5.19.2.4 int [suspend_pcb_main](#) (int *argc*, char ** *argv*)

5.20 modules/r3/context.h File Reference

Context Switching.

```
#include <system.h>
#include "../r2/pcb.h"
#include "../errno.h"
#include "../cmd_orders.h"
Include dependency graph for context.h:
```



Data Structures

- struct [context](#)

Context structure that holds the 15 CPU register values to begin and resume process execution.

Functions

sys_call

system call interrupt

Parameters

registers	current registers
-----------	-------------------

Returns

result if there is no current process running, it will load new context. If the process is still running, it will load its old context.

- `u32int * sys_call (struct context *registers)`

load_process

loads a process into the PCB.

Parameters

pName	<i>Process Name</i>
pClass	<i>Process Class</i>
pPriority	<i>Process Priority</i>
*function()	<i>A function pointer</i>

Returns

new_pcb Returns the values of the new PCB

- struct [pcb_struct](#) * [load_process](#) (const char *pName, const enum [process_class](#) pClass, const unsigned char pPriority, void(*function)())

load_r3_main

Loads the main function of R3.

Parameters

argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

Returns

0 when finished with execution.

- int [load_r3_main](#) (int argc, char **argv)

Variables

- struct [context](#) * [old_context](#)
- struct [pcb_struct](#) * [cop](#)

5.20.1 Detailed Description

Context Switching.

Author

Thunder Krakens

Date

March 18th, 2016

Version

R3

5.20.2 Function Documentation

5.20.2.1 `struct pcb_struct* load_process (const char * pName, const enum process_class pClass, const unsigned char pPriority, void(*)() function)`

5.20.2.2 `int load_r3_main (int argc, char ** argv)`

5.20.2.3 `u32int* sys_call (struct context * registers)`

5.20.3 Variable Documentation

5.20.3.1 `struct pcb_struct* cop`

5.20.3.2 `struct context* old_context`

5.21 modules/r5/mcb.c File Reference

Memory Control Block.

Data Structures

- struct `cmcb`
Complete Memory Control Block Struct.
- struct `lmcb`
Limited Memory Control Block Struct.
- struct `mcb`
Memory Control Block Struct.

Enumerations

- enum `mcb_type` { `free`, `allocated` }
PCB process class types.

5.21.1 Detailed Description

Memory Control Block.

Author

Thunder Krakens

Date

April 8th, 2016

Version

R5

5.21.2 Enumeration Type Documentation

5.21.2.1 enum mcb_type

PCB process class types.

Enumerator

free Process is an application process.

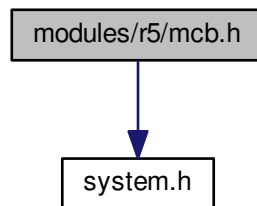
allocated Process is a system process.

5.22 modules/r5/mcb.h File Reference

Memory Control Block.

```
#include <system.h>
```

Include dependency graph for mcb.h:



Macros

- #define `MAX_HEAP_SIZE` 5000
The maximum heap size.

Functions

init_heap

Allocates all the memory for MPX.

Parameters

size	<i>Size of heap in bytes.</i>
------	-------------------------------

- void `init_heap` (u32int size)

mcb_allocate

Allocates a memory block.

Parameters

mem_size	The MCB size to be allocated.
----------	-------------------------------

Returns

Address to allocated MCB.

NULL if not enough space in free memory found.

- void * [mcb_allocate](#) (u32int mem_size)

show_mcb

Displays the allocated or free memory block's address, previous and next pointers, and block's size.

Parameters

mcb_ptr	MCB Pointer
---------	-------------

- void [show_mcb](#) (struct [mcb](#) *mcb_ptr)

show_free_mcb

Displays all the free memory.

- void [show_free_mcb](#) ()

show_allocated_mcb

Displays all the allocated MCBs.

- void [show_allocated_mcb](#) ()

show_all_mcb

Displays all the free and allocated memory.

- void [show_all_mcb](#) ()

is_mcb_empty

Checks if the heap is empty.

Returns

0 or 1 (true or false).

- int [is_mcb_empty](#) ()

mcb_free_mpx

Calls [mcb_free](#) to free memory block, used as parameter for [sys_set_free](#) in [kmain.c](#).

Parameters

mem_ptr	Memory Pointer
---------	----------------

Returns

0 when finished with execution.

- int [mcb_free_mpx](#) (void *mem_ptr)

mcb_allocate_mpx

Calls [mcb_allocate](#) to allocate memory block, used as parameter for [sys_set_malloc](#) in [kmain.c](#).

Parameters

size	Size of block in bytes to allocate.
------	-------------------------------------

Returns

Address of allocated MCB.

- [u32int mcb_allocate_mpx](#) ([u32int](#) size)

mcb_allocate_mpx2

MCB allocate MPX.

Parameters

mem_size	Block size to allocate.
name	Name of the pcb process.

Returns

Address pointer to allocated memory only used for testing in commhand for module R5.

- void * [mcb_allocate_mpx2](#) ([u32int](#) size, const char *name)

show_mcb_main.

The function of show MCB for commhand.

Parameters

argc	The number of tokens found.
argv	The array of tokens.

Returns

0 when finished with execution.

- int [show_mcb_main](#) (int argc, char **argv)

shutdown_mcb.

Shutdown the pcb during the shutdown procedure.

Returns

0 when finished with execution.

- void [shutdown_mcb](#) ()

Variables

- [u32int start_of_memory](#)

Global variable labeling start of memory.

5.22.1 Detailed Description

Memory Control Block.

Author

Thunder Krakens

Date

April 8th, 2016

Version

R5

5.22.2 Macro Definition Documentation

5.22.2.1 #define MAX_HEAP_SIZE 5000

The maximum heap size.

5.22.3 Function Documentation

5.22.3.1 void init_heap (u32int size)

5.22.3.2 int is_mcb_empty ()

5.22.3.3 void* mcb_allocate (u32int mem_size)

5.22.3.4 u32int mcb_allocate_mpx (u32int size)

5.22.3.5 void* mcb_allocate_mpx2 (u32int size, const char * name)

5.22.3.6 int mcb_free_mpx (void * mem_ptr)

5.22.3.7 void show_all_mcb ()

5.22.3.8 void show_allocated_mcb ()

5.22.3.9 void show_free_mcb ()

5.22.3.10 void show_mcb (struct mcb * mcb_ptr)

5.22.3.11 int show_mcb_main (int argc, char ** argv)

5.22.3.12 void shutdown_mcb ()

5.22.4 Variable Documentation

5.22.4.1 u32int start_of_memory

Global variable labeling start of memory.

5.23 modules/r6/ansi.h File Reference

Macros

- `#define T_RESET ""`
- `#define T_BOLD ""`
- `#define T_BOLD_OFF ""`
- `#define T_ITCS ""`
- `#define T_ITCS_OFF ""`
- `#define T_NRM ""`
- `#define T_RED ""`
- `#define T_CYAN ""`
- `#define T_WHT ""`
- `#define B_NRM ""`
- `#define B_CYAN ""`
- `#define T_DIR ""`
- `#define T_DIR_OFF ""`

5.23.1 Macro Definition Documentation

5.23.1.1 `#define B_CYAN ""`

5.23.1.2 `#define B_NRM ""`

5.23.1.3 `#define T_BOLD ""`

5.23.1.4 `#define T_BOLD_OFF ""`

5.23.1.5 `#define T_CYAN ""`

5.23.1.6 `#define T_DIR ""`

5.23.1.7 `#define T_DIR_OFF ""`

5.23.1.8 `#define T_ITCS ""`

5.23.1.9 `#define T_ITCS_OFF ""`

5.23.1.10 `#define T_NRM ""`

5.23.1.11 `#define T_RED ""`

5.23.1.12 `#define T_RESET ""`

5.23.1.13 `#define T_WHT ""`

5.24 modules/r6/disk_file_manager.h File Reference

Disk File Manager.

Functions

type_file

Prints the contents of a file

This function will print any contents of a file with the extensions "TXT", "BAT", "C", or "HTM" in pagination form.

Parameters

file_entry_ptr	The pointer to the file entry
----------------	-------------------------------

Returns

The appropriate error code. See [errno.h](#) for details.

- [error_t type_file](#) (struct [dir_entry_info](#) *file_entry_ptr)

extract_file

Extracts a file's data contents into new file outside of the disk image.

Parameters

file_entry_ptr	The pointer to the file entry
out_file_path	The destination to save the new 'copied' file

Returns

The appropriate error code. See [errno.h](#) for details.

- [error_t extract_file](#) (struct [dir_entry_info](#) *file_entry_ptr, const char *out_file_path)

import_file

Imports a file into a directory

Parameters

in_file_path	Where the file to be imported is located
dest_dir	The directory destination for the imported file

Returns

The appropriate error code. See [errno.h](#) for details.

- [error_t import_file](#) (const char *in_file_path, struct [dir_entry_info](#) *dest_dir)

move_file

Moves files from destination to another

Parameters

file_entry	The file entry pointer to be moved
dest_dir	The destination of the moved file

Returns

The appropriate error code. See [errno.h](#) for details.

- [error_t move_file](#) (struct [dir_entry_info](#) *file_entry, struct [dir_entry_info](#) *dest_dir)
- [error_t delete_file](#) (struct [dir_entry_info](#) *file_entry)

5.24.1 Detailed Description

Disk File Manager.

Author

Thunder Krakens

Date

April 28th, 2016 This contains functions that deal with files

Version

R6

5.24.2 Function Documentation

5.24.2.1 `error_t delete_file (struct dir_entry_info * file_entry)`

5.24.2.2 `error_t extract_file (struct dir_entry_info * file_entry_ptr, const char * out_file_path)`

5.24.2.3 `error_t import_file (const char * in_file_path, struct dir_entry_info * dest_dir)`

5.24.2.4 `error_t move_file (struct dir_entry_info * file_entry, struct dir_entry_info * dest_dir)`

5.24.2.5 `error_t type_file (struct dir_entry_info * file_entry_ptr)`

5.25 modules/r6/disk_folder_manager.h File Reference

Disk Folder Manager.

Macros

- `#define FOLDER_STACK_SIZE 1000`

Functions

folder_manager_init

Initializes the folder manager

Initializes the folder stack. Reads the boot sector's volume label and sets it as the root folder.

- `void folder_manager_init ()`

push_folder

Pushes the current folder onto the folder stack

Pushes current onto the folder stack and sets the child folder as the current folder.

Parameters

child_folder_ptr	the pointer to a child folder of the current folder
------------------	---

- void [push_folder](#) (struct [dir_entry_info](#) *child_folder_ptr)

pop_folder

Pops folder off the folder stack

The current folder is now set to the previous folder from the popped folder.

- void [pop_folder](#) ()

print_dir_entry_info

Prints the file/directory's detailed information

Displays the filename, extension, logical file size, and starting cluster of the specified file/directory.

Parameters

folder_ptr	pointer to the file entry
------------	---------------------------

- void [print_dir_entry_info](#) (struct [dir_entry_info](#) *entry_ptr)

list_dir_entry_report

Prints all content's detailed information of the current directory

Displays the filename, extension, logical file size, and starting cluster of all the files/directories contained in the current directory

- void [list_dir_entry_report](#) ()

list_dir_entry_short

Displays all contents in the current directory

Displays all the filenames and/or directories in the current folder.

- void [list_dir_entry_short](#) ()

print_curr_path

Displays the current directory's path

- void [print_curr_path](#) ()

rename_entry

Renames indicated file/folder's name

Ensures that it is a unique file name by checking all of the file names in the current file before renaming it.

Parameters

parent_dir_entry	A pointer to the current directory
file_entry	A pointer to the file entry to rename

new_name	<i>The file entry's new name</i>
----------	----------------------------------

Returns

The appropriate error code. See [errno.h](#) for details.

- [error_t](#) [rename_entry](#) (struct [dir_entry_info](#) *parent_dir_entry, struct [dir_entry_info](#) *file_entry, char *new_name)

get_entry_by_name

Retrieves the file/directory by name

Parameters

parent_dir_entry	<i>A pointer to the current directory of the file</i>
nameStr	<i>The name of the file/directory to retrieve</i>

Returns

Returns a pointer to the correct [dir_entry_info](#) struct or NULL if it is not found

- struct [dir_entry_info](#) * [get_entry_by_name](#) (const struct [dir_entry_info](#) *parent_dir_entry, char *nameStr)

get_entry

Retrieves the file/directory by it's full file path

Parameters

full_path	<i>The full file path of the</i>
-----------	----------------------------------

- struct [dir_entry_info](#) * [get_entry](#) (char *full_path)

change_dir

Change the current directory to the specified path

Parameters

full_path	<i>The path to the new location</i>
-----------	-------------------------------------

- void [change_dir](#) (char *full_path)

list_files_entry_ext

Lists all the files with the indicated extension

Used when the wildcard '*' is indicated as the filename. (Ex: *.txt will list all the files with the 'txt' extension)

Parameters

ext	<i>extension of entry</i>
-----	---------------------------

- void [list_files_entry_ext](#) (const char *ext)

list_files_entry_name

Lists all the files with the indicated name

Used when the wildcard '*' is indicated as the extension. (Ex: filename.* will list all the files with the name 'filename')

Parameters

name	<i>Name of the file</i>
------	-------------------------

- void [list_files_entry_name](#) (const char *name)

list_file_report

Lists the indicated file's full details

*Displays the filename, extension, logical file size, and starting cluster of the indicated name and extension. If wildcard is true (1) then it will list the full details of the indicated. (Example. *.txt will list the full report of all files with the 'txt' extension)*

Parameters

name	<i>name of file</i>
ext	<i>extension of file</i>
wildcard	<i>1 or 0 (true or false)</i>

- void [list_file_report](#) (const char *name, const char *ext, int wildcard)

print_report_heading

Prints the heading of the report.

- void [print_report_heading](#) ()

5.25.1 Detailed Description

Disk Folder Manager.

Author

Thunder Krakens

Date

April 28th, 2016 This contains functions that are related to the directory/folder.

Version

R6

5.25.2 Macro Definition Documentation

5.25.2.1 `#define FOLDER_STACK_SIZE 1000`

5.25.3 Function Documentation

5.25.3.1 `void change_dir (char * full_path)`

5.25.3.2 `void folder_manager_init ()`

5.25.3.3 `struct dir_entry_info* get_entry (char * full_path)`

5.25.3.4 struct dir_entry_info* get_entry_by_name (const struct dir_entry_info * *parent_dir_entry*, char * *nameStr*)

5.25.3.5 void list_dir_entry_report ()

5.25.3.6 void list_dir_entry_short ()

5.25.3.7 void list_file_report (const char * *name*, const char * *ext*, int *wildcard*)

5.25.3.8 void list_files_entry_ext (const char * *ext*)

5.25.3.9 void list_files_entry_name (const char * *name*)

5.25.3.10 void pop_folder ()

5.25.3.11 void print_curr_path ()

5.25.3.12 void print_dir_entry_info (struct dir_entry_info * *entry_ptr*)

5.25.3.13 void print_report_heading ()

5.25.3.14 void push_folder (struct dir_entry_info * *child_folder_ptr*)

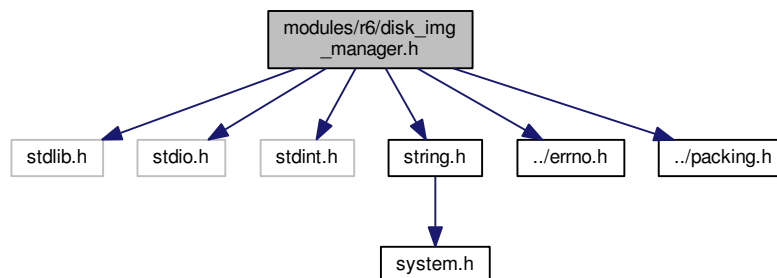
5.25.3.15 error_t rename_entry (struct dir_entry_info * *parent_dir_entry*, struct dir_entry_info * *file_entry*, char * *new_name*)

5.26 modules/r6/disk_img_manager.h File Reference

Disk Image Manager.

```
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include "../errno.h"
#include "../packing.h"
```

Include dependency graph for disk_img_manager.h:



Data Structures

- struct [img_boot_sector](#)
Structure containing the Boot Sector's information and data.
- struct [dir_entry_info](#)
Structure containing the directory/file entry's information and data in sector.
- struct [data_sector](#)
Structure containing the sector's data.
- struct [fat_date](#)
Structure containing the date information.
- struct [fat_time](#)
Structure containing the time information.

Macros

- #define [ATTRIBUTE_READ](#) 0x01
- #define [ATTRIBUTE_HIDD](#) 0x02
- #define [ATTRIBUTE_SYST](#) 0x04
- #define [ATTRIBUTE_VOLL](#) 0x08
- #define [ATTRIBUTE_SUBD](#) 0x10
- #define [ATTRIBUTE_ARCH](#) 0x20
- #define [ATTRIBUTE_UUS1](#) 0x40
- #define [ATTRIBUTE_UUS2](#) 0x80
- #define [ATTR_ARCH_ONLY](#) (![ATTRIBUTE_SYST](#) & ![ATTRIBUTE_VOLL](#) & ![ATTRIBUTE_SUBD](#) & ![ATTRIBUTE_UUS1](#) & ![ATTRIBUTE_UUS2](#))

Functions

load_image_file

Loads the indeicated image file

Parameters

<code>path_to_file</code>	<i>String path to the image file</i>
---------------------------	--------------------------------------

Returns

The appropriate error code. See [errno.h](#) for details.

- [error_t load_image_file](#) (const char *path_to_file)

write_image_file

Writes to image file

Will write all changes made to the image

Returns

The appropriate error code. See [errno.h](#) for details.

- [error_t write_image_file](#) ()

print_boot_sec_info

Displays the boot sector information

Parameters

boot_sec	Pointer to the boot sector
----------	----------------------------

- void [print_boot_sec_info](#) (const struct [img_boot_sector](#) *boot_sec)

clean_buffer*Clears the buffer*

- void [clean_buffers](#) ()

ch_arr_to_str*Converts character array to a string**Parameters*

dest	Variable to hold converted string
src	Source char array
size	size of char array

- void [ch_arr_to_str](#) (char *dest, const char *src, const unsigned int size)

str_to_ch_arr*Convert string to character array**Parameters*

dest	Variable to hold converted char array
src	Source string
size	size of string

- void [str_to_ch_arr](#) (char *dest, const char *src, const unsigned int size)

get_fat_val*Retrieve FAT values**Parameters*

copy_index	Copies FAT Index
byte_index	Retrieves index in bytes

- uint8_t * [get_fat_val](#) (const unsigned int copy_index, const unsigned int byte_index)

fat*Retrieve specified 12-bit value from the FAT array.**Converts 2*1 byte to 12 bit.**Parameters*

fat_val	FAT Value
cluster_index	Cluster Index

- void [fat](#) (uint16_t *fat_val, const uint16_t cluster_index)

get_data_ptr*Retrieves the memory address of the data*

Parameters

data_area_sec_index	Data sector index
---------------------	-------------------

- void * [get_data_ptr](#) (const uint16_t data_area_sec_index)

write_fat

Writes the FAT value at the cluster index

Parameters

fat_val	FAT value
cluster_index	Cluster index

- void [write_fat](#) (const uint16_t fat_val, const uint16_t cluster_index)

find_unused_fat

Retrieves unused FAT index

Returns

the index of the FAT value for the fat array

- uint16_t [find_unused_fat](#) ()

calc_free_space

Calculates the free space in FAT

Returns

the value of free space.

- uint64_t [calc_free_space](#) ()

str_to_upper_case

Converts string characters to uppercase

Parameters

str	String to conver to uppercase
len	The length of the string

- void [str_to_upper_case](#) (char *str, const unsigned int len)

separate_file_name

Separates the filename string by name and extension.

Parameters

full_name	The full name of the file (Example: file.ext)
file_name	The variable to store the filename
file_ext	The variable to store the file's extension

Returns

The appropriate error code. See [errno.h](#) for details.

- [error_t seperate_file_name](#) (const char *full_name, char *file_name, char *file_ext)

get_fat_date

Stores FAT's date

Parameters

out_date	<i>A pointer to the struct that will hold the date</i>
fat_date_value	<i>The FAT date value</i>

- void [get_fat_date](#) (struct [fat_date](#) *out_date, const uint16_t fat_date_value)

get_fat_date_str

Stores the FAT's date in a string

Parameters

out_str	<i>The string to store the date's information</i>
fat_date_value	<i>The FAT date value</i>

- void [get_fat_date_str](#) (char *out_str, const uint16_t fat_date_value)

set_fat_time

Sets the FAT time

Parameters

in_time	<i>A pointer the time structure containing the information</i>
out_fat_time_value	<i>The FAT time value</i>

- void [set_fat_time](#) (const struct [fat_time](#) *in_time, uint16_t *out_fat_time_value)

get_fat_time

Stores the FAT time

Parameters

out_time	<i>A pointer to the structure containing the time</i>
fat_date_value	<i>The FAT time value</i>

- void [get_fat_time](#) (struct [fat_time](#) *out_time, const uint16_t fat_time_value)

get_fat_time_str

Stores the FAT's time in a string

Parameters

out_str	<i>The string to store the time's information</i>
fat_date_value	<i>The FAT date value</i>

- void [get_fat_time_str](#) (char *out_str, const uint16_t fat_time_value)

Variables

- struct [img_boot_sector](#) * [boot_sec](#)
- struct [dir_entry_info](#) * [root_dir_file_arr](#)
- struct [data_sector](#) * [data_area](#)
- struct [dir_entry_info](#) * [root_dir_entry](#)

5.26.1 Detailed Description

Disk Image Manager.

Author

Thunder Krakens

Date

April 28th, 2016

Version

R6

5.26.2 Macro Definition Documentation

5.26.2.1 `#define ATTR_ARCH_ONLY (!ATTRIBUTE_SYST & !ATTRIBUTE_VOLL & !ATTRIBUTE_SUBD & !ATTRIBUTE_UUS1 & !ATTRIBUTE_UUS2)`

5.26.2.2 `#define ATTRIBUTE_ARCH 0x20`

5.26.2.3 `#define ATTRIBUTE_HIDD 0x02`

5.26.2.4 `#define ATTRIBUTE_READ 0x01`

5.26.2.5 `#define ATTRIBUTE_SUBD 0x10`

5.26.2.6 `#define ATTRIBUTE_SYST 0x04`

5.26.2.7 `#define ATTRIBUTE_UUS1 0x40`

5.26.2.8 `#define ATTRIBUTE_UUS2 0x80`

5.26.2.9 `#define ATTRIBUTE_VOLL 0x08`

5.26.3 Function Documentation

5.26.3.1 `uint64_t calc_free_space ()`

5.26.3.2 `void ch_arr_to_str (char * dest, const char * src, const unsigned int size)`

5.26.3.3 `void clean_buffers ()`

5.26.3.4 `void fat (uint16_t * fat_val, const uint16_t cluster_index)`

5.26.3.5 `uint16_t find_unused_fat ()`

5.26.3.6 `void* get_data_ptr (const uint16_t data_area_sec_index)`

5.26.3.7 `void get_fat_date (struct fat_date * out_date, const uint16_t fat_date_value)`

5.26.3.8 void get_fat_date_str (char * *out_str*, const uint16_t *fat_date_value*)

5.26.3.9 void get_fat_time (struct fat_time * *out_time*, const uint16_t *fat_time_value*)

5.26.3.10 void get_fat_time_str (char * *out_str*, const uint16_t *fat_time_value*)

5.26.3.11 uint8_t* get_fat_val (const unsigned int *copy_index*, const unsigned int *byte_index*)

5.26.3.12 error_t load_image_file (const char * *path_to_file*)

5.26.3.13 void print_boot_sec_info (const struct img_boot_sector * *boot_sec*)

5.26.3.14 error_t seperate_file_name (const char * *full_name*, char * *file_name*, char * *file_ext*)

5.26.3.15 void set_fat_time (const struct fat_time * *in_time*, uint16_t * *out_fat_time_value*)

5.26.3.16 void str_to_ch_arr (char * *dest*, const char * *src*, const unsigned int *size*)

5.26.3.17 void str_to_upper_case (char * *str*, const unsigned int *len*)

5.26.3.18 void write_fat (const uint16_t *fat_val*, const uint16_t *cluster_index*)

5.26.3.19 error_t write_image_file ()

5.26.4 Variable Documentation

5.26.4.1 struct img_boot_sector* boot_sec

5.26.4.2 struct data_sector* data_area

5.26.4.3 struct dir_entry_info* root_dir_entry

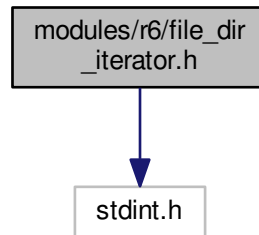
5.26.4.4 struct dir_entry_info* root_dir_file_arr

5.27 modules/r6/file_dir_iterator.h File Reference

File Directory Iterator.

```
#include <stdint.h>
```

Include dependency graph for file_dir_iterator.h:



Macros

- `#define` [ROOT_DIR_SEC_INDEX](#) 0

Functions

`init_file_itr`

Initializes the file iterator at the specified sector index

Returns

A pointer to the initialized file_itr

- `struct file_itr * init_file_itr (const uint16_t sec_index)`

`fitr_begin`

Starts the iterator

Parameters

<code>itr_ptr</code>	<i>Pointer of the iterator</i>
----------------------	--------------------------------

- `void fitr_begin (struct file_itr *itr_ptr)`

`fitr_end`

Checks if the iterator has reached the end of FAT (the last cluster in the file)

Parameters

<code>itr_ptr</code>	<i>Pointer of the iterator</i>
----------------------	--------------------------------

Returns

0 or 1 (true or false)

- uint8_t [fitr_end](#) (struct file_itr *itr_ptr)

fitr_next

Jumps to the next section in FAT

Parameters

itr_ptr	Pointer of the iterator
---------	-------------------------

- void [fitr_next](#) (struct file_itr *itr_ptr)

fitr_get

Retrieves the data at the iterator

Parameters

itr_ptr	Pointer of the iterator
---------	-------------------------

- struct [data_sector](#) * [fitr_get](#) (struct file_itr *itr_ptr)

init_dir_itr

Initializes the directory iterator at the specified sector's index

Parameters

sec_index	Sector index
-----------	--------------

- struct [dir_itr](#) * [init_dir_itr](#) (const uint16_t sec_index)

dirtr_set_filter

Set's the directory iterator's attribute filter

Parameters

itr_ptr	Pointer of the iterator
attr_filter	Attribute filter

- void [dirtr_set_filter](#) (struct [dir_itr](#) *itr_ptr, uint8_t attr_filter)

dirtr_begin

Starts the directory iterator

Parameters

itr_ptr	Pointer of the iterator
---------	-------------------------

- void [dirtr_begin](#) (struct [dir_itr](#) *itr_ptr)

dirtr_set_find_unused

Set up a search in directory iterator

Parameters

itr_ptr	Pointer of the iterator
---------	-------------------------

- void [dirtr_set_find_unused](#) (struct [dir_itr](#) *itr_ptr)

dirtr_end

Check's if the iterator has reach to the last cluster in the file

Parameters

itr_ptr	Pointer of the iterator
---------	-------------------------

- uint8_t [ditr_end](#) (struct [dir_itr](#) *itr_ptr)

ditr_next

Finds the next entry in directory

Parameters

itr_ptr	Pointer of the iterator
---------	-------------------------

- void [ditr_next](#) (struct [dir_itr](#) *itr_ptr)

ditr_get

Gets the pointer that point to the current entry.

If the iterator had approached to the end, this will return NULL instead;

Parameters

itr_ptr	Pointer of the iterator
---------	-------------------------

Returns

The pointer that point to the current entry. NULL if reached to the end.

- struct [dir_entry_info](#) * [ditr_get](#) (struct [dir_itr](#) *itr_ptr)

init_img_writer

Initialize the image writer

Parameters

entry_ptr	The pointer of entry that this writer will write with.
-----------	--

- struct [img_writer](#) * [init_img_writer](#) (struct [dir_entry_info](#) *entry_ptr)

iw_write

Function to write the image

Parameters

writer_ptr	Pointer of the writer
data	The new data sector that needs to write.

- void [iw_write](#) (struct [img_writer](#) *writer_ptr, const struct [data_sector](#) *data)

5.27.1 Detailed Description

File Directory Iterator.

Author

Thunder Krakens

Date

April 28th, 2016

Version

R6

5.27.2 Macro Definition Documentation

5.27.2.1 #define ROOT_DIR_SEC_INDEX 0

5.27.3 Function Documentation

5.27.3.1 void dtr_begin (struct dir_itr * itr_ptr)

5.27.3.2 uint8_t dtr_end (struct dir_itr * itr_ptr)

5.27.3.3 struct dir_entry_info* dtr_get (struct dir_itr * itr_ptr)

5.27.3.4 void dtr_next (struct dir_itr * itr_ptr)

5.27.3.5 void dtr_set_filter (struct dir_itr * itr_ptr, uint8_t attr_filter)

5.27.3.6 void dtr_set_find_unused (struct dir_itr * itr_ptr)

5.27.3.7 void fitr_begin (struct file_itr * itr_ptr)

5.27.3.8 uint8_t fitr_end (struct file_itr * itr_ptr)

5.27.3.9 struct data_sector* fitr_get (struct file_itr * itr_ptr)

5.27.3.10 void fitr_next (struct file_itr * itr_ptr)

5.27.3.11 struct dir_itr* init_dir_itr (const uint16_t sec_index)

5.27.3.12 struct file_itr* init_file_itr (const uint16_t sec_index)

5.27.3.13 struct img_writer* init_img_writer (struct dir_entry_info * entry_ptr)

5.27.3.14 void iw_write (struct img_writer * writer_ptr, const struct data_sector * data)

Index

- [_kmalloc](#)
 - [heap.h, 39](#)
- [ATTR_ARCH_ONLY](#)
 - [disk_img_manager.h, 90](#)
- [ATTRIBUTE_ARCH](#)
 - [disk_img_manager.h, 90](#)
- [ATTRIBUTE_HIDD](#)
 - [disk_img_manager.h, 90](#)
- [ATTRIBUTE_READ](#)
 - [disk_img_manager.h, 90](#)
- [ATTRIBUTE_SUBD](#)
 - [disk_img_manager.h, 90](#)
- [ATTRIBUTE_SYST](#)
 - [disk_img_manager.h, 90](#)
- [ATTRIBUTE_UUS1](#)
 - [disk_img_manager.h, 90](#)
- [ATTRIBUTE_UUS2](#)
 - [disk_img_manager.h, 90](#)
- [ATTRIBUTE_VOLL](#)
 - [disk_img_manager.h, 90](#)
- [access](#)
 - [gdt_entry, 17](#)
- [accessed](#)
 - [page_entry, 28](#)
- [alloc](#)
 - [heap.h, 39](#)
- [allocate_pcb](#)
 - [pcb.h, 69](#)
- [allocated](#)
 - [mcb.c, 75](#)
- [ansi.h](#)
 - [B_CYAN, 79](#)
 - [B_NRM, 79](#)
 - [T_BOLD, 79](#)
 - [T_BOLD_OFF, 79](#)
 - [T_CYAN, 79](#)
 - [T_DIR, 79](#)
 - [T_DIR_OFF, 79](#)
 - [T_ITCS, 79](#)
 - [T_ITCS_OFF, 79](#)
 - [T_NRM, 79](#)
 - [T_RED, 79](#)
 - [T_RESET, 79](#)
 - [T_WHT, 79](#)
- [asm](#)
 - [system.h, 46](#)
- [atoi](#)
 - [string.h, 44](#)
- [attributes](#)
 - [dir_entry_info, 12](#)
- [B_CYAN](#)
 - [ansi.h, 79](#)
- [B_NRM](#)
 - [ansi.h, 79](#)
- [base](#)
 - [gdt_descriptor, 16](#)
 - [heap, 18](#)
 - [idt_descriptor, 19](#)
- [base_high](#)
 - [gdt_entry, 17](#)
 - [idt_entry, 19](#)
- [base_low](#)
 - [gdt_entry, 17](#)
 - [idt_entry, 19](#)
- [base_mid](#)
 - [gdt_entry, 17](#)
- [begin_address](#)
 - [cmcb, 7](#)
- [block](#)
 - [index_entry, 23](#)
- [block_pcb](#)
 - [pcb.h, 69](#)
- [blocked_queue](#)
 - [pcb.c, 62](#)
- [boot_sec](#)
 - [disk_img_manager.h, 91](#)
- [boot_sign](#)
 - [img_boot_sector, 21](#)
- [byte_per_sector](#)
 - [img_boot_sector, 21](#)
- [COM1](#)
 - [serial.h, 36](#)
- [COM2](#)
 - [serial.h, 36](#)
- [COM3](#)
 - [serial.h, 36](#)
- [COM4](#)
 - [serial.h, 36](#)
- [COMMHAND_PCB_NAME](#)

- pcb.h, 68
- calc_free_space
 - disk_img_manager.h, 90
- ch_arr_to_str
 - disk_img_manager.h, 90
- change_dir
 - disk_folder_manager.h, 84
- class
 - pcb_struct, 31
- clean_buffers
 - disk_img_manager.h, 90
- clear_bit
 - paging.h, 40
- cli
 - system.h, 46
- cmcb, 7
 - begin_address, 7
 - size, 7
 - type, 7
- cmd_orders.h
 - FUNCTIONS_BEGIN, 48
 - GETDATE, 48
 - GETTIME, 48
 - HELP, 48
 - LOADR3, 48
 - MCB_FUNC_END, 48
 - MPX_FUNC_END, 48
 - NUM_OF_FUNCTIONS, 48
 - PCB_FUNC_END, 48
 - RESUMEPCB, 48
 - SETDATE, 48
 - SETPCBPRIO, 48
 - SETTIME, 48
 - SHOWMCB, 48
 - SHOWPCB, 48
 - SHUTDOWN, 49
 - SUSPDPCB, 49
 - VERSION, 49
 - WITH_R2_TEMP_CMD, 49
 - WITH_R3_TEMP_CMD, 49
 - WITH_R5_TEMP_CMD, 49
- comm_type
 - r1.h, 57
- command_line_parser
 - r1.h, 58
- commhand
 - r1.h, 58
- complete_mcb
 - mcb, 26
- context, 8
 - cs, 9
 - ds, 9
 - eax, 9
 - ebp, 9
 - ebx, 9
 - ecx, 9
 - edi, 9
 - edx, 9
 - eflags, 9
 - eip, 9
 - es, 9
 - esi, 9
 - esp, 10
 - fs, 10
 - gs, 10
- context.h
 - cop, 74
 - load_process, 74
 - load_r3_main, 74
 - old_context, 74
 - sys_call, 74
- cop
 - context.h, 74
- count
 - pcb_queue, 30
- create_date
 - dir_entry_info, 12
- create_time
 - dir_entry_info, 12
- cs
 - context, 9
- data
 - data_sector, 10
- data_area
 - disk_img_manager.h, 91
- data_sector, 10
 - data, 10
- date_time, 11
 - day_m, 11
 - day_w, 11
 - day_y, 11
 - hour, 11
 - min, 11
 - mon, 11
 - sec, 11
 - year, 11
- day
 - fat_date, 14
- day_m
 - date_time, 11
- day_w
 - date_time, 11
- day_y
 - date_time, 11
- delete_file
 - disk_file_manager.h, 81
- device_id

- param, 29
- dir_entry_info, 11
 - attributes, 12
 - create_date, 12
 - create_time, 12
 - extension, 12
 - file_name, 13
 - file_size, 13
 - first_log_clu, 13
 - ignore1, 13
 - last_acc_date, 13
 - last_wri_date, 13
 - last_wri_time, 13
 - reserved, 13
- dir_itr, 14
- dirty
 - page_entry, 28
- disk_file_manager.h
 - delete_file, 81
 - extract_file, 81
 - import_file, 81
 - move_file, 81
 - type_file, 81
- disk_folder_manager.h
 - change_dir, 84
 - folder_manager_init, 84
 - get_entry, 84
 - get_entry_by_name, 84
 - list_dir_entry_report, 85
 - list_dir_entry_short, 85
 - list_file_report, 85
 - list_files_entry_ext, 85
 - list_files_entry_name, 85
 - pop_folder, 85
 - print_curr_path, 85
 - print_dir_entry_info, 85
 - print_report_heading, 85
 - push_folder, 85
 - rename_entry, 85
- disk_img_manager.h
 - ATTR_ARCH_ONLY, 90
 - ATTRIBUTE_ARCH, 90
 - ATTRIBUTE_HIDD, 90
 - ATTRIBUTE_READ, 90
 - ATTRIBUTE_SUBD, 90
 - ATTRIBUTE_SYST, 90
 - ATTRIBUTE_UUS1, 90
 - ATTRIBUTE_UUS2, 90
 - ATTRIBUTE_VOLL, 90
 - boot_sec, 91
 - calc_free_space, 90
 - ch_arr_to_str, 90
 - clean_buffers, 90
 - data_area, 91
 - fat, 90
 - find_unused_fat, 90
 - get_data_ptr, 90
 - get_fat_date, 90
 - get_fat_date_str, 90
 - get_fat_time, 91
 - get_fat_time_str, 91
 - get_fat_val, 91
 - load_image_file, 91
 - print_boot_sec_info, 91
 - root_dir_entry, 91
 - root_dir_file_arr, 91
 - seperate_file_name, 91
 - set_fat_time, 91
 - str_to_ch_arr, 91
 - str_to_upper_case, 91
 - write_fat, 91
 - write_image_file, 91
- ditr_begin
 - file_dir_iterator.h, 96
- ditr_end
 - file_dir_iterator.h, 96
- ditr_get
 - file_dir_iterator.h, 96
- ditr_next
 - file_dir_iterator.h, 96
- ditr_set_filter
 - file_dir_iterator.h, 96
- ditr_set_find_unused
 - file_dir_iterator.h, 96
- documentation/mainpage.dox, 33
- ds
 - context, 9
- E_EMPTPCB
 - errno.h, 50
- E_FILE_NF
 - errno.h, 50
- E_FOLDFUL
 - errno.h, 50
- E_FREEMEM
 - errno.h, 51
- E_INVATTRS
 - errno.h, 51
- E_INVPARA
 - errno.h, 51
- E_INVSTRF
 - errno.h, 51
- E_INVUSRI
 - errno.h, 51
- E_NAMEDUP
 - errno.h, 51
- E_NAMEINV
 - errno.h, 51

E_NOERROR
 errno.h, 51
 E_NOSPACE
 errno.h, 51
 E_NULL_PTR
 errno.h, 51
 E_PCB_SYS
 errno.h, 51
 E_PROGERR
 errno.h, 51
 EXIT
 mpx_supt.h, 54
 eax
 context, 9
 ebp
 context, 9
 ebx
 context, 9
 ecx
 context, 9
 edi
 context, 9
 edx
 context, 9
 eflags
 context, 9
 eip
 context, 9
 empty
 index_entry, 23
 errno.h
 E_EMPTPCB, 50
 E_FILE_NF, 50
 E_FOLDFUL, 50
 E_FREEMEM, 51
 E_INVATTRS, 51
 E_INVPARA, 51
 E_INVSTRF, 51
 E_INVUSRI, 51
 E_NAMEDUP, 51
 E_NAMEINV, 51
 E_NOERROR, 51
 E_NOSPACE, 51
 E_NULL_PTR, 51
 E_PCB_SYS, 51
 E_PROGERR, 51
 error_t, 51
 error_t
 errno.h, 51
 es
 context, 9
 esi
 context, 9
 esp
 context, 10
 extension
 dir_entry_info, 12
 extract_file
 disk_file_manager.h, 81
 FOLDER_STACK_SIZE
 disk_folder_manager.h, 84
 FUNCTIONS_BEGIN
 cmd_orders.h, 48
 fat
 disk_img_manager.h, 90
 fat_copies_num
 img_boot_sector, 21
 fat_date, 14
 day, 14
 mon, 14
 year, 14
 fat_time, 15
 hr, 15
 mi, 15
 se, 15
 file_dir_iterator.h
 ditr_begin, 96
 ditr_end, 96
 ditr_get, 96
 ditr_next, 96
 ditr_set_filter, 96
 ditr_set_find_unused, 96
 fitr_begin, 96
 fitr_end, 96
 fitr_get, 96
 fitr_next, 96
 init_dir_itr, 96
 init_file_itr, 96
 init_img_writer, 96
 iw_write, 96
 file_iter, 15
 file_name
 dir_entry_info, 13
 file_size
 dir_entry_info, 13
 file_sys_type
 img_boot_sector, 21
 find_pcb
 pcb.h, 69
 find_unused_fat
 disk_img_manager.h, 90
 first_free
 paging.h, 40
 first_log_clu
 dir_entry_info, 13
 fitr_begin
 file_dir_iterator.h, 96

- fitr_end
 - file_dir_iterator.h, 96
- fitr_get
 - file_dir_iterator.h, 96
- fitr_next
 - file_dir_iterator.h, 96
- flags
 - gdt_entry, 17
 - idt_entry, 19
- folder_manager_init
 - disk_folder_manager.h, 84
- footer, 16
 - head, 16
- frameaddr
 - page_entry, 28
- free
 - mcb.c, 75
- free_pcb
 - pcb.h, 69
- fs
 - context, 10
- GDT_CS_ID
 - system.h, 46
- GDT_DS_ID
 - system.h, 46
- GETDATE
 - cmd_orders.h, 48
- GETTIME
 - cmd_orders.h, 48
- gdt_descriptor, 16
 - base, 16
 - limit, 16
- gdt_entry, 17
 - access, 17
 - base_high, 17
 - base_low, 17
 - base_mid, 17
 - flags, 17
 - limit_low, 17
- gdt_init_entry
 - tables.h, 38
- get_bit
 - paging.h, 40
- get_data_ptr
 - disk_img_manager.h, 90
- get_date
 - sys_clock.h, 61
- get_date_main
 - sys_clock.h, 61
- get_entry
 - disk_folder_manager.h, 84
- get_entry_by_name
 - disk_folder_manager.h, 84
- get_fat_date
 - disk_img_manager.h, 90
- get_fat_date_str
 - disk_img_manager.h, 90
- get_fat_time
 - disk_img_manager.h, 91
- get_fat_time_str
 - disk_img_manager.h, 91
- get_fat_val
 - disk_img_manager.h, 91
- get_input_line
 - serial.h, 36
- get_op_code
 - mpx_supt.h, 54
- get_page
 - paging.h, 40
- get_running_process
 - pcb.h, 69
- get_stack_base
 - pcb.h, 69
- get_stack_top
 - pcb.h, 69
- get_time
 - sys_clock.h, 61
- get_time_main
 - sys_clock.h, 61
- gs
 - context, 10
- HELP
 - cmd_orders.h, 48
- head
 - footer, 16
 - pcb_queue, 30
- head_num
 - img_boot_sector, 21
- header, 17
 - index_id, 18
 - size, 18
- heap, 18
 - base, 18
 - index, 18
 - max_size, 18
 - min_size, 19
- heap.h
 - _kmalloc, 39
 - alloc, 39
 - init_kheap, 39
 - KHEAP_BASE, 39
 - KHEAP_MIN, 39
 - KHEAP_SIZE, 39
 - kfree, 39
 - kmalloc, 39
 - make_heap, 39

TABLE_SIZE, 39
 help
 r1.h, 57
 help_usages
 r1.h, 58
 hlt
 system.h, 46
 hour
 date_time, 11
 hr
 fat_time, 15
 IDLE
 mpx_supt.h, 54
 IDLE_PCB_NAME
 pcb.h, 68
 id
 index_table, 24
 idle
 mpx_supt.h, 54
 idt_descriptor, 19
 base, 19
 limit, 19
 idt_entry, 19
 base_high, 19
 base_low, 19
 flags, 19
 sselect, 19
 zero, 20
 idt_set_gate
 tables.h, 38
 ignore1
 dir_entry_info, 13
 img_boot_sector, 21
 ignore2
 img_boot_sector, 21
 ignore3
 img_boot_sector, 22
 ignore4
 img_boot_sector, 22
 ignore5
 img_boot_sector, 22
 img_boot_sector, 20
 boot_sign, 21
 byte_per_sector, 21
 fat_copies_num, 21
 file_sys_type, 21
 head_num, 21
 ignore1, 21
 ignore2, 21
 ignore3, 22
 ignore4, 22
 ignore5, 22
 reserved_sec_num, 22
 root_dir_max_num, 22
 sec_num, 22
 sec_per_fat_num, 22
 sec_per_track, 22
 sector_per_cluster, 22
 total_sec_fat32, 23
 vol_id, 23
 vol_label, 23
 img_writer, 23
 import_file
 disk_file_manager.h, 81
 inb
 io.h, 34
 include/core/asm.h, 33
 include/core/interrupts.h, 33
 include/core/io.h, 34
 include/core/serial.h, 34
 include/core/tables.h, 37
 include/mem/heap.h, 38
 include/mem/paging.h, 39
 include/string.h, 40
 include/system.h, 45
 index
 heap, 18
 index_entry, 23
 block, 23
 empty, 23
 size, 24
 index_id
 header, 18
 index_table, 24
 id, 24
 table, 24
 init_dir_itr
 file_dir_iterator.h, 96
 init_file_itr
 file_dir_iterator.h, 96
 init_gdt
 tables.h, 38
 init_heap
 mcb.h, 78
 init_idt
 tables.h, 38
 init_img_writer
 file_dir_iterator.h, 96
 init_irq
 interrupts.h, 34
 init_kheap
 heap.h, 39
 init_paging
 paging.h, 40
 init_pic
 interrupts.h, 34
 init_serial

- serial.h, 36
- insert_pcb
 - pcb.h, 69
- interrupts.h
 - init_irq, 34
 - init_pic, 34
- io.h
 - inb, 34
 - outb, 34
- iret
 - system.h, 46
- irq_on
 - system.h, 46
- is_mcb_empty
 - mcb.h, 78
- is_suspended
 - pcb_struct, 31
- isspace
 - string.h, 44
- iw_write
 - file_dir_iterator.h, 96
- KHEAP_BASE
 - heap.h, 39
- KHEAP_MIN
 - heap.h, 39
- KHEAP_SIZE
 - heap.h, 39
- kfree
 - heap.h, 39
- klogv
 - system.h, 46
- kmalloc
 - heap.h, 39
- kpanic
 - system.h, 46
- LOADR3
 - cmd_orders.h, 48
- last_acc_date
 - dir_entry_info, 13
- last_wri_date
 - dir_entry_info, 13
- last_wri_time
 - dir_entry_info, 13
- limit
 - gdt_descriptor, 16
 - idt_descriptor, 19
- limit_low
 - gdt_entry, 17
- limited_mcb
 - mcb, 26
- list_dir_entry_report
 - disk_folder_manager.h, 85
- list_dir_entry_short
 - disk_folder_manager.h, 85
- list_file_report
 - disk_folder_manager.h, 85
- list_files_entry_ext
 - disk_folder_manager.h, 85
- list_files_entry_name
 - disk_folder_manager.h, 85
- lmcb, 24
 - size, 25
 - type, 25
- load_image_file
 - disk_img_manager.h, 91
- load_page_dir
 - paging.h, 40
- load_process
 - context.h, 74
- load_r3_main
 - context.h, 74
- MAX_HEAP_SIZE
 - mcb.h, 78
- MCB_FUNC_END
 - cmd_orders.h, 48
- MODULE_R1
 - mpx_supt.h, 54
- MODULE_R2
 - mpx_supt.h, 54
- MODULE_R3
 - mpx_supt.h, 54
- MODULE_R4
 - mpx_supt.h, 54
- MODULE_R5
 - mpx_supt.h, 54
- MPX_FUNC_END
 - cmd_orders.h, 48
- make_heap
 - heap.h, 39
- max_size
 - heap, 18
- mcb, 25
 - complete_mcb, 26
 - limited_mcb, 26
 - next, 26
 - prev, 26
 - r1.h, 57
- mcb.c
 - allocated, 75
 - free, 75
- mcb.c
 - mcb_type, 75
- mcb.h
 - init_heap, 78
 - is_mcb_empty, 78
 - MAX_HEAP_SIZE, 78

- mcb_allocate, 78
- mcb_allocate_mpx, 78
- mcb_allocate_mpx2, 78
- mcb_free_mpx, 78
- show_all_mcb, 78
- show_allocated_mcb, 78
- show_free_mcb, 78
- show_mcb, 78
- show_mcb_main, 78
- shutdown_mcb, 78
- start_of_memory, 78
- mcb_allocate
 - mcb.h, 78
- mcb_allocate_mpx
 - mcb.h, 78
- mcb_allocate_mpx2
 - mcb.h, 78
- mcb_free_mpx
 - mcb.h, 78
- mcb_type
 - mcb.c, 75
- memset
 - string.h, 44
- mi
 - fat_time, 15
- min
 - date_time, 11
- min_size
 - heap, 19
- modules/cmd_orders.h, 46
- modules/errno.h, 49
- modules/mpx_supt.h, 51
- modules/packing.h, 55
- modules/r1/r1.h, 56
- modules/r1/sys_clock.h, 58
- modules/r2/pcb.c, 61
- modules/r2/pcb.h, 62
- modules/r2/pcb_comm.h, 69
- modules/r3/context.h, 72
- modules/r5/mcb.c, 74
- modules/r5/mcb.h, 75
- modules/r6/ansi.h, 79
- modules/r6/disk_file_manager.h, 79
- modules/r6/disk_folder_manager.h, 81
- modules/r6/disk_img_manager.h, 85
- modules/r6/file_dir_iterator.h, 91
- mon
 - date_time, 11
 - fat_date, 14
- move_file
 - disk_file_manager.h, 81
- mpx
 - r1.h, 57
- mpx_init
 - mpx_supt.h, 54
- mpx_supt.h
 - EXIT, 54
 - get_op_code, 54
 - IDLE, 54
 - idle, 54
 - MODULE_R1, 54
 - MODULE_R2, 54
 - MODULE_R3, 54
 - MODULE_R4, 54
 - MODULE_R5, 54
 - mpx_init, 54
 - READ, 54
 - sys_alloc_mem, 54
 - sys_free_mem, 54
 - sys_req, 55
 - sys_set_free, 55
 - sys_set_malloc, 55
 - WRITE, 54
- NULL
 - system.h, 46
- NUM_OF_FUNCTIONS
 - cmd_orders.h, 48
- name
 - pcb_struct, 32
- new_frame
 - paging.h, 40
- next
 - mcb, 26
 - pcb_struct, 32
- no_warn
 - system.h, 46
- nop
 - system.h, 46
- old_context
 - context.h, 74
- op_code
 - param, 29
- outb
 - io.h, 34
- PACKED
 - packing.h, 56
- PAGE_SIZE
 - paging.h, 40
- PCB_FUNC_END
 - cmd_orders.h, 48
- packing.h
 - PACKED, 56
- page_dir, 26
 - tables, 27
 - tables_phys, 27
- page_entry, 27

- accessed, 28
- dirty, 28
- frameaddr, 28
- present, 28
- reserved, 28
- usermode, 28
- writable, 28
- page_table, 28
- pages, 29
- pages
 - page_table, 29
- paging.h
 - clear_bit, 40
 - first_free, 40
 - get_bit, 40
 - get_page, 40
 - init_paging, 40
 - load_page_dir, 40
 - new_frame, 40
 - PAGE_SIZE, 40
 - set_bit, 40
- param, 29
 - device_id, 29
 - op_code, 29
- pcb
 - r1.h, 57
- pcb.h
 - pcb_class_app, 68
 - pcb_class_sys, 68
- pcb.c
 - blocked_queue, 62
 - ready_queue, 62
- pcb.h
 - allocate_pcb, 69
 - block_pcb, 69
 - COMMHAND_PCB_NAME, 68
 - find_pcb, 69
 - free_pcb, 69
 - get_running_process, 69
 - get_stack_base, 69
 - get_stack_top, 69
 - IDLE_PCB_NAME, 68
 - insert_pcb, 69
 - pcb_init, 69
 - process_class, 68
 - remove_pcb, 69
 - resume_pcb, 69
 - SIZE_OF_PCB_NAME, 68
 - SIZE_OF_STACK, 68
 - save_running_process, 69
 - set_pcb_priority, 69
 - setup_pcb, 69
 - show_all_processes, 69
 - show_blocked_processes, 69
 - show_pcb, 69
 - show_ready_processes, 69
 - shutdown_pcb, 69
 - suspend_pcb, 69
 - unblock_pcb, 69
- pcb_class_app
 - pcb.h, 68
- pcb_class_sys
 - pcb.h, 68
- pcb_comm.h
 - resume_pcb_main, 71
 - set_pcb_priority_main, 71
 - show_pcb_main, 71
 - suspend_pcb_main, 71
- pcb_init
 - pcb.h, 69
- pcb_queue, 29
 - count, 30
 - head, 30
 - tail, 30
- pcb_struct, 31
 - class, 31
 - is_suspended, 31
 - name, 32
 - next, 32
 - prev, 32
 - priority, 32
 - running_state, 32
 - stack_base, 32
 - stack_top, 32
- pop_folder
 - disk_folder_manager.h, 85
- present
 - page_entry, 28
- prev
 - mcb, 26
 - pcb_struct, 32
- print_boot_sec_info
 - disk_img_manager.h, 91
- print_curr_path
 - disk_folder_manager.h, 85
- print_dir_entry_info
 - disk_folder_manager.h, 85
- print_help
 - r1.h, 58
- print_report_heading
 - disk_folder_manager.h, 85
- printf
 - string.h, 44
- priority
 - pcb_struct, 32
- process_class
 - pcb.h, 68
- push_folder

- disk_folder_manager.h, 85
- r1.h
 - help, 57
 - mcb, 57
 - mpx, 57
 - pcb, 57
- r1.h
 - comm_type, 57
 - command_line_parser, 58
 - commhand, 58
 - help_usages, 58
 - print_help, 58
- READ
 - mpx_supt.h, 54
- RESUMEPCB
 - cmd_orders.h, 48
- ready_queue
 - pcb.c, 62
- remove_pcb
 - pcb.h, 69
- rename_entry
 - disk_folder_manager.h, 85
- reserved
 - dir_entry_info, 13
 - page_entry, 28
- reserved_sec_num
 - img_boot_sector, 22
- resume_pcb
 - pcb.h, 69
- resume_pcb_main
 - pcb_comm.h, 71
- root_dir_entry
 - disk_img_manager.h, 91
- root_dir_file_arr
 - disk_img_manager.h, 91
- root_dir_max_num
 - img_boot_sector, 22
- running_state
 - pcb_struct, 32
- SETDATE
 - cmd_orders.h, 48
- SETPCBPRIO
 - cmd_orders.h, 48
- SETTIME
 - cmd_orders.h, 48
- SHOWMCB
 - cmd_orders.h, 48
- SHOWPCB
 - cmd_orders.h, 48
- SHUTDOWN
 - cmd_orders.h, 49
- SIZE_OF_PCB_NAME
 - pcb.h, 68
- SIZE_OF_STACK
 - pcb.h, 68
- SUSPDPCB
 - cmd_orders.h, 49
- save_running_process
 - pcb.h, 69
- se
 - fat_time, 15
- sec
 - date_time, 11
- sec_num
 - img_boot_sector, 22
- sec_per_fat_num
 - img_boot_sector, 22
- sec_per_track
 - img_boot_sector, 22
- sector_per_cluster
 - img_boot_sector, 22
- seperate_file_name
 - disk_img_manager.h, 91
- serial.h
 - COM1, 36
 - COM2, 36
 - COM3, 36
 - COM4, 36
 - get_input_line, 36
 - init_serial, 36
 - serial_print, 36
 - serial_println, 36
 - set_serial_in, 37
 - set_serial_out, 37
 - WithEcho, 36
 - WithoutEcho, 36
- serial_print
 - serial.h, 36
- serial_println
 - serial.h, 36
- set_bit
 - paging.h, 40
- set_date
 - sys_clock.h, 61
- set_date_main
 - sys_clock.h, 61
- set_date_str
 - sys_clock.h, 61
- set_fat_time
 - disk_img_manager.h, 91
- set_pcb_priority
 - pcb.h, 69
- set_pcb_priority_main
 - pcb_comm.h, 71
- set_serial_in
 - serial.h, 37
- set_serial_out

- serial.h, 37
- set_time
 - sys_clock.h, 61
- set_time_main
 - sys_clock.h, 61
- set_time_str
 - sys_clock.h, 61
- setup_pcb
 - pcb.h, 69
- show_all_mcb
 - mcb.h, 78
- show_all_processes
 - pcb.h, 69
- show_allocated_mcb
 - mcb.h, 78
- show_blocked_processes
 - pcb.h, 69
- show_free_mcb
 - mcb.h, 78
- show_mcb
 - mcb.h, 78
- show_mcb_main
 - mcb.h, 78
- show_pcb
 - pcb.h, 69
- show_pcb_main
 - pcb_comm.h, 71
- show_ready_processes
 - pcb.h, 69
- shutdown_mcb
 - mcb.h, 78
- shutdown_pcb
 - pcb.h, 69
- size
 - cmcb, 7
 - header, 18
 - index_entry, 24
 - lmcb, 25
- size_t
 - system.h, 46
- sprintf
 - string.h, 44
- sselect
 - idt_entry, 19
- stack_base
 - pcb_struct, 32
- stack_top
 - pcb_struct, 32
- start_of_memory
 - mcb.h, 78
- sti
 - system.h, 46
- str_to_ch_arr
 - disk_img_manager.h, 91
- str_to_upper_case
 - disk_img_manager.h, 91
- strcat
 - string.h, 45
- strcmp
 - string.h, 45
- strcpy
 - string.h, 45
- string.h
 - atoi, 44
 - isspace, 44
 - memset, 44
 - printf, 44
 - sprintf, 44
 - strcat, 45
 - strcmp, 45
 - strcpy, 45
 - strlen, 45
 - strtok, 45
- strlen
 - string.h, 45
- strtok
 - string.h, 45
- suspend_pcb
 - pcb.h, 69
- suspend_pcb_main
 - pcb_comm.h, 71
- sys_alloc_mem
 - mpx_supt.h, 54
- sys_call
 - context.h, 74
- sys_clock.h
 - get_date, 61
 - get_date_main, 61
 - get_time, 61
 - get_time_main, 61
 - set_date, 61
 - set_date_main, 61
 - set_date_str, 61
 - set_time, 61
 - set_time_main, 61
 - set_time_str, 61
- sys_free_mem
 - mpx_supt.h, 54
- sys_req
 - mpx_supt.h, 55
- sys_set_free
 - mpx_supt.h, 55
- sys_set_malloc
 - mpx_supt.h, 55
- system.h
 - asm, 46
 - cli, 46
 - GDT_CS_ID, 46

- GDT_DS_ID, 46
- hlt, 46
- iret, 46
- irq_on, 46
- klogv, 46
- kpanic, 46
- NULL, 46
- no_warn, 46
- nop, 46
- size_t, 46
- sti, 46
- u16int, 46
- u32int, 46
- u8int, 46
- volatile, 46
- T_BOLD
 - ansi.h, 79
- T_BOLD_OFF
 - ansi.h, 79
- T_CYAN
 - ansi.h, 79
- T_DIR
 - ansi.h, 79
- T_DIR_OFF
 - ansi.h, 79
- T_ITCS
 - ansi.h, 79
- T_ITCS_OFF
 - ansi.h, 79
- T_NRM
 - ansi.h, 79
- T_RED
 - ansi.h, 79
- T_RESET
 - ansi.h, 79
- T_WHT
 - ansi.h, 79
- TABLE_SIZE
 - heap.h, 39
- table
 - index_table, 24
- tables
 - page_dir, 27
- tables.h
 - gdt_init_entry, 38
 - idt_set_gate, 38
 - init_gdt, 38
 - init_idt, 38
- tables_phys
 - page_dir, 27
- tail
 - pcb_queue, 30
- total_sec_fat32
 - img_boot_sector, 23
- type
 - cmcb, 7
 - lmcb, 25
- type_file
 - disk_file_manager.h, 81
- u16int
 - system.h, 46
- u32int
 - system.h, 46
- u8int
 - system.h, 46
- unblock_pcb
 - pcb.h, 69
- usermode
 - page_entry, 28
- VERSION
 - cmd_orders.h, 49
- vol_id
 - img_boot_sector, 23
- vol_label
 - img_boot_sector, 23
- volatile
 - system.h, 46
- WITH_R2_TEMP_CMD
 - cmd_orders.h, 49
- WITH_R3_TEMP_CMD
 - cmd_orders.h, 49
- WITH_R5_TEMP_CMD
 - cmd_orders.h, 49
- WRITE
 - mpx_supt.h, 54
- WithEcho
 - serial.h, 36
- WithoutEcho
 - serial.h, 36
- write_fat
 - disk_img_manager.h, 91
- write_image_file
 - disk_img_manager.h, 91
- writable
 - page_entry, 28
- year
 - date_time, 11
 - fat_date, 14
- zero
 - idt_entry, 20