

mpx-spring2017-npennf
NPE-NPX.R6.04282017

Generated by Doxygen 1.8.11

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	boot_sector Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	6
3.1.2.1	bootSig	6
3.1.2.2	bytesPerSector	6
3.1.2.3	fileSystemType	6
3.1.2.4	ignore_1	6
3.1.2.5	ignore_2	6
3.1.2.6	ignore_3	6
3.1.2.7	ignore_4	6
3.1.2.8	ignore_5	6
3.1.2.9	maxRootDirEntries	6
3.1.2.10	numFATCopies	6
3.1.2.11	numHeads	6
3.1.2.12	numReservedSectors	6
3.1.2.13	numSectors	6
3.1.2.14	sectorCountFAT32	6

3.1.2.15	sectorsPerCluster	6
3.1.2.16	sectorsPerFAT	6
3.1.2.17	sectorsPerTrack	6
3.1.2.18	volId	6
3.1.2.19	volName	6
3.2	cmcb Struct Reference	7
3.2.1	Field Documentation	7
3.2.1.1	beginningAddr	7
3.2.1.2	memSize	7
3.2.1.3	name	7
3.2.1.4	next	7
3.2.1.5	prev	7
3.2.1.6	size	7
3.2.1.7	type	7
3.3	context Struct Reference	8
3.3.1	Field Documentation	8
3.3.1.1	cs	8
3.3.1.2	ds	8
3.3.1.3	eax	8
3.3.1.4	ebp	8
3.3.1.5	ebx	8
3.3.1.6	ecx	8
3.3.1.7	edi	8
3.3.1.8	edx	8
3.3.1.9	eflags	8
3.3.1.10	eip	8
3.3.1.11	es	8
3.3.1.12	esi	8
3.3.1.13	esp	8
3.3.1.14	fs	8

3.3.1.15	gs	8
3.4	date_time Struct Reference	9
3.4.1	Detailed Description	9
3.4.2	Field Documentation	9
3.4.2.1	day_m	9
3.4.2.2	day_w	9
3.4.2.3	day_y	9
3.4.2.4	hour	9
3.4.2.5	min	9
3.4.2.6	mon	9
3.4.2.7	sec	9
3.4.2.8	year	9
3.5	dir_entry Struct Reference	9
3.5.1	Detailed Description	10
3.5.2	Field Documentation	10
3.5.2.1	attributes	10
3.5.2.2	creationDate	10
3.5.2.3	creationTime	10
3.5.2.4	extension	10
3.5.2.5	filename	10
3.5.2.6	fileSize	10
3.5.2.7	firstLogicalCluster	10
3.5.2.8	ignore	10
3.5.2.9	lastAccess	10
3.5.2.10	lastWriteDate	10
3.5.2.11	lastWriteTime	10
3.5.2.12	reserved	10
3.6	fat_tables Struct Reference	11
3.6.1	Detailed Description	11
3.6.2	Field Documentation	11

3.6.2.1	fat1	11
3.6.2.2	fat2	11
3.6.2.3	numEntries	11
3.7	footer Struct Reference	11
3.7.1	Detailed Description	12
3.7.2	Field Documentation	12
3.7.2.1	head	12
3.8	functionDef Struct Reference	12
3.8.1	Field Documentation	12
3.8.1.1	funcPointer	12
3.8.1.2	helpString	12
3.8.1.3	name	12
3.9	gdt_descriptor_struct Struct Reference	12
3.9.1	Field Documentation	13
3.9.1.1	base	13
3.9.1.2	limit	13
3.10	gdt_entry_struct Struct Reference	13
3.10.1	Field Documentation	13
3.10.1.1	access	13
3.10.1.2	base_high	13
3.10.1.3	base_low	13
3.10.1.4	base_mid	13
3.10.1.5	flags	13
3.10.1.6	limit_low	13
3.11	header Struct Reference	14
3.11.1	Detailed Description	14
3.11.2	Field Documentation	14
3.11.2.1	index_id	14
3.11.2.2	size	14
3.12	heap Struct Reference	14

3.12.1 Detailed Description	15
3.12.2 Field Documentation	15
3.12.2.1 base	15
3.12.2.2 index	15
3.12.2.3 max_size	15
3.12.2.4 min_size	15
3.13 idt_entry_struct Struct Reference	15
3.13.1 Field Documentation	15
3.13.1.1 base_high	15
3.13.1.2 base_low	15
3.13.1.3 flags	15
3.13.1.4 sselect	15
3.13.1.5 zero	15
3.14 idt_struct Struct Reference	16
3.14.1 Field Documentation	16
3.14.1.1 base	16
3.14.1.2 limit	16
3.15 index_entry Struct Reference	16
3.15.1 Field Documentation	16
3.15.1.1 block	16
3.15.1.2 empty	16
3.15.1.3 size	16
3.16 index_table Struct Reference	17
3.16.1 Detailed Description	17
3.16.2 Field Documentation	17
3.16.2.1 id	17
3.16.2.2 table	17
3.17 Imcb Struct Reference	17
3.17.1 Field Documentation	18
3.17.1.1 memSize	18

3.17.1.2	size	18
3.17.1.3	type	18
3.18	node Struct Reference	18
3.18.1	Detailed Description	18
3.18.2	Field Documentation	19
3.18.2.1	data	19
3.18.2.2	next	19
3.18.2.3	prev	19
3.19	page_dir Struct Reference	19
3.19.1	Detailed Description	19
3.19.2	Field Documentation	20
3.19.2.1	tables	20
3.19.2.2	tables_phys	20
3.20	page_entry Struct Reference	20
3.20.1	Detailed Description	20
3.20.2	Field Documentation	20
3.20.2.1	accessed	20
3.20.2.2	dirty	20
3.20.2.3	frameaddr	20
3.20.2.4	present	20
3.20.2.5	reserved	20
3.20.2.6	usermode	20
3.20.2.7	writable	20
3.21	page_table Struct Reference	21
3.21.1	Detailed Description	21
3.21.2	Field Documentation	21
3.21.2.1	pages	21
3.22	param Struct Reference	21
3.22.1	Field Documentation	22
3.22.1.1	device_id	22
3.22.1.2	op_code	22
3.23	pcb Struct Reference	22
3.23.1	Field Documentation	22
3.23.1.1	isSuspended	22
3.23.1.2	priority	22
3.23.1.3	processClass	22
3.23.1.4	processName	22
3.23.1.5	stackBottom	22
3.23.1.6	stackTop	22
3.23.1.7	state	22

4 File Documentation	23
4.1 include/boolean.h File Reference	23
4.1.1 Enumeration Type Documentation	23
4.1.1.1 boolean	23
4.2 include/core/asm.h File Reference	24
4.3 include/core/comHandler.h File Reference	24
4.3.1 Function Documentation	25
4.3.1.1 addComHistory(char *newCom)	25
4.3.1.2 addFunctionDef(char *name, const char *helpString, const char *(func↵ Pointer)(char **args, int numArgs))	25
4.3.1.3 eraseCurrentRow(int endIndex, int insertionIndex)	26
4.3.1.4 executeCommand(char *commandString)	26
4.3.1.5 getComHistory(int isPrev)	26
4.3.1.6 getFunctionDef(char *name)	26
4.3.1.7 getHelpString(char *name)	27
4.3.1.8 getInput()	27
4.3.1.9 initCommandHandler()	27
4.3.1.10 printStart()	27
4.3.1.11 returnToInsertionPoint(int endIndex, int insertionIndex)	27
4.3.1.12 setupCommands()	28
4.4 include/core/commands.h File Reference	28
4.4.1 Function Documentation	28
4.4.1.1 date(char **args, int numArgs)	28
4.4.1.2 help(char **args, int numArgs)	29
4.4.1.3 shutdown(char **args, int numArgs)	29
4.4.1.4 version(char **args, int numArgs)	30
4.5 include/core/help.h File Reference	30
4.5.1 Macro Definition Documentation	31
4.5.1.1 HELP_COMMAND_DATE	31
4.5.1.2 HELP_COMMAND_HELP	31
4.5.1.3 HELP_COMMAND_SHUTDOWN	31

4.5.1.4	HELP_COMMAND_VERSION	32
4.5.1.5	HELP_INVALID_ARGUMENTS	32
4.5.1.6	HELP_UNKNOWN_COMMAND	32
4.6	include/modules/R2/commands/help.h File Reference	32
4.6.1	Macro Definition Documentation	33
4.6.1.1	HELP_R2_COMMAND_PPCB	33
4.6.1.2	HELP_R2_COMMAND_RPCB	33
4.6.1.3	HELP_R2_COMMAND_SHOWPCB	33
4.6.1.4	HELP_R2_COMMAND_SPCB	33
4.7	include/modules/R3/commands/help.h File Reference	34
4.7.1	Macro Definition Documentation	34
4.7.1.1	HELP_R3_COMMAND_LOAD	34
4.7.1.2	HELP_R3_COMMAND_YIELD	34
4.8	include/modules/R5/commands/help.h File Reference	35
4.8.1	Macro Definition Documentation	35
4.8.1.1	HELP_R5_COMMAND_SHOWMEMORY	35
4.9	include/modules/R5/help.h File Reference	36
4.9.1	Macro Definition Documentation	36
4.9.1.1	HELP_R5_COMMAND_ALLOC	36
4.9.1.2	HELP_R5_COMMAND_EMPTY	36
4.9.1.3	HELP_R5_COMMAND_FREE	37
4.9.1.4	HELP_R5_COMMAND_HEAP	37
4.10	include/core/interrupts.h File Reference	37
4.10.1	Function Documentation	37
4.10.1.1	init_irq(void)	37
4.10.1.2	init_pic(void)	38
4.11	include/core/io.h File Reference	38
4.11.1	Macro Definition Documentation	38
4.11.1.1	inb	38
4.11.1.2	outb	38

4.12	include/core/pcb.h File Reference	39
4.12.1	Macro Definition Documentation	40
4.12.1.1	APPLICATION	40
4.12.1.2	BLOCKED	40
4.12.1.3	READY	40
4.12.1.4	RUNNING	40
4.12.1.5	SYSTEM	40
4.12.2	Typedef Documentation	40
4.12.2.1	pcb	40
4.12.3	Function Documentation	40
4.12.3.1	allocatePCB()	40
4.12.3.2	checkParamClass(int processClass)	40
4.12.3.3	checkParamName(const char *processName)	41
4.12.3.4	checkParamPriority(int priority)	41
4.12.3.5	freePCB(pcb *pcbPtr)	41
4.12.3.6	setupPCB(const char *processName, int processClass, int priority)	42
4.13	include/core/queue.h File Reference	42
4.13.1	Typedef Documentation	43
4.13.1.1	node	43
4.13.2	Function Documentation	43
4.13.2.1	findPCB(const char *processName)	43
4.13.2.2	getBlockedQueue()	44
4.13.2.3	getReadyQueue()	44
4.13.2.4	getSuspendedBlockedQueue()	44
4.13.2.5	getSuspendedReadyQueue()	44
4.13.2.6	insertPCB(pcb *p)	44
4.13.2.7	popBlocked()	45
4.13.2.8	popReady()	45
4.13.2.9	popSuspendedBlocked()	45
4.13.2.10	popSuspendedReady()	45

4.13.2.11 removePCB(pcb *p)	45
4.14 include/core/serial.h File Reference	46
4.14.1 Macro Definition Documentation	46
4.14.1.1 COM1	46
4.14.1.2 COM2	46
4.14.1.3 COM3	46
4.14.1.4 COM4	46
4.14.2 Function Documentation	46
4.14.2.1 init_serial(int device)	46
4.14.2.2 serial_print(const char *msg)	47
4.14.2.3 serial_println(const char *msg)	47
4.14.2.4 set_serial_in(int device)	47
4.14.2.5 set_serial_out(int device)	48
4.15 include/core/tables.h File Reference	48
4.15.1 Function Documentation	49
4.15.1.1 __attribute__((packed)) idt_entry	49
4.15.1.2 gdt_init_entry(int idx, u32int base, u32int limit, u8int access, u8int flags)	49
4.15.1.3 idt_set_gate(u8int idx, u32int base, u16int sel, u8int flags)	49
4.15.1.4 init_gdt()	50
4.15.1.5 init_idt()	50
4.15.2 Variable Documentation	50
4.15.2.1 access	50
4.15.2.2 base	50
4.15.2.3 base_high	50
4.15.2.4 base_low	50
4.15.2.5 base_mid	50
4.15.2.6 flags	50
4.15.2.7 limit	50
4.15.2.8 limit_low	50
4.15.2.9 sselect	50

4.15.2.10 zero	50
4.16 include/core/version.h File Reference	50
4.16.1 Macro Definition Documentation	51
4.16.1.1 OS_VERSION	51
4.17 include/math.h File Reference	51
4.17.1 Function Documentation	51
4.17.1.1 bcdToDec(unsigned char bcd)	51
4.17.1.2 decToBcd(unsigned char dec)	52
4.18 include/mem/heap.h File Reference	52
4.18.1 Macro Definition Documentation	53
4.18.1.1 KHEAP_BASE	53
4.18.1.2 KHEAP_MIN	53
4.18.1.3 KHEAP_SIZE	53
4.18.1.4 TABLE_SIZE	53
4.18.2 Function Documentation	53
4.18.2.1 _kmalloc(u32int size, int align, u32int *phys_addr)	53
4.18.2.2 alloc(u32int size, heap *hp, int align)	53
4.18.2.3 init_kheap()	54
4.18.2.4 kfree()	54
4.18.2.5 kmalloc(u32int size)	54
4.18.2.6 make_heap(u32int base, u32int max, u32int min)	54
4.18.3 Variable Documentation	54
4.18.3.1 __attribute__	55
4.19 include/mem/memoryControl.h File Reference	55
4.19.1 Macro Definition Documentation	56
4.19.1.1 ALLOCATED	56
4.19.1.2 FREE	56
4.19.2 Typedef Documentation	56
4.19.2.1 cmcb	56
4.19.2.2 lmcb	56

4.19.3	Function Documentation	56
4.19.3.1	allocateMemory(int size)	56
4.19.3.2	deallocateMemory(void *memPointer)	57
4.19.3.3	getAllocatedHead()	57
4.19.3.4	getFreeHead()	57
4.19.3.5	initializeHeap(int size)	57
4.19.3.6	isEmpty()	58
4.20	include/mem/paging.h File Reference	58
4.20.1	Macro Definition Documentation	59
4.20.1.1	PAGE_SIZE	59
4.20.2	Function Documentation	59
4.20.2.1	clear_bit(u32int addr)	59
4.20.2.2	first_free()	59
4.20.2.3	get_bit(u32int addr)	59
4.20.2.4	get_page(u32int addr, page_dir *dir, int make_table)	60
4.20.2.5	init_paging()	60
4.20.2.6	load_page_dir(page_dir *new_page_dir)	60
4.20.2.7	new_frame(page_entry *page)	61
4.20.2.8	set_bit(u32int addr)	61
4.21	include/modules/mpx_supt.h File Reference	61
4.21.1	Macro Definition Documentation	63
4.21.1.1	EXIT	63
4.21.1.2	IDLE	63
4.21.1.3	MODULE_R1	63
4.21.1.4	MODULE_R2	63
4.21.1.5	MODULE_R3	63
4.21.1.6	MODULE_R4	63
4.21.1.7	MODULE_R5	63
4.21.1.8	READ	63
4.21.1.9	WRITE	63

4.21.2	Typedef Documentation	63
4.21.2.1	context	63
4.21.3	Function Documentation	63
4.21.3.1	getCOPName()	63
4.21.3.2	idle()	64
4.21.3.3	memset(void *s, int c, size_t n)	64
4.21.3.4	mpx_init(int cur_mod)	65
4.21.3.5	sys_alloc_mem(u32int size)	65
4.21.3.6	sys_call(context *registers)	65
4.21.3.7	sys_free_mem(void *ptr)	65
4.21.3.8	sys_req(int op_code)	66
4.21.3.9	sys_set_free(boolean(func)(void *))	66
4.21.3.10	sys_set_malloc(void *(*func)(int))	66
4.22	include/modules/R2/commands/help_temp.h File Reference	67
4.22.1	Macro Definition Documentation	67
4.22.1.1	HELP_R2_COMMAND_BPCB	67
4.22.1.2	HELP_R2_COMMAND_CPCB	68
4.22.1.3	HELP_R2_COMMAND_DPCB	68
4.22.1.4	HELP_R2_COMMAND_UPCB	68
4.23	include/modules/R2/commands/perm.h File Reference	68
4.23.1	Function Documentation	69
4.23.1.1	registerR2PermCommands()	69
4.23.1.2	resumePcb(char **args, int numArgs)	69
4.23.1.3	setPriorityPcb(char **args, int numArgs)	70
4.23.1.4	showPcbInfo(char **args, int numArgs)	70
4.23.1.5	suspendPcb(char **args, int numArgs)	71
4.24	include/modules/R2/commands/status.h File Reference	72
4.24.1	Macro Definition Documentation	72
4.24.1.1	RESUME_PCB_SUCCESS	72
4.24.1.2	RESUME_PCBS_SUCCESS	72

4.24.1.3	SUSPEND_PCB_SUCCESS	72
4.24.1.4	UNKNOWN_PCB_NAME	72
4.24.1.5	UPDATE_PRIORITY_SUCCESS	72
4.25	include/modules/R2/commands/status_temp.h File Reference	73
4.25.1	Macro Definition Documentation	73
4.25.1.1	BLOCK_PCB_SUCCESS	73
4.25.1.2	CREATE_PCB_SUCCESS	73
4.25.1.3	DELETE_PCB_SUCCESS	73
4.25.1.4	PROCESS_NAME_ALREADY_EXISTS	73
4.25.1.5	UNBLOCK_PCB_SUCCESS	73
4.26	include/modules/R2/commands/temp.h File Reference	74
4.26.1	Function Documentation	75
4.26.1.1	blockPcb(char **args, int numArgs)	75
4.26.1.2	createPcb(char **args, int numArgs)	75
4.26.1.3	deletePcb(char **args, int numArgs)	75
4.26.1.4	registerR2TempCommands()	76
4.26.1.5	unblockPcb(char **args, int numArgs)	76
4.27	include/modules/R3/commands/r3commands.h File Reference	76
4.27.1	Function Documentation	77
4.27.1.1	loadr3(char **args, int numArgs)	77
4.27.1.2	registerR3Commands()	78
4.27.1.3	yield(char **args, int numArgs)	78
4.28	include/modules/R3/processes.h File Reference	78
4.28.1	Function Documentation	79
4.28.1.1	proc1()	79
4.28.1.2	proc2()	79
4.28.1.3	proc3()	79
4.28.1.4	proc4()	79
4.28.1.5	proc5()	79
4.29	include/modules/R5/commands/r5commands.h File Reference	79

4.29.1	Function Documentation	80
4.29.1.1	registerR5PermCommands()	80
4.29.1.2	showMemory(char **args, int numArgs)	80
4.30	include/modules/R5/memCommands.h File Reference	81
4.30.1	Function Documentation	82
4.30.1.1	allocateMem(char **args, int numArgs)	82
4.30.1.2	freeMemory(char **args, int numArgs)	82
4.30.1.3	initHeap(char **args, int numArgs)	82
4.30.1.4	isEmptyCom(char **args, int numArgs)	82
4.30.1.5	registerR5TempCommands()	83
4.31	include/regex.h File Reference	83
4.31.1	Function Documentation	83
4.31.1.1	testRegex(const char *regex, const char *stringToCheck)	83
4.32	include/string.h File Reference	84
4.32.1	Function Documentation	84
4.32.1.1	atoi(const char *s)	84
4.32.1.2	isChar(const char c)	85
4.32.1.3	isdigit(const char c)	85
4.32.1.4	isLowerChar(const char c)	85
4.32.1.5	isspace(const char *c)	86
4.32.1.6	isUpperChar(const char c)	86
4.32.1.7	itoa(int num, char *str, int base)	86
4.32.1.8	reverse(char *str, int len)	87
4.32.1.9	strcat(char *s1, const char *s2)	87
4.32.1.10	strcmp(const char *s1, const char *s2)	87
4.32.1.11	strcpy(char *s1, const char *s2)	87
4.32.1.12	strlen(const char *s)	88
4.32.1.13	strtok(char *s1, const char *s2)	88
4.33	include/system.h File Reference	88
4.33.1	Macro Definition Documentation	89

4.33.1.1	asm	89
4.33.1.2	cli	89
4.33.1.3	GDT_CS_ID	89
4.33.1.4	GDT_DS_ID	89
4.33.1.5	hlt	89
4.33.1.6	iret	89
4.33.1.7	no_warn	89
4.33.1.8	nop	90
4.33.1.9	NULL	90
4.33.1.10	sti	90
4.33.1.11	volatile	90
4.33.2	Typedef Documentation	90
4.33.2.1	size_t	90
4.33.2.2	u16int	90
4.33.2.3	u32int	90
4.33.2.4	u8int	90
4.33.3	Function Documentation	90
4.33.3.1	klogv(const char *msg)	90
4.33.3.2	kpanic(const char *msg)	90
4.34	include/time.h File Reference	91
4.34.1	Macro Definition Documentation	92
4.34.1.1	APR	92
4.34.1.2	AUG	92
4.34.1.3	CONTROL_PORT	92
4.34.1.4	DATA_PORT	92
4.34.1.5	DAY_MONTH	92
4.34.1.6	DAY_WEEK	92
4.34.1.7	DEC	92
4.34.1.8	FEB	92
4.34.1.9	FRI	92

4.34.1.10 HOURS	92
4.34.1.11 JAN	92
4.34.1.12 JUL	93
4.34.1.13 JUN	93
4.34.1.14 MAR	93
4.34.1.15 MAY	93
4.34.1.16 MINUTES	93
4.34.1.17 MON	93
4.34.1.18 MONTH	93
4.34.1.19 NMI_DISABLE	93
4.34.1.20 NMI_ENABLE	93
4.34.1.21 NOV	93
4.34.1.22 OCT	93
4.34.1.23 SAT	93
4.34.1.24 SECONDS	93
4.34.1.25 SEP	93
4.34.1.26 SUN	93
4.34.1.27 THU	93
4.34.1.28 TIME_DELIM	93
4.34.1.29 TUE	94
4.34.1.30 WED	94
4.34.1.31 YEAR	94
4.34.2 Function Documentation	94
4.34.2.1 getDateTime()	94
4.34.2.2 getDayOfMonth()	94
4.34.2.3 getDayOfWeek()	94
4.34.2.4 getHours()	94
4.34.2.5 getMinutes()	95
4.34.2.6 getMonth()	95
4.34.2.7 getSeconds()	95

4.34.2.8	getYear()	95
4.34.2.9	isLeapYear(int year)	95
4.34.2.10	setDateTime(date_time)	96
4.34.2.11	setDayOfMonth(unsigned char dayOfMonth)	96
4.34.2.12	setDayOfWeek(unsigned char dayOfWeek)	96
4.34.2.13	setHours(unsigned char hours)	96
4.34.2.14	setMinutes(unsigned char minutes)	96
4.34.2.15	setMonth(unsigned char month)	97
4.34.2.16	setSeconds(unsigned char seconds)	97
4.34.2.17	setYear(unsigned char year)	97
4.34.2.18	updateDayOfWeek(date_time *dateTime)	97
4.34.2.19	updateDayOfYear(date_time *dateTime)	97
4.34.3	Variable Documentation	98
4.34.3.1	DAYS_IN_MONTH	98
4.35	kernel/core/comHandler.c File Reference	98
4.35.1	Function Documentation	99
4.35.1.1	addComHistory(char *newCom)	99
4.35.1.2	addFunctionDef(char *name, const char *helpString, const char *(func↵ Pointer)(char **args, int numArgs))	99
4.35.1.3	eraseCurrentRow(int endIndex, int insertionIndex)	99
4.35.1.4	executeCommand(char *commandString)	100
4.35.1.5	getComHistory(int isPrev)	100
4.35.1.6	getFunctionDef(char *name)	100
4.35.1.7	getHelpString(char *name)	100
4.35.1.8	getInput()	101
4.35.1.9	help(char **args, int numArgs)	101
4.35.1.10	initCommandHandler()	101
4.35.1.11	printStart()	101
4.35.1.12	returnToInsertionPoint(int endIndex, int insertionIndex)	101
4.35.1.13	setupCommands()	102
4.35.1.14	shutdown(char **args, int numArgs)	102

4.35.2	Variable Documentation	102
4.35.2.1	buffer	102
4.35.2.2	comHistory	102
4.35.2.3	comHistoryPos	102
4.35.2.4	continueHandle	102
4.35.2.5	functionDefs	102
4.35.2.6	functionInsertPoint	102
4.36	kernel/core/commands.c File Reference	103
4.36.1	Function Documentation	103
4.36.1.1	date(char **args, int numArgs)	103
4.36.1.2	version(char **args, int numArgs)	104
4.37	kernel/core/interrupts.c File Reference	104
4.37.1	Macro Definition Documentation	105
4.37.1.1	ICW1	105
4.37.1.2	ICW4	105
4.37.1.3	io_wait	105
4.37.1.4	PIC1	106
4.37.1.5	PIC2	106
4.37.2	Function Documentation	106
4.37.2.1	bounds()	106
4.37.2.2	breakpoint()	106
4.37.2.3	coprocessor()	106
4.37.2.4	coprocessor_segment()	106
4.37.2.5	debug()	106
4.37.2.6	device_not_available()	106
4.37.2.7	divide_error()	106
4.37.2.8	do_bounds()	106
4.37.2.9	do_breakpoint()	106
4.37.2.10	do_coprocessor()	106
4.37.2.11	do_coprocessor_segment()	106

4.37.2.12 do_debug()	106
4.37.2.13 do_device_not_available()	106
4.37.2.14 do_divide_error()	106
4.37.2.15 do_double_fault()	106
4.37.2.16 do_general_protection()	106
4.37.2.17 do_invalid_op()	106
4.37.2.18 do_invalid_tss()	106
4.37.2.19 do_isr()	106
4.37.2.20 do_nmi()	106
4.37.2.21 do_overflow()	107
4.37.2.22 do_page_fault()	107
4.37.2.23 do_reserved()	107
4.37.2.24 do_segment_not_present()	107
4.37.2.25 do_stack_segment()	107
4.37.2.26 double_fault()	107
4.37.2.27 general_protection()	107
4.37.2.28 init_irq(void)	107
4.37.2.29 init_pic(void)	107
4.37.2.30 invalid_op()	107
4.37.2.31 invalid_tss()	107
4.37.2.32 isr0()	107
4.37.2.33 nmi()	107
4.37.2.34 overflow()	107
4.37.2.35 page_fault()	107
4.37.2.36 reserved()	107
4.37.2.37 rtc_isr()	107
4.37.2.38 segment_not_present()	107
4.37.2.39 stack_segment()	107
4.37.2.40 sys_call_isr()	107
4.37.3 Variable Documentation	108

4.37.3.1	idt_entries	108
4.38	kernel/core/kmain.c File Reference	108
4.38.1	Function Documentation	108
4.38.1.1	kmain(void)	108
4.39	kernel/core/pcb.c File Reference	108
4.39.1	Function Documentation	109
4.39.1.1	allocatePCB()	109
4.39.1.2	checkParamClass(int processClass)	109
4.39.1.3	checkParamName(const char *processName)	110
4.39.1.4	checkParamPriority(int priority)	110
4.39.1.5	freePCB(pcb *pcbPtr)	110
4.39.1.6	setupPCB(const char *processName, int processClass, int priority)	111
4.40	kernel/core/queue.c File Reference	111
4.40.1	Enumeration Type Documentation	112
4.40.1.1	queue	112
4.40.2	Function Documentation	112
4.40.2.1	_findNode(const char *processName)	112
4.40.2.2	_findNodeInQueue(queue q, const char *processName)	113
4.40.2.3	_insertFIFO(queue q, node *newNode)	113
4.40.2.4	_insertPriority(queue q, node *newNode)	113
4.40.2.5	_newNode(pcb *p)	113
4.40.2.6	findPCB(const char *processName)	114
4.40.2.7	getBlockedQueue()	114
4.40.2.8	getReadyQueue()	114
4.40.2.9	getSuspendedBlockedQueue()	114
4.40.2.10	getSuspendedReadyQueue()	115
4.40.2.11	insertPCB(pcb *p)	115
4.40.2.12	popBlocked()	115
4.40.2.13	popReady()	115
4.40.2.14	popSuspendedBlocked()	115

4.40.2.15 popSuspendedReady()	116
4.40.2.16 removePCB(pcb *p)	116
4.40.3 Variable Documentation	116
4.40.3.1 queues	116
4.41 kernel/core/serial.c File Reference	116
4.41.1 Macro Definition Documentation	117
4.41.1.1 NO_ERROR	117
4.41.2 Function Documentation	117
4.41.2.1 init_serial(int device)	117
4.41.2.2 serial_print(const char *msg)	117
4.41.2.3 serial_println(const char *msg)	118
4.41.2.4 set_serial_in(int device)	118
4.41.2.5 set_serial_out(int device)	118
4.41.3 Variable Documentation	118
4.41.3.1 serial_port_in	118
4.41.3.2 serial_port_out	119
4.42 kernel/core/system.c File Reference	119
4.42.1 Function Documentation	119
4.42.1.1 klogv(const char *msg)	119
4.42.1.2 kpanic(const char *msg)	119
4.43 kernel/core/tables.c File Reference	120
4.43.1 Function Documentation	121
4.43.1.1 gdt_init_entry(int idx, u32int base, u32int limit, u8int access, u8int flags)	121
4.43.1.2 idt_set_gate(u8int idx, u32int base, u16int sel, u8int flags)	121
4.43.1.3 init_gdt()	121
4.43.1.4 init_idt()	121
4.43.1.5 write_gdt_ptr(u32int, size_t)	121
4.43.1.6 write_idt_ptr(u32int)	121
4.43.2 Variable Documentation	121
4.43.2.1 gdt_entries	121

4.43.2.2	<code>gdt_ptr</code>	121
4.43.2.3	<code>idt_entries</code>	122
4.43.2.4	<code>idt_ptr</code>	122
4.44	kernel/mem/heap.c File Reference	122
4.44.1	Function Documentation	122
4.44.1.1	<code>_kmalloc(u32int size, int page_align, u32int *phys_addr)</code>	122
4.44.1.2	<code>alloc(u32int size, heap *h, int align)</code>	123
4.44.1.3	<code>kmalloc(u32int size)</code>	123
4.44.1.4	<code>make_heap(u32int base, u32int max, u32int min)</code>	123
4.44.2	Variable Documentation	124
4.44.2.1	<code>__end</code>	124
4.44.2.2	<code>_end</code>	124
4.44.2.3	<code>curr_heap</code>	124
4.44.2.4	<code>end</code>	124
4.44.2.5	<code>kdir</code>	124
4.44.2.6	<code>kheap</code>	124
4.44.2.7	<code>phys_alloc_addr</code>	124
4.45	kernel/mem/memoryControl.c File Reference	124
4.45.1	Function Documentation	125
4.45.1.1	<code>_mergeAdjacentFree()</code>	125
4.45.1.2	<code>_placeStructs(int size, void *pos, int type, cmcb *prev, cmcb *next)</code>	125
4.45.1.3	<code>allocateMemory(int size)</code>	125
4.45.1.4	<code>deallocateMemory(void *memPointer)</code>	126
4.45.1.5	<code>getAllocatedHead()</code>	126
4.45.1.6	<code>getFreeHead()</code>	126
4.45.1.7	<code>initializeHeap(int size)</code>	126
4.45.1.8	<code>isEmpty()</code>	127
4.45.2	Variable Documentation	127
4.45.2.1	<code>allocatedHead</code>	127
4.45.2.2	<code>freeHead</code>	127

4.45.2.3	isInitialized	127
4.45.2.4	memAllocated	127
4.45.2.5	memHeap	127
4.45.2.6	memSize	127
4.46	kernel/mem/paging.c File Reference	127
4.46.1	Function Documentation	128
4.46.1.1	clear_bit(u32int addr)	128
4.46.1.2	first_free()	129
4.46.1.3	get_bit(u32int addr)	129
4.46.1.4	get_page(u32int addr, page_dir *dir, int make_table)	129
4.46.1.5	init_paging()	129
4.46.1.6	load_page_dir(page_dir *new_dir)	130
4.46.1.7	new_frame(page_entry *page)	130
4.46.1.8	set_bit(u32int addr)	130
4.46.2	Variable Documentation	130
4.46.2.1	cdir	130
4.46.2.2	frames	130
4.46.2.3	kdir	130
4.46.2.4	kheap	130
4.46.2.5	mem_size	130
4.46.2.6	nframes	130
4.46.2.7	page_size	130
4.46.2.8	phys_alloc_addr	131
4.47	lib/math.c File Reference	131
4.47.1	Function Documentation	131
4.47.1.1	bcdToDec(unsigned char bcd)	131
4.47.1.2	decToBcd(unsigned char dec)	131
4.48	lib/regex.c File Reference	132
4.48.1	Function Documentation	132
4.48.1.1	testRegex(const char *regex, const char *stringToCheck)	132

4.49 lib/string.c File Reference	133
4.49.1 Function Documentation	134
4.49.1.1 atoi(const char *s)	134
4.49.1.2 isChar(const char c)	134
4.49.1.3 isdigit(const char c)	134
4.49.1.4 isLowerChar(const char c)	134
4.49.1.5 isspace(const char *c)	135
4.49.1.6 isUpperChar(const char c)	135
4.49.1.7 itoa(int num, char *str, int base)	135
4.49.1.8 reverse(char *str, int len)	135
4.49.1.9 strcat(char *s1, const char *s2)	136
4.49.1.10 strcmp(const char *s1, const char *s2)	136
4.49.1.11 strcpy(char *cpy, const char *ori)	136
4.49.1.12 strlen(const char *s)	136
4.49.1.13 strtok(char *s1, const char *s2)	137
4.50 lib/time.c File Reference	137
4.50.1 Function Documentation	138
4.50.1.1 getDateTime()	138
4.50.1.2 getDayOfMonth()	138
4.50.1.3 getDayOfWeek()	138
4.50.1.4 getHours()	139
4.50.1.5 getMinutes()	139
4.50.1.6 getMonth()	139
4.50.1.7 getSeconds()	139
4.50.1.8 getYear()	139
4.50.1.9 isLeapYear(int year)	139
4.50.1.10 setDateTime(date_time dateTime)	140
4.50.1.11 setDayOfMonth(unsigned char day)	140
4.50.1.12 setDayOfWeek(unsigned char day)	140
4.50.1.13 setHours(unsigned char hour)	140

4.50.1.14	setMinutes(unsigned char min)	141
4.50.1.15	setMonth(unsigned char mon)	141
4.50.1.16	setSeconds(unsigned char sec)	141
4.50.1.17	setYear(unsigned char year)	141
4.50.1.18	updateDayOfWeek(date_time *dateTime)	141
4.50.1.19	updateDayOfYear(date_time *dateTime)	142
4.50.2	Variable Documentation	142
4.50.2.1	DAYS_IN_MONTH	142
4.51	modules/mpx_supt.c File Reference	142
4.51.1	Function Documentation	143
4.51.1.1	getCOPName()	143
4.51.1.2	idle()	143
4.51.1.3	memset(void *s, int c, size_t n)	143
4.51.1.4	mpx_init(int cur_mod)	144
4.51.1.5	sys_alloc_mem(u32int size)	144
4.51.1.6	sys_call(context *registers)	144
4.51.1.7	sys_free_mem(void *ptr)	144
4.51.1.8	sys_req(int op_code)	145
4.51.1.9	sys_set_free(boolean(func)(void *))	145
4.51.1.10	sys_set_malloc(void *(*func)(int))	145
4.51.2	Variable Documentation	145
4.51.2.1	callerContext	145
4.51.2.2	cop	145
4.51.2.3	current_module	145
4.51.2.4	params	145
4.51.2.5	student_free	145
4.51.2.6	student_malloc	146
4.52	modules/R2/commands/perm.c File Reference	146
4.52.1	Function Documentation	146
4.52.1.1	printPcbInfo(pcb *p)	146

4.52.1.2	printQueueInfo(node *queue)	146
4.52.1.3	registerR2PermCommands()	146
4.52.1.4	resumePcb(char **args, int numArgs)	146
4.52.1.5	setPriorityPcb(char **args, int numArgs)	147
4.52.1.6	showPcbInfo(char **args, int numArgs)	147
4.52.1.7	suspendPcb(char **args, int numArgs)	148
4.53	modules/R2/commands/temp.c File Reference	148
4.53.1	Function Documentation	149
4.53.1.1	blockPcb(char **args, int numArgs)	149
4.53.1.2	createPcb(char **args, int numArgs)	150
4.53.1.3	deletePcb(char **args, int numArgs)	150
4.53.1.4	registerR2TempCommands()	151
4.53.1.5	unblockPcb(char **args, int numArgs)	151
4.54	modules/R3/commands/r3commands.c File Reference	151
4.54.1	Macro Definition Documentation	152
4.54.1.1	P1_NAME	152
4.54.1.2	P2_NAME	152
4.54.1.3	P3_NAME	152
4.54.1.4	P4_NAME	152
4.54.1.5	P5_NAME	152
4.54.2	Function Documentation	152
4.54.2.1	loadr3(char **args, int numArgs)	152
4.54.2.2	registerR3Commands()	152
4.54.2.3	yield(char **args, int numArgs)	153
4.55	modules/R3/procsr3.c File Reference	153
4.55.1	Macro Definition Documentation	154
4.55.1.1	RC_1	154
4.55.1.2	RC_2	154
4.55.1.3	RC_3	154
4.55.1.4	RC_4	154

4.55.1.5	RC_5	154
4.55.2	Function Documentation	154
4.55.2.1	proc1()	154
4.55.2.2	proc2()	154
4.55.2.3	proc3()	154
4.55.2.4	proc4()	154
4.55.2.5	proc5()	154
4.56	modules/R5/commands/r5commands.c File Reference	154
4.56.1	Function Documentation	155
4.56.1.1	printBlockInfo(cmbc *blockList)	155
4.56.1.2	printCmbcInfo(cmbc *block)	155
4.56.1.3	registerR5PermCommands()	155
4.56.1.4	showMemory(char **args, int numArgs)	155
4.57	modules/R5/memCommands.c File Reference	155
4.57.1	Function Documentation	156
4.57.1.1	allocateMem(char **args, int numArgs)	156
4.57.1.2	freeMemory(char **args, int numArgs)	156
4.57.1.3	initHeap(char **args, int numArgs)	157
4.57.1.4	isEmptyCom(char **args, int numArgs)	157
4.57.1.5	registerR5TempCommands()	157
4.58	r6/fat.c File Reference	157
4.58.1	Function Documentation	158
4.58.1.1	_getDiskOffsetForDirEntry(int idx)	158
4.58.1.2	_getFirstFreeIndexInDirs(dir_entry *dirs, int maxSize)	159
4.58.1.3	_getFirstFreeIndexInSector(dir_entry *dirs)	159
4.58.1.4	_getFirstOpenSector()	159
4.58.1.5	_loadBootSectorInfo()	159
4.58.1.6	_loadFATTables()	159
4.58.1.7	_loadRootDirectroy()	159
4.58.1.8	_loadSectorAsDirectoryEntries(uint16_t sector)	159

4.58.1.9	_prepNewDirSector(uint16_t sector)	160
4.58.1.10	_readDirectoryEntry(dir_entry *dir)	160
4.58.1.11	_refreshDirectory()	160
4.58.1.12	_saveDirEntry(dir_entry *dir)	160
4.58.1.13	_saveFATTables()	160
4.58.1.14	changeToDirectory(uint16_t sector)	160
4.58.1.15	changeToParentDirectory()	161
4.58.1.16	destroy()	161
4.58.1.17	getBootSector()	161
4.58.1.18	getCurrentDirectory()	161
4.58.1.19	getCurrentDirectoryMaxSize()	161
4.58.1.20	getFATTables()	161
4.58.1.21	getFileFromSector(uint16_t sector, int size)	161
4.58.1.22	initialize(FILE *diskImage)	162
4.58.1.23	moveFile(int idx, uint16_t destSector)	162
4.58.1.24	setFilename(int idx, const char *filename, const char *fileExt)	162
4.58.2	Variable Documentation	162
4.58.2.1	_BootSector	162
4.58.2.2	_CurrDirSize	162
4.58.2.3	_CurrentDirectory	163
4.58.2.4	_DiskImage	163
4.58.2.5	_FATTables	163
4.58.2.6	_isCurrentRoot	163
4.59	r6/fat.h File Reference	163
4.59.1	Macro Definition Documentation	164
4.59.1.1	ARCHIVE	164
4.59.1.2	BAD_CLUSTER	164
4.59.1.3	BOOT_SECTOR_OFFSET	164
4.59.1.4	DATA_AREA_OFFSET	164
4.59.1.5	DELETED	164

4.59.1.6	DIR_ENTRY_SIZE	165
4.59.1.7	FAT1_OFFSET	165
4.59.1.8	FAT2_OFFSET	165
4.59.1.9	HIDDEN	165
4.59.1.10	LAST_CLUSTER_BEGIN	165
4.59.1.11	LAST_CLUSTER_END	165
4.59.1.12	MAX_EXT_LENGTH	165
4.59.1.13	MAX_FILENAME_LENGTH	165
4.59.1.14	READ_ONLY	165
4.59.1.15	REMAINING_FREE	165
4.59.1.16	RESERVED_CLUSTER_BEGIN	165
4.59.1.17	RESERVED_CLUSTER_END	165
4.59.1.18	ROOT_DIRECTORY_OFFSET	165
4.59.1.19	SUBDIRECTORY	165
4.59.1.20	SYSTEM	165
4.59.1.21	UNUSED	165
4.59.1.22	VOLUME_LABEL	165
4.59.2	Function Documentation	165
4.59.2.1	changeToDirectory(uint16_t cluster)	165
4.59.2.2	changeToParentDirectory()	166
4.59.2.3	destroy()	166
4.59.2.4	getBootSector()	166
4.59.2.5	getCurrentDirectory()	166
4.59.2.6	getCurrentDirectoryMaxSize()	167
4.59.2.7	getFATTables()	167
4.59.2.8	getFileFromSector(uint16_t cluster, int size)	167
4.59.2.9	initialize(FILE *diskImage)	167
4.59.2.10	moveFile(int idx, uint16_t destSector)	168
4.59.2.11	setFilename(int idx, const char *filename, const char *fileExt)	168
4.60	r6/main.c File Reference	168

4.60.1	Function Documentation	169
4.60.1.1	<code>_callCommand(char *command)</code>	169
4.60.1.2	<code>_extcmp(const char *n1, const char *n2)</code>	169
4.60.1.3	<code>_fncmp(const char *n1, const char *n2)</code>	169
4.60.1.4	<code>_getClusterOfFileWithName(const char *name, const char *ext)</code>	170
4.60.1.5	<code>_getIndexOfFileWithName(const char *name, const char *ext)</code>	170
4.60.1.6	<code>_getSizeOfFileWithName(const char *name, const char *ext)</code>	170
4.60.1.7	<code>_launchCommandInterface()</code>	170
4.60.1.8	<code>_nameCmpHelper(const char *n1, const char *n2, int maxElements)</code>	170
4.60.1.9	<code>_printBootSectorInfo()</code>	170
4.60.1.10	<code>_printDirectoryEntries(dir_entry *entries, int maxEntries)</code>	171
4.60.1.11	<code>_printDirectoryEntriesByFileName(dir_entry *entries, int maxEntries, char *name, char *fileExt)</code>	171
4.60.1.12	<code>_printDirectoryEntriesByType(dir_entry *entries, int maxEntries, char *ext)</code>	171
4.60.1.13	<code>_printFATTableInfo()</code>	171
4.60.1.14	<code>_printFile(uint16_t sector, int fileSize, bool pag)</code>	171
4.60.1.15	<code>main(int numArgs, char *args[])</code>	171
4.60.2	Variable Documentation	171
4.60.2.1	<code>depth</code>	171
4.60.2.2	<code>diskImage</code>	171
4.60.2.3	<code>filename</code>	171
4.60.2.4	<code>imageName</code>	171
4.60.2.5	<code>paths</code>	171
4.60.2.6	<code>printFileFlag</code>	171

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

boot_sector	5
cmcb	7
context	8
date_time	9
dir_entry	9
fat_tables	11
footer	11
functionDef	12
gdt_descriptor_struct	12
gdt_entry_struct	13
header	14
heap	14
idt_entry_struct	15
idt_struct	16
index_entry	16
index_table	17
lmcb	17
node	18
page_dir	19
page_entry	20
page_table	21
param	21
pcb	22

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

include/ boolean.h	23
include/ math.h	51
include/ regex.h	83
include/ string.h	84
include/ system.h	88
include/ time.h	91
include/core/ asm.h	24
include/core/ comHandler.h	24
include/core/ commands.h	28
include/core/ help.h	30
include/core/ interrupts.h	37
include/core/ io.h	38
include/core/ pcb.h	39
include/core/ queue.h	42
include/core/ serial.h	46
include/core/ tables.h	48
include/core/ version.h	50
include/mem/ heap.h	52
include/mem/ memoryControl.h	55
include/mem/ paging.h	58
include/modules/ mpx_supt.h	61
include/modules/R2/commands/ help.h	32
include/modules/R2/commands/ help_temp.h	67
include/modules/R2/commands/ perm.h	68
include/modules/R2/commands/ status.h	72
include/modules/R2/commands/ status_temp.h	73
include/modules/R2/commands/ temp.h	74
include/modules/R3/ processes.h	78
include/modules/R3/commands/ help.h	34
include/modules/R3/commands/ r3commands.h	76
include/modules/R5/ help.h	36
include/modules/R5/ memCommands.h	81
include/modules/R5/commands/ help.h	35
include/modules/R5/commands/ r5commands.h	79
kernel/core/ comHandler.c	98

kernel/core/commands.c	103
kernel/core/interrupts.c	104
kernel/core/kmain.c	108
kernel/core/pcb.c	108
kernel/core/queue.c	111
kernel/core/serial.c	116
kernel/core/system.c	119
kernel/core/tables.c	120
kernel/mem/heap.c	122
kernel/mem/memoryControl.c	124
kernel/mem/paging.c	127
lib/math.c	131
lib/regex.c	132
lib/string.c	133
lib/time.c	137
modules/mpx_supt.c	142
modules/R2/commands/perm.c	146
modules/R2/commands/temp.c	148
modules/R3/procsr3.c	153
modules/R3/commands/r3commands.c	151
modules/R5/memCommands.c	155
modules/R5/commands/r5commands.c	154
r6/fat.c	157
r6/fat.h	163
r6/main.c	168

Chapter 3

Data Structure Documentation

3.1 boot_sector Struct Reference

```
#include <fat.h>
```

Data Fields

- `uint8_t ignore_1` [11]
- `uint16_t bytesPerSector`
- `uint8_t sectorsPerCluster`
- `uint16_t numReservedSectors`
- `uint8_t numFATCopies`
- `uint16_t maxRootDirEntries`
- `uint16_t numSectors`
- `uint8_t ignore_2` [1]
- `uint16_t sectorsPerFAT`
- `uint16_t sectorsPerTrack`
- `uint16_t numHeads`
- `uint8_t ignore_3` [4]
- `uint32_t sectorCountFAT32`
- `uint8_t ignore_4` [2]
- `uint8_t bootSig`
- `uint32_t volId`
- `unsigned char volName` [12]
- `unsigned char fileType` [9]
- `uint8_t ignore_5` [450]

3.1.1 Detailed Description

Struct representing the boot sector

3.1.2 Field Documentation

3.1.2.1 `uint8_t bootSig`

3.1.2.2 `uint16_t bytesPerSector`

3.1.2.3 `unsigned char fileType[9]`

3.1.2.4 `uint8_t ignore_1[11]`

3.1.2.5 `uint8_t ignore_2[1]`

3.1.2.6 `uint8_t ignore_3[4]`

3.1.2.7 `uint8_t ignore_4[2]`

3.1.2.8 `uint8_t ignore_5[450]`

3.1.2.9 `uint16_t maxRootDirEntries`

3.1.2.10 `uint8_t numFATCopies`

3.1.2.11 `uint16_t numHeads`

3.1.2.12 `uint16_t numReservedSectors`

3.1.2.13 `uint16_t numSectors`

3.1.2.14 `uint32_t sectorCountFAT32`

3.1.2.15 `uint8_t sectorsPerCluster`

3.1.2.16 `uint16_t sectorsPerFAT`

3.1.2.17 `uint16_t sectorsPerTrack`

3.1.2.18 `uint32_t volId`

3.1.2.19 `unsigned char volName[12]`

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

- [r6/fat.h](#)

3.2 cmcb Struct Reference

```
#include <memoryControl.h>
```

Collaboration diagram for cmcb:



Data Fields

- int [type](#)
- void * [beginningAddr](#)
- int [size](#)
- int [memSize](#)
- const char * [name](#)
- struct [cmcb](#) * [next](#)
- struct [cmcb](#) * [prev](#)

3.2.1 Field Documentation

3.2.1.1 void* [beginningAddr](#)

3.2.1.2 int [memSize](#)

3.2.1.3 const char* [name](#)

3.2.1.4 struct [cmcb](#)* [next](#)

3.2.1.5 struct [cmcb](#)* [prev](#)

3.2.1.6 int [size](#)

3.2.1.7 int [type](#)

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

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

3.3 context Struct Reference

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

Data Fields

- [u32int gs](#)
- [u32int fs](#)
- [u32int es](#)
- [u32int ds](#)
- [u32int edi](#)
- [u32int esi](#)
- [u32int ebp](#)
- [u32int esp](#)
- [u32int ebx](#)
- [u32int edx](#)
- [u32int ecx](#)
- [u32int eax](#)
- [u32int eip](#)
- [u32int cs](#)
- [u32int eflags](#)

3.3.1 Field Documentation

3.3.1.1 [u32int cs](#)

3.3.1.2 [u32int ds](#)

3.3.1.3 [u32int eax](#)

3.3.1.4 [u32int ebp](#)

3.3.1.5 [u32int ebx](#)

3.3.1.6 [u32int ecx](#)

3.3.1.7 [u32int edi](#)

3.3.1.8 [u32int edx](#)

3.3.1.9 [u32int eflags](#)

3.3.1.10 [u32int eip](#)

3.3.1.11 [u32int es](#)

3.3.1.12 [u32int esi](#)

3.3.1.13 [u32int esp](#)

3.3.1.14 [u32int fs](#)

3.3.1.15 [u32int gs](#)

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

- [include/modules/mpx_supt.h](#)

3.4 date_time Struct Reference

```
#include <time.h>
```

Data Fields

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

3.4.1 Detailed Description

Structure representing a date and time.

3.4.2 Field Documentation

3.4.2.1 int day_m

3.4.2.2 int day_w

3.4.2.3 int day_y

3.4.2.4 int hour

3.4.2.5 int min

3.4.2.6 int mon

3.4.2.7 int sec

3.4.2.8 int year

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

- include/[time.h](#)

3.5 dir_entry Struct Reference

```
#include <fat.h>
```

Data Fields

- unsigned char [filename](#) [9]
- unsigned char [extension](#) [4]
- uint8_t [attributes](#)
- uint16_t [reserved](#)
- uint16_t [creationTime](#)
- uint16_t [creationDate](#)
- uint16_t [lastAccess](#)
- uint16_t [ignore](#)
- uint16_t [lastWriteTime](#)
- uint16_t [lastWriteDate](#)
- uint16_t [firstLogicalCluster](#)
- uint32_t [fileSize](#)

3.5.1 Detailed Description

Struct representing a directory entry

3.5.2 Field Documentation

3.5.2.1 uint8_t attributes

3.5.2.2 uint16_t creationDate

3.5.2.3 uint16_t creationTime

3.5.2.4 unsigned char extension[4]

3.5.2.5 unsigned char filename[9]

3.5.2.6 uint32_t fileSize

3.5.2.7 uint16_t firstLogicalCluster

3.5.2.8 uint16_t ignore

3.5.2.9 uint16_t lastAccess

3.5.2.10 uint16_t lastWriteDate

3.5.2.11 uint16_t lastWriteTime

3.5.2.12 uint16_t reserved

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

- [r6/fat.h](#)

3.6 fat_tables Struct Reference

```
#include <fat.h>
```

Data Fields

- int [numEntries](#)
- uint16_t * [fat1](#)
- uint16_t * [fat2](#)

3.6.1 Detailed Description

Struct containing the FAT table information

3.6.2 Field Documentation

3.6.2.1 uint16_t* fat1

3.6.2.2 uint16_t* fat2

3.6.2.3 int numEntries

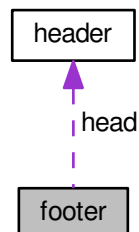
The documentation for this struct was generated from the following file:

- [r6/fat.h](#)

3.7 footer Struct Reference

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

Collaboration diagram for footer:



Data Fields

- [header head](#)

3.7.1 Detailed Description

Heap allocation footer.

3.7.2 Field Documentation

3.7.2.1 header head

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

- `include/mem/heap.h`

3.8 functionDef Struct Reference

```
#include <comHandler.h>
```

Data Fields

- `char * name`
- `const char * helpString`
- `const char *(* funcPointer)(char **args, int numArgs)`

3.8.1 Field Documentation

3.8.1.1 `const char *(* funcPointer)(char **args, int numArgs)`

3.8.1.2 `const char* helpString`

3.8.1.3 `char* name`

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

- `include/core/comHandler.h`

3.9 gdt_descriptor_struct Struct Reference

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

Data Fields

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

3.9.1 Field Documentation

3.9.1.1 u32int base

3.9.1.2 u16int limit

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

- `include/core/tables.h`

3.10 gdt_entry_struct Struct Reference

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

Data Fields

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

3.10.1 Field Documentation

3.10.1.1 u8int access

3.10.1.2 u8int base_high

3.10.1.3 u16int base_low

3.10.1.4 u8int base_mid

3.10.1.5 u8int flags

3.10.1.6 u16int limit_low

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

- `include/core/tables.h`

3.11 header Struct Reference

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

Data Fields

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

3.11.1 Detailed Description

Heap allocation header.

3.11.2 Field Documentation

3.11.2.1 int index_id

3.11.2.2 int size

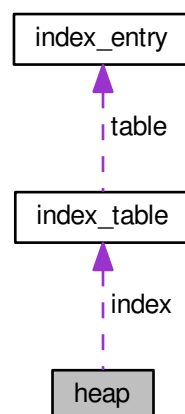
The documentation for this struct was generated from the following file:

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

3.12 heap Struct Reference

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

Collaboration diagram for heap:



Data Fields

- [index_table](#) index
- [u32int](#) base
- [u32int](#) max_size
- [u32int](#) min_size

3.12.1 Detailed Description

Heap structure

3.12.2 Field Documentation

3.12.2.1 [u32int](#) base

3.12.2.2 [index_table](#) index

3.12.2.3 [u32int](#) max_size

3.12.2.4 [u32int](#) min_size

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

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

3.13 idt_entry_struct Struct Reference

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

Data Fields

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

3.13.1 Field Documentation

3.13.1.1 [u16int](#) base_high

3.13.1.2 [u16int](#) base_low

3.13.1.3 [u8int](#) flags

3.13.1.4 [u16int](#) sselect

3.13.1.5 [u8int](#) zero

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

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

3.14 idt_struct Struct Reference

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

Data Fields

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

3.14.1 Field Documentation

3.14.1.1 u32int base

3.14.1.2 u16int limit

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

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

3.15 index_entry Struct Reference

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

Data Fields

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

3.15.1 Field Documentation

3.15.1.1 u32int block

3.15.1.2 int empty

3.15.1.3 int size

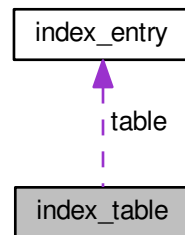
The documentation for this struct was generated from the following file:

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

3.16 index_table Struct Reference

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

Collaboration diagram for index_table:



Data Fields

- [index_entry table](#) [TABLE_SIZE]
- [int id](#)

3.16.1 Detailed Description

Kernel heap index table.

3.16.2 Field Documentation

3.16.2.1 int id

3.16.2.2 index_entry table[TABLE_SIZE]

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

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

3.17 Imcb Struct Reference

```
#include <memoryControl.h>
```

Data Fields

- int [type](#)
- int [size](#)
- int [memSize](#)

3.17.1 Field Documentation

3.17.1.1 int memSize

3.17.1.2 int size

3.17.1.3 int type

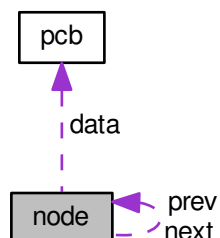
The documentation for this struct was generated from the following file:

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

3.18 node Struct Reference

```
#include <queue.h>
```

Collaboration diagram for node:



Data Fields

- struct [pcb](#) * [data](#)
- struct [node](#) * [next](#)
- struct [node](#) * [prev](#)

3.18.1 Detailed Description

The struct representing a node in a queue

3.18.2 Field Documentation

3.18.2.1 struct pcb* data

3.18.2.2 struct node* next

3.18.2.3 struct node* prev

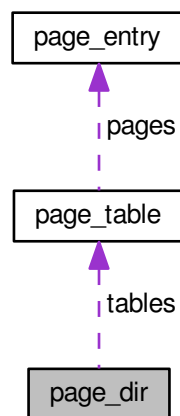
The documentation for this struct was generated from the following file:

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

3.19 page_dir Struct Reference

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

Collaboration diagram for page_dir:



Data Fields

- [page_table](#) * [tables](#) [1024]
- [u32int](#) [tables_phys](#) [1024]

3.19.1 Detailed Description

Page directory structure Limited to 1024 tables for now

3.19.2 Field Documentation

3.19.2.1 `page_table*` `tables[1024]`

3.19.2.2 `u32int` `tables_phys[1024]`

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

- `include/mem/paging.h`

3.20 `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

3.20.1 Detailed Description

Page entry structure Describes a single page in memory

3.20.2 Field Documentation

3.20.2.1 `u32int` `accessed`

3.20.2.2 `u32int` `dirty`

3.20.2.3 `u32int` `frameaddr`

3.20.2.4 `u32int` `present`

3.20.2.5 `u32int` `reserved`

3.20.2.6 `u32int` `usermode`

3.20.2.7 `u32int` `writeable`

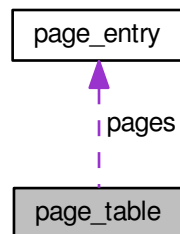
The documentation for this struct was generated from the following file:

- `include/mem/paging.h`

3.21 page_table Struct Reference

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

Collaboration diagram for page_table:



Data Fields

- [page_entry pages](#) [1024]

3.21.1 Detailed Description

Page table structure Contains 1024 pages/frames

3.21.2 Field Documentation

3.21.2.1 [page_entry pages](#)[1024]

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

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

3.22 param Struct Reference

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

Data Fields

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

3.22.1 Field Documentation

3.22.1.1 int device_id

3.22.1.2 int op_code

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

- include/modules/[mpx_supt.h](#)

3.23 pcb Struct Reference

```
#include <pcb.h>
```

Data Fields

- char * [processName](#)
- int [processClass](#)
- int [priority](#)
- int [isSuspended](#)
- int [state](#)
- unsigned char * [stackTop](#)
- unsigned char * [stackBottom](#)

3.23.1 Field Documentation

3.23.1.1 int isSuspended

3.23.1.2 int priority

3.23.1.3 int processClass

3.23.1.4 char* processName

3.23.1.5 unsigned char* stackBottom

3.23.1.6 unsigned char* stackTop

3.23.1.7 int state

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

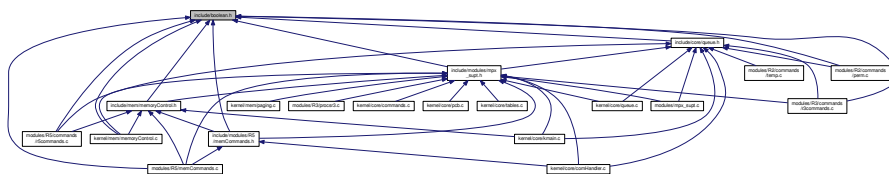
- include/core/[pcb.h](#)

Chapter 4

File Documentation

4.1 include/boolean.h File Reference

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



Enumerations

- enum `boolean` { `false` = 0, `true` = 1 }

4.1.1 Enumeration Type Documentation

4.1.1.1 enum `boolean`

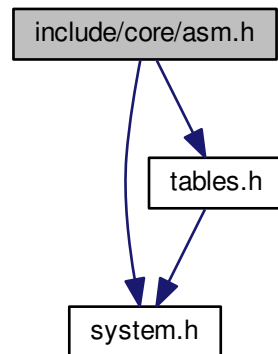
Enumerator

false

true

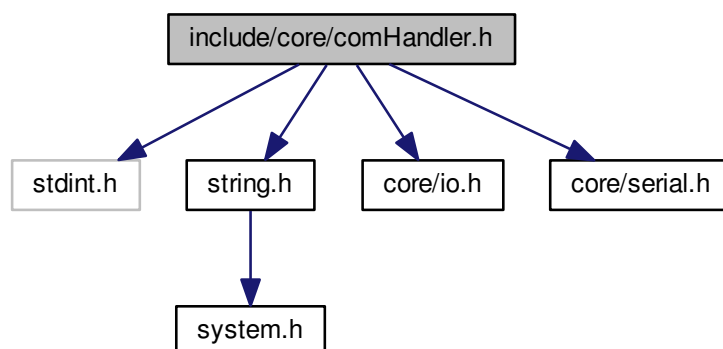
4.2 include/core/asm.h File Reference

```
#include <system.h>
#include <tables.h>
Include dependency graph for asm.h:
```

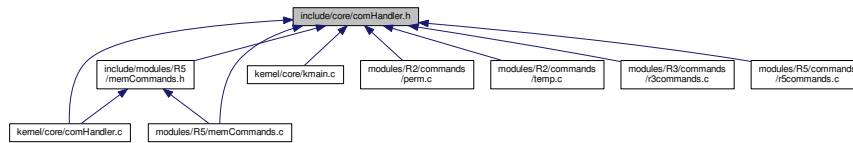


4.3 include/core/comHandler.h File Reference

```
#include <stdint.h>
#include <string.h>
#include <core/io.h>
#include <core/serial.h>
Include dependency graph for comHandler.h:
```



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



Data Structures

- struct [functionDef](#)

Functions

- void [addFunctionDef](#) (char *name, const char *helpString, const char *(funcPointer)(char **args, int numArgs))
- [functionDef](#) [getFunctionDef](#) (char *name)
- const char * [getHelpString](#) (char *name)
- char * [getComHistory](#) (int isPrev)
- void [addComHistory](#) (char *newCom)
- void [printStart](#) ()
- void [returnToInsertionPoint](#) (int endIndex, int insertionIndex)
- void [eraseCurrentRow](#) (int endIndex, int insertionIndex)
- char * [getInput](#) ()
- void [executeCommand](#) (char *commandString)
- void [setupCommands](#) ()
- void [initCommandHandler](#) ()

4.3.1 Function Documentation

4.3.1.1 void addComHistory (char * newCom)

Helper function to add a command to the command history array

Parameters

<i>newCom</i>	string to add to the command history
---------------	--------------------------------------

4.3.1.2 void addFunctionDef (char * name, const char * helpString, const char * funcPointer)(char **args, int numArgs)

Adds function definition struct, created from provided params to the functionDefs array This allows the function to be called in the command handler by its name

Parameters

<i>name</i>	- string representation of the function
-------------	-----------------------------------------

Parameters

<i>helpString</i>	- const string to be displayed for help
<i>funcPointer</i>	- pointer to the function, must return const char* and take in arguments: char** args and int numArgs

4.3.1.3 void eraseCurrentRow (int *endIndex*, int *insertionIndex*)

Helper function to remove all printed chars on the current line of input back to the >>

Parameters

<i>endIndex</i>	- index of last char printed
<i>insertionIndex</i>	- index of where insertion point should be

4.3.1.4 void executeCommand (char * *commandString*)

Gets the command in the given commandString param and executes it, printing the provided output string

Parameters

<i>commandString</i>	string containing the command name and any args
----------------------	-------------------------------------------------

4.3.1.5 char* getComHistory (int *isPrev*)

Helper function to get the next or previous command from the command history

Parameters

<i>isPrev</i>	integer denoting if to get the previous command
---------------	-------------------------------------------------

Returns

string of the command

4.3.1.6 functionDef getFunctionDef (char * *name*)

Gets the [functionDef](#) struct corresponding to the name provided, returns a [functionDef](#) with null funcPointer if none are found

Parameters

<i>name</i>	- name of the functionDef
-------------	-------------------------------------------

Returns

[functionDef](#)4.3.1.7 `const char* getHelpString (char * name)`

Gets the help string from the struct for the function name provided

Parameters

<i>name</i>	- name associated with the struct from which to get the help string
-------------	---------------------------------------------------------------------

Returns

const char* help string

4.3.1.8 `char* getInput ()`

Polls the input for characters and handles special key strokes such as delete, backspace, arrows, etc. and returns the input string

Returns

string that was input

4.3.1.9 `void initCommandHandler ()`

Main function of the comHandler that initializes the command handler, continually loops taking in input commands, manages the comHistory, and executes given commands

4.3.1.10 `void printStart ()`

Helper function to print out the beginning line tag: ">>"

4.3.1.11 `void returnToInsertionPoint (int endIndex, int insertionIndex)`

Helper function to move the insertion point from the end of the line to the correct placement

Parameters

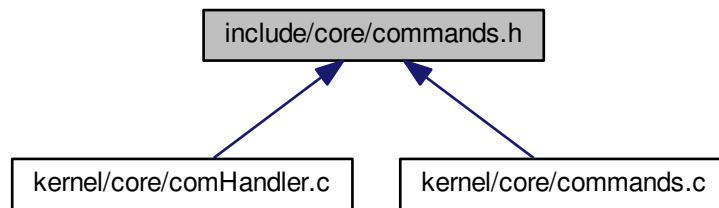
<i>endIndex</i>	- index of last char printed
<i>insertionIndex</i>	- index of where insertion point should be

4.3.1.12 void setupCommands ()

Initialization function to add commands ot the functionDefs array

4.4 include/core/commands.h File Reference

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



Functions

- const char * [version](#) (char **args, int numArgs)
- const char * [help](#) (char **args, int numArgs)
- const char * [shutdown](#) (char **args, int numArgs)
- const char * [date](#) (char **args, int numArgs)

4.4.1 Function Documentation

4.4.1.1 const char* date (char ** args, int numArgs)

Returns the current date/time in ISO-8601 format. Improperly specified date/times are ignored.

Usage: date [-date] [-time] [-setdate yyyy-MM-dd] [-settime hh:mm:ss]

Args: [no args] - Return the date and time -date - Return the date -time - Return the time -setdate - Sets the date to the specified date (returns the new date/time) -settime - Sets the time to the specified time (returns the new date/time)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

The ISO-8601 formatted date

Returns the current date/time in ISO-8601 format. Improperly specified date/times are ignored.

Usage: date [-date] [-time] [-setdate yyyy-MM-dd] [-settime hh:mm:ss]

Args: [no args] - Return the date and time -date - Return the date -time - Return the time -setdate - Sets the date to the specified date (returns the new date) -settime - Sets the time to the specified time (returns the new time)

Parameters

<i>args</i>	The arguments to pass to the function
-------------	---------------------------------------

Returns

The ISO-8601 formatted date

4.4.1.2 const char* help (char ** args, int numArgs)

Returns help for the specified commands.

Usage: help commandName

Args: [no args] - Returns the help for the help command commandName - The name of the command to get help for

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

The help string

4.4.1.3 const char* shutdown (char ** args, int numArgs)

Shuts down the OS after asking for confirmation.

Usage: shutdown [--confirm]

Args: [no args] - Displays confirmation prompt --confirm - Auto-confirms shutdown

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

True if shutdown was confirmed

4.4.1.4 const char* version (char ** args, int numArgs)

Returns the current version of the OS.

Usage: version

Args: [no args] - Returns the version

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

The version of the OS.

Returns the current version of the OS.

Usage: version

Args: [no args] - Returns the version

Parameters

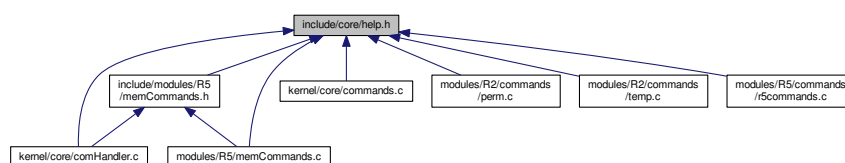
<i>args</i>	The arguments to pass to the function
-------------	---------------------------------------

Returns

The version of the OS.

4.5 include/core/help.h File Reference

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



Macros

- `#define` [HELP_UNKNOWN_COMMAND](#) ((const char*) "Unknown command.")
- `#define` [HELP_INVALID_ARGUMENTS](#) ((const char*) "Invalid arguments. Please check the documentation for this command.")
- `#define` [HELP_COMMAND_VERSION](#)
- `#define` [HELP_COMMAND_HELP](#)
- `#define` [HELP_COMMAND_SHUTDOWN](#)
- `#define` [HELP_COMMAND_DATE](#)

4.5.1 Macro Definition Documentation

4.5.1.1 `#define` [HELP_COMMAND_DATE](#)

Value:

```
((const char*) \
    "Prints the current date/time in ISO-8601 format.\n" \
    "Improperly specified date/times are ignored.\n" \
    "\n" \
    "Usage: date [--date] [--time] [--setdate yyyy-MM-dd] [--settime hh:mm:ss]\n" \
    "\n" \
    "Args:\n" \
    "    [no args] - Return the date and time\n" \
    "    --time - Return the time\n" \
    "    --date - Return the date\n" \
    "    --setdate - Sets the date to the specified date (returns the new date/time)\n" \
    "    --settime - Sets the time to the specified time (returns the new date/time)")
```

4.5.1.2 `#define` [HELP_COMMAND_HELP](#)

Value:

```
((const char*) \
    "Prints help for the specified commands.\n" \
    "\n" \
    "Usage: help commandName\n" \
    "\n" \
    "Args:\n" \
    "    [no args] - Returns the help for the help command\n" \
    "    commandName - The name of the command to get help for")
```

4.5.1.3 `#define` [HELP_COMMAND_SHUTDOWN](#)

Value:

```
((const char*) \
    "Shuts down the OS after asking for confirmation.\n" \
    "\n" \
    "Usage: shutdown [--confirm]\n" \
    "\n" \
    "Args:\n" \
    "    [no args] - Displays confirmation prompt\n" \
    "    --confirm - Auto-confirms shutdown")
```

4.5.1.4 #define HELP_COMMAND_VERSION

Value:

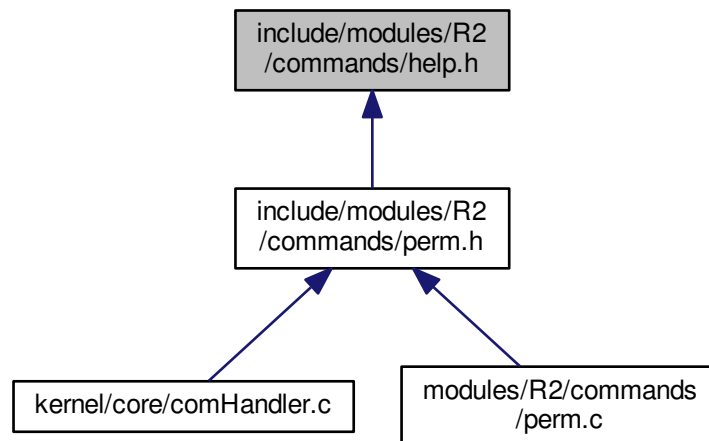
```
((const char*) \
  "Prints the current version of the OS.\n" \
  "\n" \
  "Usage: version\n" \
  "\n" \
  "Args:\n" \
  "    [no args] - Returns the version")
```

4.5.1.5 #define HELP_INVALID_ARGUMENTS ((const char*) "Invalid arguments. Please check the documentation for this command.")

4.5.1.6 #define HELP_UNKNOWN_COMMAND ((const char*) "Unknown command.")

4.6 include/modules/R2/commands/help.h File Reference

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



Macros

- #define [HELP_R2_COMMAND_SPCB](#)
- #define [HELP_R2_COMMAND_RPCB](#)
- #define [HELP_R2_COMMAND_PPCB](#)
- #define [HELP_R2_COMMAND_SHOWPCB](#)

4.6.1 Macro Definition Documentation

4.6.1.1 #define HELP_R2_COMMAND_PPCB

Value:

```
((const char*) \
  "Sets a PCB's priority and reinserts the process into the correct place in the correct queue.\n" \
  "\n" \
  "Usage: ppcb name priority\n" \
  "\n" \
  "Args:\n" \
  "    name - The name of the process to set the priority on (must exist)\n" \
  "    priority - The new priority (between 0 and 9)")
```

4.6.1.2 #define HELP_R2_COMMAND_RPCB

Value:

```
((const char*) \
  "Places a PCB into the not suspended state and reinserts it into the appropriate queue.\n" \
  "\n" \
  "Usage: rpcb name\n" \
  "\n" \
  "Args:\n" \
  "    name - The name of the process to resume (must exist)\n" \
  "    --all - Resumes all processes")
```

4.6.1.3 #define HELP_R2_COMMAND_SHOWPCB

Value:

```
((const char*) \
  "Displays the following information for the specified PCBs:\n" \
  "    Process Name:\n" \
  "    Class:\n" \
  "    State:\n" \
  "    Suspended Status:\n" \
  "    Priority:\n" \
  "\n" \
  "Usage: showpcb [--all] [--ready] [--blocked] [--name pcbName]\n" \
  "    (at least 1 must be specified)\n" \
  "\n" \
  "Args:\n" \
  "    [no args] - Shows the help for this command\n" \
  "    --all - Displays information for all PCBs\n" \
  "    --ready - Displays information for ready PCBs\n" \
  "    --blocked - Displays information for blocked PCBs\n" \
  "    --suspended - Displays information for suspended PCBs\n" \
  "    --name - Displays information for the specified PCB (can be used multiple times)")
```

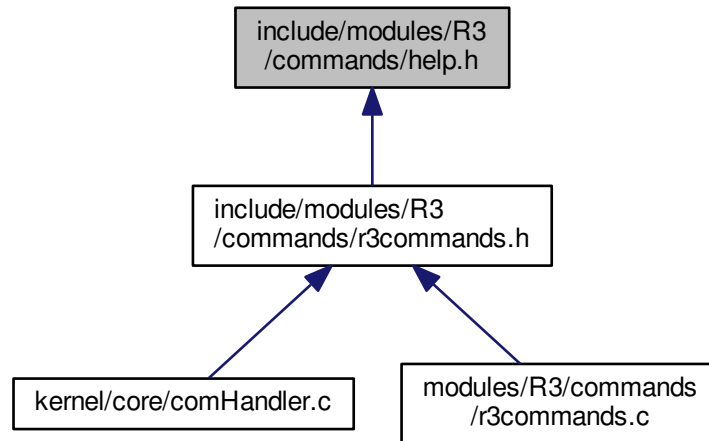
4.6.1.4 #define HELP_R2_COMMAND_SPCB

Value:

```
((const char*) \
  "Places a PCB into the suspended state and reinserts it into the appropriate queue.\n" \
  "\n" \
  "Usage: spcb name\n" \
  "\n" \
  "Args:\n" \
  "    name - The name of the process to suspend (must exist)")
```

4.7 include/modules/R3/commands/help.h File Reference

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



Macros

- `#define` [HELP_R3_COMMAND_YIELD](#)
- `#define` [HELP_R3_COMMAND_LOAD](#)

4.7.1 Macro Definition Documentation

4.7.1.1 `#define` [HELP_R3_COMMAND_LOAD](#)

Value:

```
((const char*) \
    "Loads the r3 processes to the queue.\n"\
    "\n"\
    "Usage: loadr3\n"\
    "\n"\
    "Args:\n"\
    "    [no args] - loads processes\n")
```

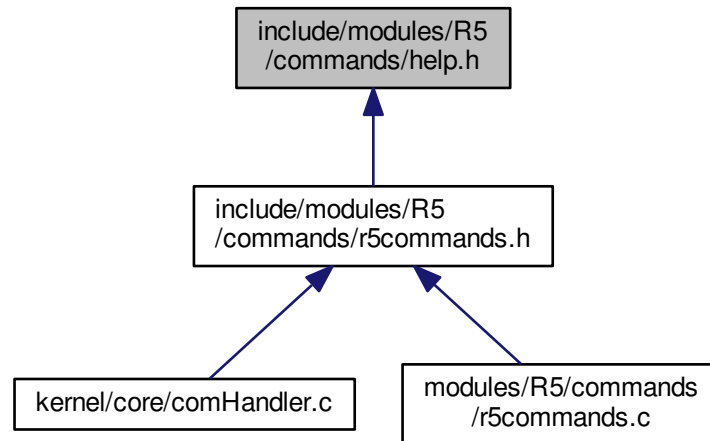
4.7.1.2 `#define` [HELP_R3_COMMAND_YIELD](#)

Value:

```
((const char*) \
    "Yields command handler to allow other processes to run.\n"\
    "\n"\
    "Usage: yield\n"\
    "\n"\
    "Args:\n"\
    "    [no args] - yields command handler\n")
```

4.8 include/modules/R5/commands/help.h File Reference

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



Macros

- `#define HELP_R5_COMMAND_SHOWMEMORY`

4.8.1 Macro Definition Documentation

4.8.1.1 `#define HELP_R5_COMMAND_SHOWMEMORY`

Value:

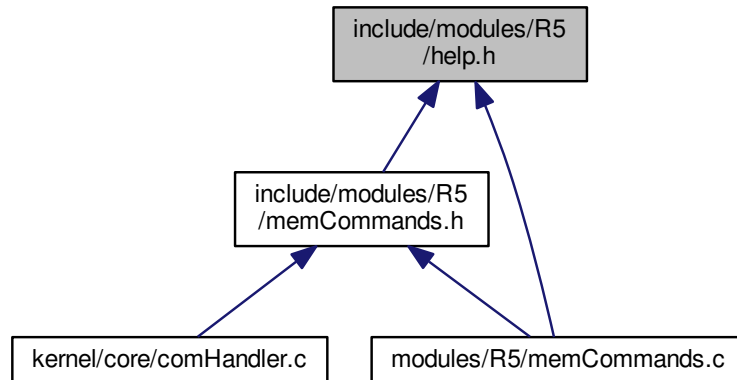
```

((const char*) \
  "Displays the following information for the specified CMCB's:\n"\
  "CMCB Type:\n"\
  "Beginning Memory Address:\n"\
  "Block Size:\n"\
  "Memory Size:\n"\
  "Process Name:\n"\
  "\n"\
  "Usage: showMemory [--all] [--free] [--allocated]\n"\
  "\n"\
  "Args:\n"\
  "  [no args] - Shows the help for this command\n"\
  "  --all - Displays both free and allocated memory\n"\
  "  --free - Displays free memory\n"\
  "  --allocated - Displays allocated memory")

```

4.9 include/modules/R5/help.h File Reference

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



Macros

- `#define HELP_R5_COMMAND_HEAP`
- `#define HELP_R5_COMMAND_ALLOC`
- `#define HELP_R5_COMMAND_FREE`
- `#define HELP_R5_COMMAND_EMPTY`

4.9.1 Macro Definition Documentation

4.9.1.1 `#define HELP_R5_COMMAND_ALLOC`

Value:

```

((const char*) \
  "Allocates memory block if memory is available\n"\
  "\n"\
  "Usage: allocMem size\n"\
  "\n"\
  "Args:\n"\
  "    size - The size of the memory in bytes")

```

4.9.1.2 `#define HELP_R5_COMMAND_EMPTY`

Value:

```

((const char*) \
  "Checks if memory is empty\n"\
  "\n"\
  "Usage: isEmpty \n"\
  "\n"\
  "Args:\n"\
  "    ")

```

4.9.1.3 #define HELP_R5_COMMAND_FREE

Value:

```
((const char*) \
  "Free up memory the memory at the given address\n"\
  "\n"\
  "Usage: freeMem address\n"\
  "\n"\
  "Args:\n"\
  "    address - the integer address of the memory")
```

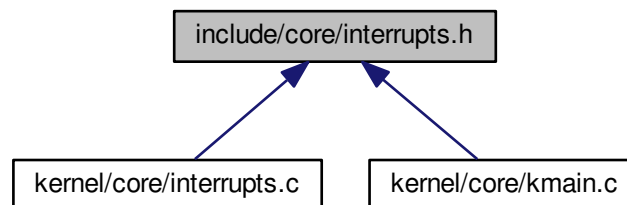
4.9.1.4 #define HELP_R5_COMMAND_HEAP

Value:

```
((const char*) \
  "Initializes the heap\n"\
  "\n"\
  "Usage: initHeap size\n"\
  "\n"\
  "Args:\n"\
  "    size - The size of the heap in bytes ")
```

4.10 include/core/interrupts.h File Reference

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



Functions

- void [init_irq](#) (void)
- void [init_pic](#) (void)

4.10.1 Function Documentation

4.10.1.1 void init_irq(void)

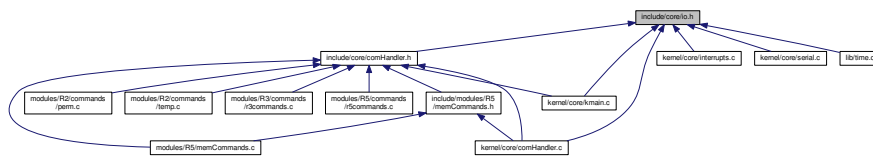
Installs the initial interrupt handlers for the first 32 irq lines. Most do a panic for now.

4.10.1.2 void init_pic (void)

Initializes the programmable interrupt controllers and performs the necessary remapping of IRQs. Leaves interrupts turned off.

4.11 include/core/io.h File Reference

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



Macros

- `#define outb(port, data) asm volatile ("outb %%a1,%%dx" : : "a" (data), "d" (port))`
- `#define inb(port)`

4.11.1 Macro Definition Documentation

4.11.1.1 #define inb(port)

Value:

```

({
    unsigned char r;
    asm volatile ("inb %%dx,%%a1": "=a" (r): "d" (port));
    r;
})

```

Reads a byte of data from a port.

Parameters

<i>port</i>	The port to read the data from
-------------	--------------------------------

Returns

The byte from the port

4.11.1.2 #define outb(port, data) asm volatile ("outb %%a1,%%dx" : : "a" (data), "d" (port))

Writes a byte of data to a port.

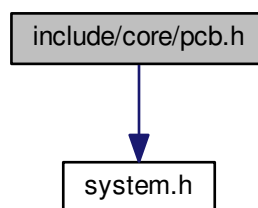
Parameters

<i>port</i>	The port to write the data to
<i>data</i>	The byte to write

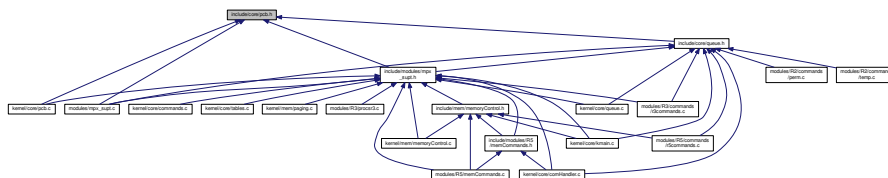
4.12 include/core/pcb.h File Reference

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

Include dependency graph for pcb.h:



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



Data Structures

- struct pcb

Macros

- #define BLOCKED 0
- #define READY 1
- #define RUNNING 2
- #define SYSTEM 0
- #define APPLICATION 1

Typedefs

- typedef struct pcb pcb

Functions

- `pcb * allocatePCB ()`
- `int freePCB (pcb *pcbPtr)`
- `pcb * setupPCB (const char *processName, int processClass, int priority)`
- `int checkParamName (const char *processName)`
- `int checkParamClass (int processClass)`
- `int checkParamPriority (int priority)`

4.12.1 Macro Definition Documentation

4.12.1.1 `#define APPLICATION 1`

4.12.1.2 `#define BLOCKED 0`

4.12.1.3 `#define READY 1`

4.12.1.4 `#define RUNNING 2`

4.12.1.5 `#define SYSTEM 0`

4.12.2 Typedef Documentation

4.12.2.1 `typedef struct pcb pcb`

4.12.3 Function Documentation

4.12.3.1 `pcb* allocatePCB ()`

Allocates memory for a new PCB and returns a pointer to it

Returns

PCB pointer or Null if error occurs

Allocates memory for a new PCB and returns a pointer to it

`freePCB` should be used when done using the `pcb` to free the memory in use

Returns

PCB pointer or Null if error occurs

4.12.3.2 `int checkParamClass (int processClass)`

Validates that the `processClass` is valid

Parameters

<i>processClass</i>	- int
---------------------	-------

Returns

integer 0 or 1 if valid

4.12.3.3 int checkParamName (const char * *processName*)

Validates that the processName is valid

Parameters

<i>processName</i>	- const char * processName
--------------------	----------------------------

Returns

integer 0 or 1 if valid

4.12.3.4 int checkParamPriority (int *priority*)

Validates that the priority is valid

Parameters

<i>priority</i>	- int
-----------------	-------

Returns

integer 0 or 1 if valid

4.12.3.5 int freePCB (pcb * *pcbPtr*)

Frees memory that is allocated for the pcb provided

Parameters

<i>pcbPtr</i>	pointer to pcb to be freed
---------------	----------------------------

Returns

integer code - 1 if successful, -1 otherwise

Frees memory that is allocated for the pcb provided

Parameters

<i>pcbPtr</i>	pointer to pcb to be freed
---------------	----------------------------

Returns

integer code - 1 if successful, 0 otherwise

4.12.3.6 `pcb* setupPCB (const char * processName, int processClass, int priority)`

Allocates memory for a new PCB and sets it with given params

Parameters

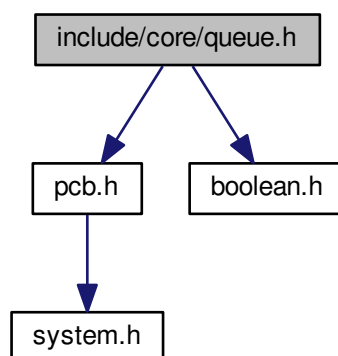
<i>processName</i>	- const string name
<i>processClass</i>	- integer identifying as system or application process (0, 1)
<i>priority</i>	- integer between 0 and 9 indicating priority

Returns

PCB pointer to new pcb or NULL if there were errors

4.13 `include/core/queue.h` File Reference

```
#include "pcb.h"  
#include "boolean.h"  
Include dependency graph for queue.h:
```



Returns

A pointer to the PCB, or null if not found

4.13.2.2 node* getBlockedQueue ()

Gets the head node of the blocked queue.

Returns

The head node of the blocked queue

4.13.2.3 node* getReadyQueue ()

Gets the head node of the ready queue.

Returns

The head node of the ready queue

4.13.2.4 node* getSuspendedBlockedQueue ()

Gets the head node of the suspended-blocked queue.

Returns

The head node of the suspended-blocked queue

4.13.2.5 node* getSuspendedReadyQueue ()

Gets the head node of the suspended-ready queue.

Returns

The head node of the suspended-ready queue

4.13.2.6 boolean insertPCB (pcb * p)

Inserts the PCB into the appropriate queue.

Parameters

<i>p</i>	The PCB to insert.
----------	--------------------

Returns

true if the PCB was inserted, false otherwise

4.13.2.7 `pcb* popBlocked ()`

Pops the next node off of the blocked queue.

Returns

The next node of the blocked queue, or NULL if it is empty

4.13.2.8 `pcb* popReady ()`

Pops the next node off of the ready queue.

Returns

The next node of the ready queue, or NULL if it is empty

4.13.2.9 `pcb* popSuspendedBlocked ()`

Pops the next node off of the suspended-blocked queue.

Returns

The next node of the suspended-blocked queue, or NULL if it is empty

4.13.2.10 `pcb* popSuspendedReady ()`

Pops the next node off of the suspended-ready queue.

Returns

The next node of the suspended-ready queue, or NULL if it is empty

4.13.2.11 `boolean removePCB (pcb * p)`

Removes the given PCB from it's queue.

Parameters

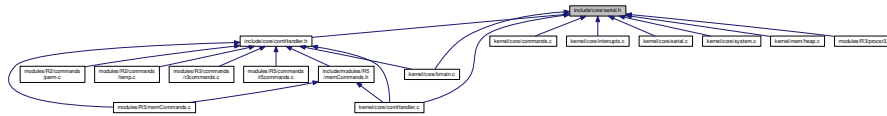
<i>p</i>	The PCB to remove
----------	-------------------

Returns

true if the PCB was removed, false otherwise

4.14 include/core/serial.h File Reference

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



Macros

- `#define COM1 0x3f8`
- `#define COM2 0x2f8`
- `#define COM3 0x3e8`
- `#define COM4 0x2e8`

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)`

4.14.1 Macro Definition Documentation

4.14.1.1 `#define COM1 0x3f8`

4.14.1.2 `#define COM2 0x2f8`

4.14.1.3 `#define COM3 0x3e8`

4.14.1.4 `#define COM4 0x2e8`

4.14.2 Function Documentation

4.14.2.1 `int init_serial (int device)`

Initializes devices for user interaction, logging, ...

Parameters

<i>device</i>	The device to initialize
---------------	--------------------------

Returns

The error code

4.14.2.2 int serial_print (const char * *msg*)

Writes a message to the active serial output device.

Parameters

<i>msg</i>	The message to write
------------	----------------------

Returns

The error code

4.14.2.3 int serial_println (const char * *msg*)

Writes a message to the active serial output device. Appends a newline character.

Parameters

<i>msg</i>	The message to write
------------	----------------------

Returns

The error code

4.14.2.4 int set_serial_in (int *device*)

Sets serial_port_in to the given device address. All serial input, such as console input via a virtual machine, QE↔MU/Bohc/etc, will be directed to the device.

Parameters

<i>device</i>	The divce to set as input
---------------	---------------------------

Returns

The error code

4.14.2.5 int set_serial_out (int device)

Sets serial_port_out to the given device address. All serial output, such as that from serial_println, will be directed to this device.

Parameters

<i>device</i>	The device to set as output
---------------	-----------------------------

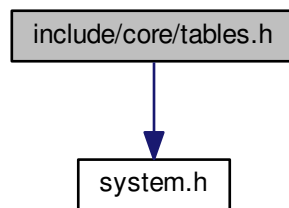
Returns

The error code

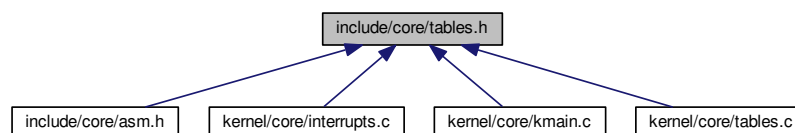
4.15 include/core/tables.h File Reference

```
#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](#)
- struct [idt_struct](#)
- struct [gdt_descriptor_struct](#)
- struct [gdt_entry_struct](#)

Functions

- struct `idt_entry_struct __attribute__((packed)) idt_entry`
- 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 ()`

Variables

- `u16int base_low`
- `u16int sselect`
- `u8int zero`
- `u8int flags`
- `u16int base_high`
- `u16int limit`
- `u32int base`
- `u16int limit_low`
- `u8int base_mid`
- `u8int access`

4.15.1 Function Documentation

4.15.1.1 `struct idt_entry_struct __attribute__((packed)) ((packed))`

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

Installs a new table entry into the global descriptor table.

Parameters

<i>idx</i>	
<i>base</i>	
<i>limit</i>	
<i>access</i>	
<i>flags</i>	

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

Installs a new gate entry into the IDT.

Parameters

<i>idx</i>	
<i>base</i>	
<i>sel</i>	
<i>flags</i>	

4.15.1.4 void init_gdt ()

Creates the global descriptor table and installs it using the defined assembly routine.

4.15.1.5 void init_idt ()

Creates the interrupt descriptor table and writes the pointer using the defined assembly function.

4.15.2 Variable Documentation

4.15.2.1 u8int access

4.15.2.2 u32int base

4.15.2.3 u8int base_high

4.15.2.4 u16int base_low

4.15.2.5 u8int base_mid

4.15.2.6 u8int flags

4.15.2.7 u16int limit

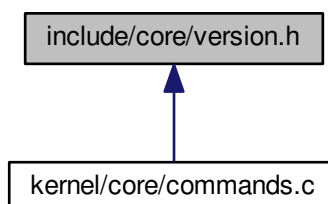
4.15.2.8 u16int limit_low

4.15.2.9 u16int sselect

4.15.2.10 u8int zero

4.16 include/core/version.h File Reference

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



Macros

- `#define OS_VERSION ((const char*) "NPE-MPX.R5.04282017")`

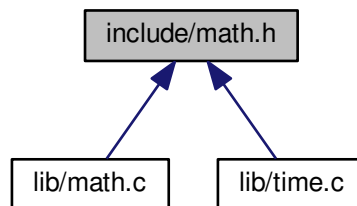
4.16.1 Macro Definition Documentation

4.16.1.1 `#define OS_VERSION ((const char*) "NPE-MPX.R5.04282017")`

The current OS version.

4.17 include/math.h File Reference

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



Functions

- unsigned char `bcdToDec` (unsigned char *bcd*)
- unsigned char `decToBcd` (unsigned char *dec*)

4.17.1 Function Documentation

4.17.1.1 unsigned char `bcdToDec` (unsigned char *bcd*)

Converts a BCD encoded byte to a decimal encoded byte

Parameters

<i>bcd</i>	The value to convert
------------	----------------------

Returns

The decimal value

4.17.1.2 unsigned char decToBcd (unsigned char *dec*)

Converts a decimal encoded byte to a BCD encoded byte

Parameters

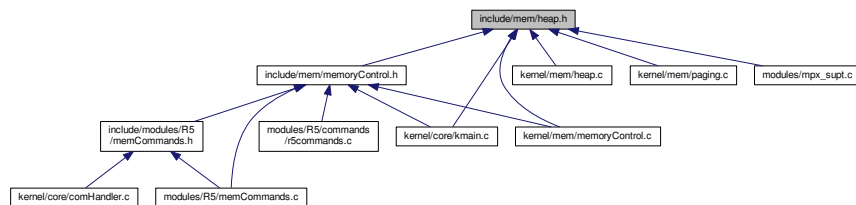
<i>dec</i>	The value to convert
------------	----------------------

Returns

The BCD value

4.18 include/mem/heap.h File Reference

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



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)

Variables

- typedef [__attribute__](#)

4.18.1 Macro Definition Documentation

4.18.1.1 `#define KHEAP_BASE 0xD000000`

4.18.1.2 `#define KHEAP_MIN 0x10000`

4.18.1.3 `#define KHEAP_SIZE 0x1000000`

4.18.1.4 `#define TABLE_SIZE 0x1000`

Kernel heap.

4.18.2 Function Documentation

4.18.2.1 `u32int _kmalloc (u32int size, int page_align, u32int * phys_addr)`

Base-level kernel memory allocation routine. Used to provide page alignment and access physical addresses of allocations. Called by `kmalloc` with `align=0`, `physical_address=0`.

Parameters

<i>size</i>	The amount of memory to allocate
<i>align</i>	The page alignment
<i>phys_addr</i>	The physical address

Returns

The memory address

4.18.2.2 `u32int alloc (u32int size, heap * h, int align)`

Allocates some memory using the given heap. Can specify page-alignment.

Parameters

<i>size</i>	The amount of memory to allocate
<i>hp</i>	The heap to allocate on
<i>align</i>	The page alignment

Returns

The memory address

4.18.2.3 void init_kheap ()

Initialize the kernel heap, and set it as the current heap.

4.18.2.4 u32int kfree ()

Free kernel memory.

Returns**4.18.2.5 u32int kmalloc (u32int size)**

Standard memory allocation routine. Use this unless you need to specify alignment or obtain a physical address. Calls `_kmalloc`.

Parameters

<i>size</i>	The amount of memory to allocate
-------------	----------------------------------

Returns

The memory address

4.18.2.6 heap* make_heap (u32int base, u32int max, u32int min)

Create a new heap.

Parameters

<i>base</i>	Physical start address of the heap
<i>max</i>	Maximum size the heap may grow to
<i>min</i>	Minium/Initial size

Returns

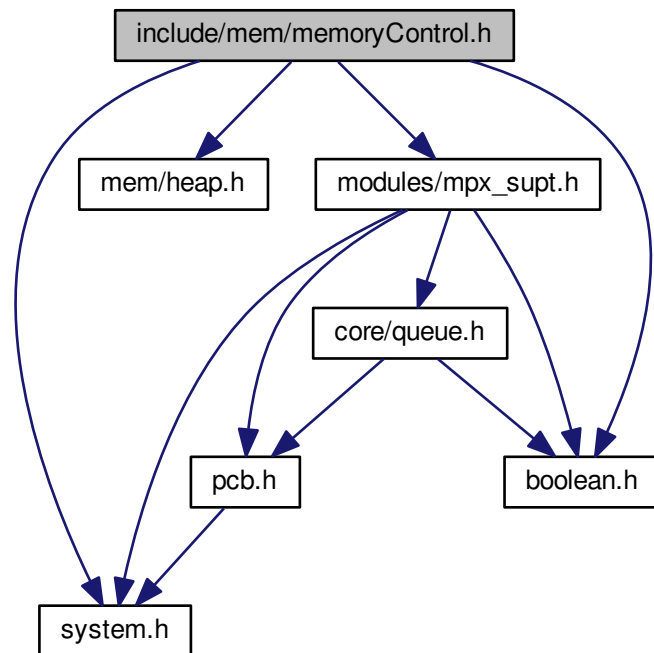
The address of the heap

4.18.3 Variable Documentation

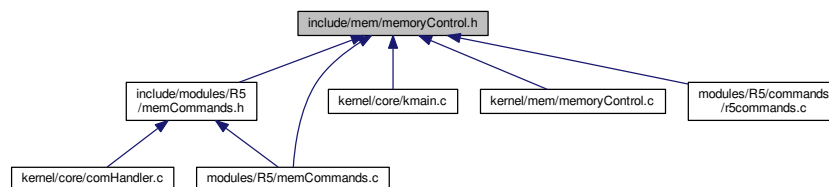
4.18.3.1 struct gdt_entry_struct __attribute__

4.19 include/mem/memoryControl.h File Reference

```
#include <system.h>
#include <mem/heap.h>
#include <modules/mpx_supt.h>
#include <boolean.h>
Include dependency graph for memoryControl.h:
```



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



Data Structures

- struct [cmcb](#)
- struct [lmcb](#)

Macros

- `#define FREE 0`
- `#define ALLOCATED 1`

Typedefs

- `typedef struct cmcb cmcb`
- `typedef struct lmcb lmcb`

Functions

- `boolean initializeHeap (int size)`
- `void * allocateMemory (int size)`
- `boolean deallocateMemory (void *memPointer)`
- `boolean isEmpty ()`
- `cmcb * getFreeHead ()`
- `cmcb * getAllocatedHead ()`

4.19.1 Macro Definition Documentation

4.19.1.1 `#define ALLOCATED 1`

4.19.1.2 `#define FREE 0`

4.19.2 Typedef Documentation

4.19.2.1 `typedef struct cmcb cmcb`

4.19.2.2 `typedef struct lmcb lmcb`

4.19.3 Function Documentation

4.19.3.1 `void* allocateMemory (int size)`

Allocates a memory block if enough memory is available

Parameters

<i>size</i>	- size of memory to allocate in bytes
-------------	---------------------------------------

Returns

pointer to the me

4.19.3.2 **boolean** deallocateMemory (void * *memPointer*)

Deallocates the block of memory at the mempointer

Parameters

<i>memPointer</i>	- pointer to the mem block
-------------------	----------------------------

Returns

boolean - tells whether successfull dealloc

Deallocates the block of memory at the mempointer

Parameters

<i>memPointer</i>	- pointer to the mem block
-------------------	----------------------------

Returns

boolean - boolean telling whether succesful dealloc

4.19.3.3 **cmcb*** getAllocatedHead ()

Returns the head to the allocated list

Returns

cmcb * to the allocated list head

4.19.3.4 **cmcb*** getFreeHead ()

Returns the head of the free list

Returns

cmcb * to the free list head

4.19.3.5 **boolean** initializeHeap (int *size*)

Initializes the heap to the provided size and creates a free mem block across it

Parameters

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

Returns

boolean - boolean denoting if heap was initialized

4.19.3.6 boolean isEmpty ()

Returns a boolean telling if all the memory is empty

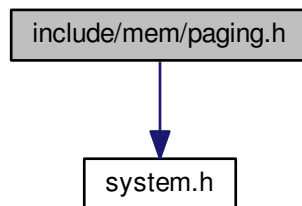
Returns

boolean

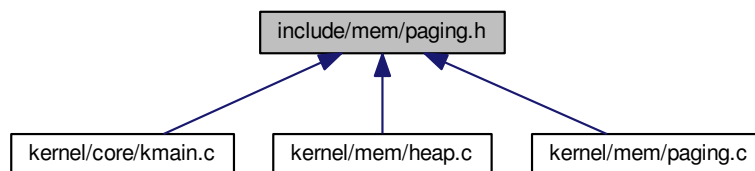
4.20 include/mem/paging.h File Reference

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

Include dependency graph for paging.h:



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

**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)

4.20.1 Macro Definition Documentation

4.20.1.1 #define PAGE_SIZE 0x1000

4.20.2 Function Documentation

4.20.2.1 void clear_bit (u32int *addr*)

Marks a page frame bit as free (0).

Parameters

<i>addr</i>	The address of the frame
-------------	--------------------------

4.20.2.2 u32int first_free ()

Finds the first free page frame.

Returns

The first free page frame

4.20.2.3 u32int get_bit (u32int *addr*)

Checks if page frame is in use.

Parameters

<i>addr</i>	The address of the frame
-------------	--------------------------

Returns

True if it is in use

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

Finds and returns a page, allocating a new page table if necessary.

Parameters

<i>addr</i>	The address of the page
<i>dir</i>	The page directory
<i>make_table</i>	Boolean to create a table if necessary

Returns

A pointer to the page

4.20.2.5 void init_paging ()

Initializes the kernel page directory and initial kernel heap area. Performs identity mapping of the kernel frames such that the virtual addresses are equivalent to the physical addresses.

4.20.2.6 void load_page_dir (page_dir * new_dir)

Sets a page directory as the current directory and enables paging via the CR0 register, The CR3 register enables address translation from linear to physical address.

http://en.wikipedia.org/wiki/Control_register#Control_registers_in_x86_↵series

Parameters

<i>new_page_dir</i>	The page directory to set as the current
---------------------	------------------------------------------

Sets a page directory as the current directory and enables paging via the CR0 register, The CR3 register enables address translation from linear to physical address.

http://en.wikipedia.org/wiki/Control_register#Control_registers_in_x86_↵series

Parameters

<i>new_page_dir</i>	The page directory to set as the current
---------------------	------------------------------------------

4.20.2.7 void new_frame (page_entry * page)

Marks a frame as in use un the frame bitmap, sets up the page, and saves* the frame index in the page.*

Parameters

<i>page</i>	The page to create the frame in
-------------	---------------------------------

Marks a frame as in use un the frame bitmap, sets up the page, and saves* the frame index in the page.

Parameters

<i>page</i>	The page to create the frame in
-------------	---------------------------------

4.20.2.8 void set_bit (u32int addr)

Marks a page frame bit as in use (1).

Parameters

<i>addr</i>	The address of the frame
-------------	--------------------------

4.21 include/modules/mpx_supt.h File Reference

```
#include <core/queue.h>
#include <core/pcb.h>
#include <boolean.h>
#include <system.h>
```


Typedefs

- typedef struct [context](#) [context](#)

Functions

- [u32int](#) * [sys_call](#) ([context](#) *registers)
- void * [memset](#) (void *s, int c, [size_t](#) n)
- int [sys_req](#) (int op_code)
- void [mpx_init](#) (int cur_mod)
- void [sys_set_malloc](#) (void *(*func)(int))
- void [sys_set_free](#) ([boolean](#)(func)(void *))
- void * [sys_alloc_mem](#) ([u32int](#) size)
- int [sys_free_mem](#) (void *ptr)
- void [idle](#) ()
- const char * [getCOPName](#) ()

4.21.1 Macro Definition Documentation

4.21.1.1 `#define EXIT 0`

4.21.1.2 `#define IDLE 1`

4.21.1.3 `#define MODULE_R1 0`

4.21.1.4 `#define MODULE_R2 1`

4.21.1.5 `#define MODULE_R3 2`

4.21.1.6 `#define MODULE_R4 4`

4.21.1.7 `#define MODULE_R5 8`

4.21.1.8 `#define READ 2`

4.21.1.9 `#define WRITE 3`

4.21.2 Typedef Documentation

4.21.2.1 typedef struct [context](#) [context](#)

4.21.3 Function Documentation

4.21.3.1 `const char* getCOPName ()`

Gets the name of the COP

Returns

const char pointer name

4.21.3.2 `void idle ()`

The idle process

4.21.3.3 `void* memset (void * s, int c, size_t n)`

Set a region of memory

Parameters

<i>s</i>	Destination
<i>c</i>	Byte to write
<i>n</i>	Count

Returns

s

4.21.3.4 void mpx_init (int *cur_mod*)

Initialize MPX support software

Parameters

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

4.21.3.5 void* sys_alloc_mem (u32int *size*)

Allocates a block of memory (similar to malloc)

Parameters

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

Returns

The allocated memory

4.21.3.6 u32int* sys_call (context * *registers*)

Changes the currently running process to that of the next ready process

Parameters

<i>registers</i>	- copy of register values
------------------	---------------------------

Returns

u32int position of stackTop

4.21.3.7 int sys_free_mem (void * *ptr*)

Frees memory

Parameters

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

Returns**4.21.3.8 int sys_req (int *op_code*)**

Generates interrupt 60H

Parameters

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

Returns

0

4.21.3.9 void sys_set_free (boolean(func)(void *))

Sets the memory free function for sys_free_mem

Parameters

<i>func</i>	Function pointer to the memory free-er
-------------	----------------------------------------

4.21.3.10 void sys_set_malloc (void (*)(int) *func*)

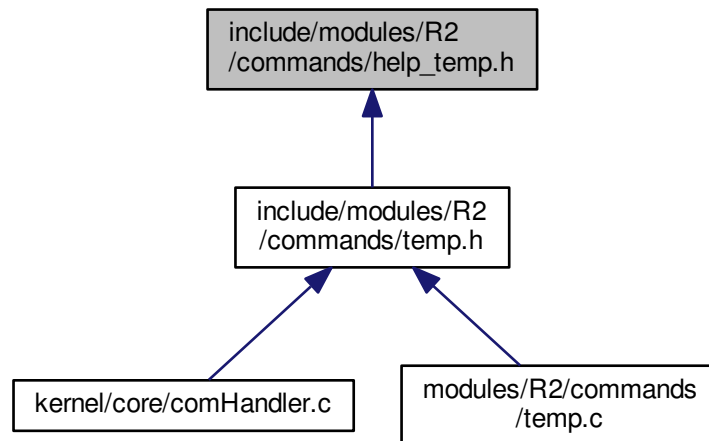
Sets the memory allocation function for sys_alloc_mem

Parameters

<i>func</i>	Function pointer to the memory allocator
-------------	------------------------------------------

4.22 include/modules/R2/commands/help_temp.h File Reference

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



Macros

- `#define HELP_R2_COMMAND_CPCB`
- `#define HELP_R2_COMMAND_DPCB`
- `#define HELP_R2_COMMAND_BPCB`
- `#define HELP_R2_COMMAND_UPCB`

4.22.1 Macro Definition Documentation

4.22.1.1 `#define HELP_R2_COMMAND_BPCB`

Value:

```

((const char*) \
    "Places a PCB into the blocked state and reinserts it into the appropriate queue.\n" \
    "\n" \
    "Note: This command will be removed in module R3/R4\n" \
    "\n" \
    "Usage: bpcb name\n" \
    "\n" \
    "Args:\n" \
    "    name - The Process Name to place into the blocked state (must exist)")

```

4.22.1.2 #define HELP_R2_COMMAND_CPCB

Value:

```
((const char*) \
    "Creates a PCB and inserts it into the appropriate queue.\n" \
    "\n" \
    "Note: This command will be removed in module R3/R4\n" \
    "\n" \
    "Usage: cpcb name class priority\n" \
    "\n" \
    "Args:\n" \
    "    name - The Process Name (must be unique)\n" \
    "    class - The Process Class (either 0 (system) or 1 (application))\n" \
    "    priority - The Process Priority (number between 0 and 9)")
```

4.22.1.3 #define HELP_R2_COMMAND_DPCB

Value:

```
((const char*) \
    "Removes a PCB from the appropriate queue and then frees all associated memory.\n" \
    "\n" \
    "Note: This command will be removed in module R3/R4\n" \
    "\n" \
    "Usage: dpcb name\n" \
    "\n" \
    "Args:\n" \
    "    name - The Process Name to remove (must exist)")
```

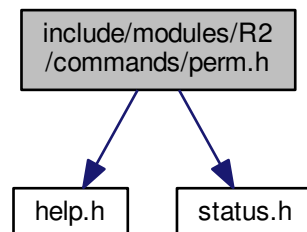
4.22.1.4 #define HELP_R2_COMMAND_UPCB

Value:

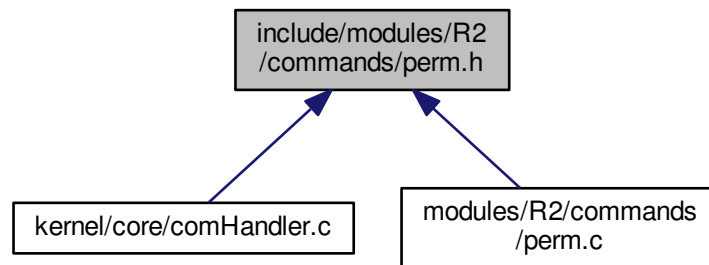
```
((const char*) \
    "Places a PCB into the unblocked state and reinserts it into the appropriate queue.\n" \
    "\n" \
    "Note: This command will be removed in module R3/R4\n" \
    "\n" \
    "Usage: upcb name\n" \
    "\n" \
    "Args:\n" \
    "    name - The Process Name to place into the unblocked state (must exist)")
```

4.23 include/modules/R2/commands/perm.h File Reference

```
#include "help.h"
#include "status.h"
Include dependency graph for perm.h:
```



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



Functions

- void [registerR2PermCommands](#) ()
- const char * [suspendPcb](#) (char **args, int numArgs)
- const char * [resumePcb](#) (char **args, int numArgs)
- const char * [setPriorityPcb](#) (char **args, int numArgs)
- const char * [showPcbInfo](#) (char **args, int numArgs)

4.23.1 Function Documentation

4.23.1.1 void registerR2PermCommands ()

Registers the permanent commands in the command handler

4.23.1.2 const char* resumePcb (char ** args, int numArgs)

Places a PCB into the not suspended state and reinserts it into the appropriate queue.

Usage: rpcb name

Args: name - The name of the process to resume (must exist)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

Places a PCB into the not suspended state and reinserts it into the appropriate queue.

Usage: rpcb name

Args: name - The name of the process to resume (must exist) -all - Resumes all processes

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.23.1.3 `const char* setPriorityPcb (char ** args, int numArgs)`

Sets a PCB's priority and reinserts the process into the correct place in the correct queue.

Usage: ppcb name priority

Args: name - The name of the process to set the priority on (must exist) priority - The new priority (between 0 and 9)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.23.1.4 `const char* showPcbInfo (char ** args, int numArgs)`

Displays the following information for the specified PCBs: Process Name: Class: State: Suspended Status↵: Priority:

Usage: showpcb [-all] [-ready] [-blocked] [-name pcbName] (at least 1 must be specified)

Args: [no args] - Shows the help for this command -all - Displays information for all PCBs -ready - Displays information for ready PCBs -blocked - Displays information for blocked PCBs -suspended - Displays information for suspended PCBs -name - Displays information for the specified PCB (can be used multiple times)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

Displays the following information for the specified PCBs: Process Name: Class: State: Suspended Status↵
: Priority:

Usage: showpcb [-all] [-ready] [-blocked] [-suspended] [-name pcbName]

Args: [no args] - Shows the help for this command -all - Displays information for all PCBs -ready - Displays information for ready PCBs -blocked - Displays information for blocked PCBs -suspended - Displays information for suspended PCBs -name - Displays information for the specified PCB

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.23.1.5 `const char* suspendPcb (char ** args, int numArgs)`

Places a PCB into the suspended state and reinserts it into the appropriate queue.

Usage: spcb name

Args: name - The name of the process to suspend (must exist) -all - Resumes all processes

Parameters

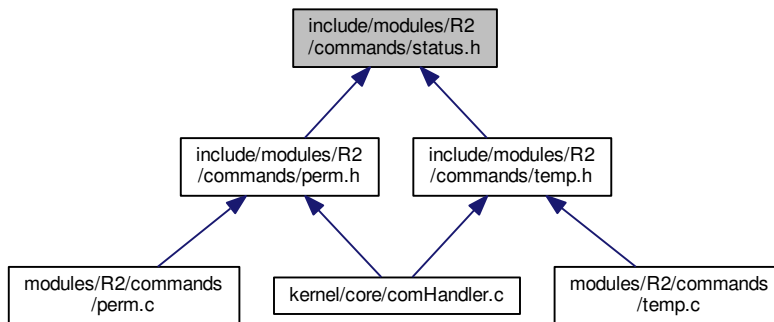
<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.24 include/modules/R2/commands/status.h File Reference

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



Macros

- `#define UNKNOWN_PCB_NAME ((const char*) "Unknown PCB name.")`
- `#define SUSPEND_PCB_SUCCESS ((const char*) "Process suspended.")`
- `#define RESUME_PCB_SUCCESS ((const char*) "Process resumed.")`
- `#define RESUME_PCBS_SUCCESS ((const char*) "All processes resumed.")`
- `#define UPDATE_PRIORITY_SUCCESS ((const char*) "Priority updated.")`

4.24.1 Macro Definition Documentation

4.24.1.1 `#define RESUME_PCB_SUCCESS ((const char*) "Process resumed.")`

4.24.1.2 `#define RESUME_PCBS_SUCCESS ((const char*) "All processes resumed.")`

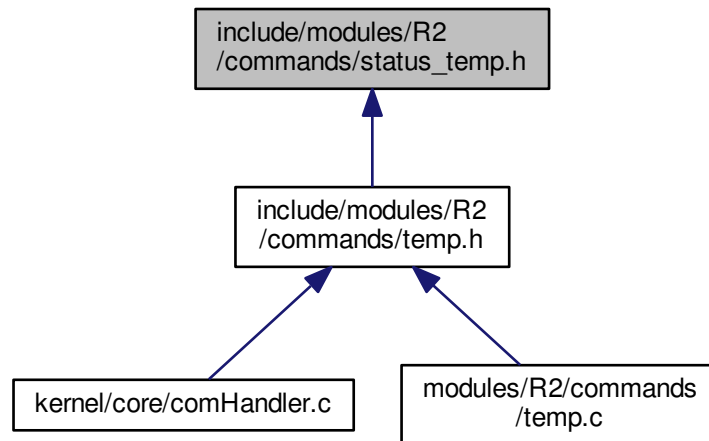
4.24.1.3 `#define SUSPEND_PCB_SUCCESS ((const char*) "Process suspended.")`

4.24.1.4 `#define UNKNOWN_PCB_NAME ((const char*) "Unknown PCB name.")`

4.24.1.5 `#define UPDATE_PRIORITY_SUCCESS ((const char*) "Priority updated.")`

4.25 include/modules/R2/commands/status_temp.h File Reference

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



Macros

- `#define CREATE_PCB_SUCCESS` ((const char*) "PCB created successfully.")
- `#define DELETE_PCB_SUCCESS` ((const char*) "PCB deleted successfully.")
- `#define BLOCK_PCB_SUCCESS` ((const char*) "PCB set to blocked.")
- `#define UNBLOCK_PCB_SUCCESS` ((const char*) "PCB set to unblocked.")
- `#define PROCESS_NAME_ALREADY_EXISTS` ((const char*) "This process name already exists")

4.25.1 Macro Definition Documentation

4.25.1.1 `#define BLOCK_PCB_SUCCESS` ((const char*) "PCB set to blocked.")

4.25.1.2 `#define CREATE_PCB_SUCCESS` ((const char*) "PCB created successfully.")

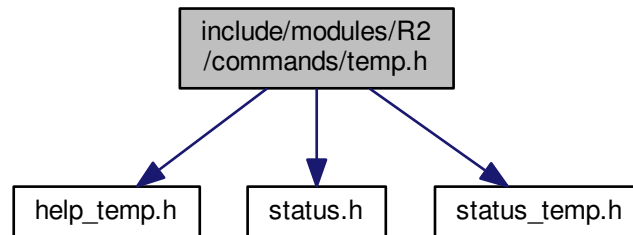
4.25.1.3 `#define DELETE_PCB_SUCCESS` ((const char*) "PCB deleted successfully.")

4.25.1.4 `#define PROCESS_NAME_ALREADY_EXISTS` ((const char*) "This process name already exists")

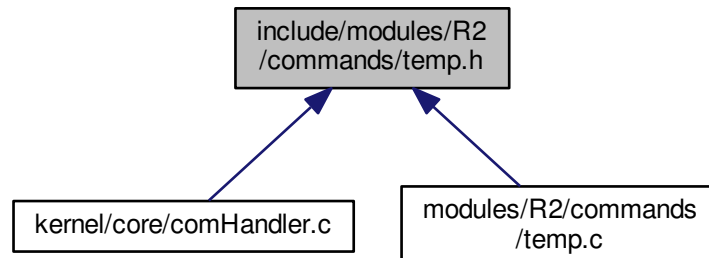
4.25.1.5 `#define UNBLOCK_PCB_SUCCESS` ((const char*) "PCB set to unblocked.")

4.26 include/modules/R2/commands/temp.h File Reference

```
#include "help_temp.h"
#include "status.h"
#include "status_temp.h"
Include dependency graph for temp.h:
```



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



Functions

- void [registerR2TempCommands](#) ()
- const char * [createPcb](#) (char **args, int numArgs)
- const char * [deletePcb](#) (char **args, int numArgs)
- const char * [blockPcb](#) (char **args, int numArgs)
- const char * [unblockPcb](#) (char **args, int numArgs)

4.26.1 Function Documentation

4.26.1.1 `const char* blockPcb (char ** args, int numArgs)`

Places a PCB into the blocked state and reinserts it into the appropriate queue.

Note: This command will be removed in module R3/R4

Usage: bpcb name

Args: name - The Process Name to place into the blocked state (must exist)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.26.1.2 `const char* createPcb (char ** args, int numArgs)`

Creates a PCB and inserts it into the appropriate queue.

Note: This command will be removed in module R3/R4

Usage: cpcb name class priority

Args: name - The Process Name (must be unique) class - The Process Class (either 0 (system) or 1 (application))
priority - The Process Priority (number between 0 and 9)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.26.1.3 `const char* deletePcb (char ** args, int numArgs)`

Removes a PCB from the appropriate queue and then frees all associated memory.

Note: This command will be removed in module R3/R4

Usage: dpcb name

Args: name - The Process Name to remove (must exist)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.26.1.4 void registerR2TempCommands ()

Registers the temporary commands in the command handler

4.26.1.5 const char* unblockPcb (char ** *args*, int *numArgs*)

Places a PCB into the unblocked state and reinserts it into the appropriate queue.

Note: This command will be removed in module R3/R4

Usage: upcb name

Args: name - The Process Name to place into the unblocked state (must exist)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

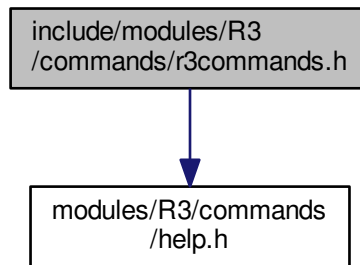
Returns

A status message indicating success/failure

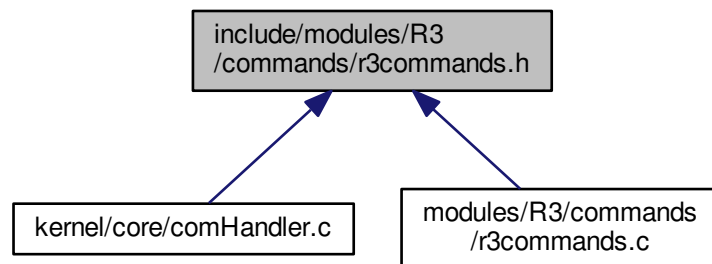
4.27 include/modules/R3/commands/r3commands.h File Reference

```
#include <modules/R3/commands/help.h>
```

Include dependency graph for r3commands.h:



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



Functions

- void `registerR3Commands` ()
- const char * `yield` (char **args, int numArgs)
- const char * `loadr3` (char **args, int numArgs)

4.27.1 Function Documentation

4.27.1.1 const char* loadr3 (char ** args, int numArgs)

Loads the r3 processes to the queue.
")

Usage: loadr3

Args: [no args] - loads processes

Parameters

<i>args</i>	The arguments to pass to the function
-------------	---------------------------------------

Returns

""

4.27.1.2 void registerR3Commands ()

Registers commands in command handler

4.27.1.3 const char* yield (char ** *args*, int *numArgs*)

Yields command handler to allow other processes to run.

Usage: yield

Args: [no args] - yields command handler

Parameters

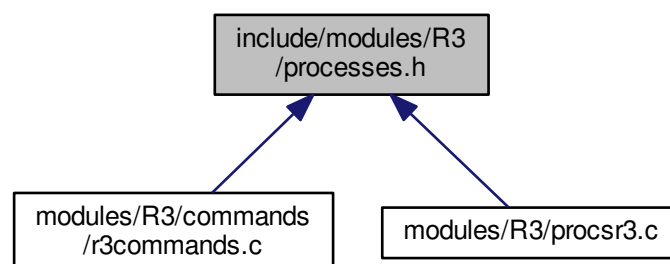
<i>args</i>	The arguments to pass to the function
-------------	---------------------------------------

Returns

""

4.28 include/modules/R3/processes.h File Reference

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



Functions

- void [proc1](#) ()
- void [proc2](#) ()
- void [proc3](#) ()
- void [proc4](#) ()
- void [proc5](#) ()

4.28.1 Function Documentation

4.28.1.1 void [proc1](#) ()

4.28.1.2 void [proc2](#) ()

4.28.1.3 void [proc3](#) ()

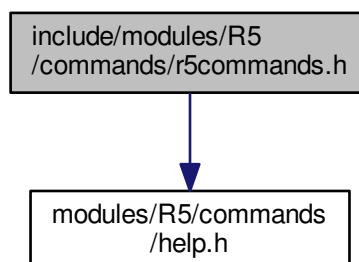
4.28.1.4 void [proc4](#) ()

4.28.1.5 void [proc5](#) ()

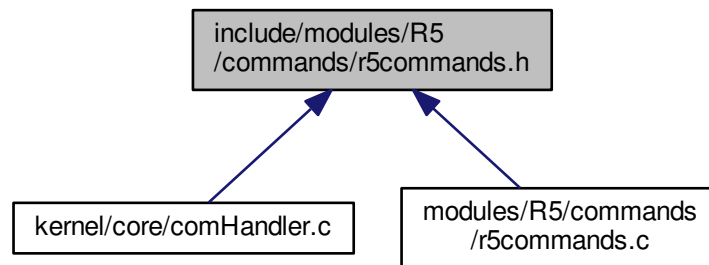
4.29 include/modules/R5/commands/r5commands.h File Reference

```
#include <modules/R5/commands/help.h>
```

Include dependency graph for r5commands.h:



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



Functions

- void [registerR5PermCommands](#) ()
- const char * [showMemory](#) (char **args, int numArgs)

4.29.1 Function Documentation

4.29.1.1 void registerR5PermCommands ()

Registers commands in command handler

Registers the permanent commands in the command handler

4.29.1.2 const char* showMemory (char ** args, int numArgs)

Displays the following information for the specified CMCB's: CMCB Type: Begining Memory Address: Block Size: Memory Size: Process Name:

Usage: showMemory [-all] [-free] [-allocated]

Args: [no args] - Shows the help for this command -all - Displays both free and allocated memory -free - Displays free memory -allocated - Displays allocated memory

Parameters

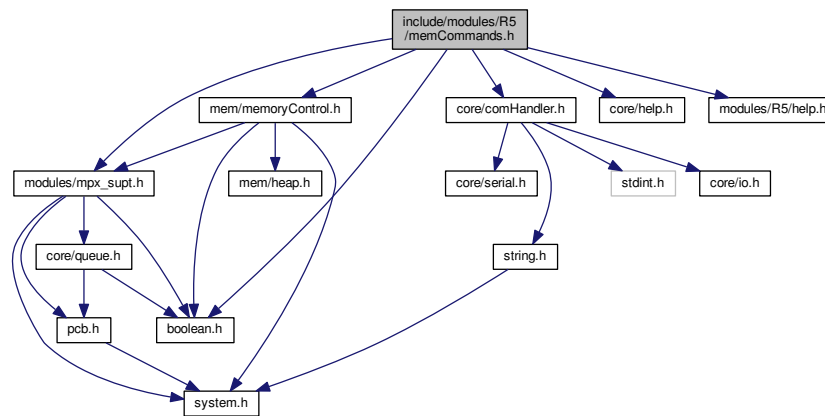
<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

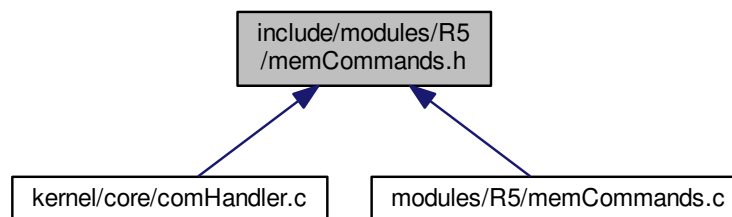
A status message indicating success/failure

4.30 include/modules/R5/memCommands.h File Reference

```
#include <mem/memoryControl.h>
#include <boolean.h>
#include <core/comHandler.h>
#include <core/help.h>
#include <modules/R5/help.h>
#include <modules/mpx_supt.h>
Include dependency graph for memCommands.h:
```



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



Functions

- void [registerR5TempCommands](#) ()
- const char * [initHeap](#) (char **args, int numArgs)
- const char * [allocateMem](#) (char **args, int numArgs)
- const char * [freeMemory](#) (char **args, int numArgs)
- const char * [isEmptyCom](#) (char **args, int numArgs)

4.30.1 Function Documentation

4.30.1.1 `const char* allocateMem (char ** args, int numArgs)`

Allocates a memory block if enough memory is available

Parameters

<i>size</i>	- size of memory to allocate in bytes
-------------	---------------------------------------

Returns

pointer to the me

Allocates a memory block if enough memory is available

Parameters

<i>size</i>	- size of memory to allocate in bytes
-------------	---------------------------------------

Returns

pointer to the me

4.30.1.2 `const char* freeMemory (char ** args, int numArgs)`

Deallocates the block of memory at the mempointer

Parameters

<i>memPointer</i>	- pointer to the mem block
-------------------	----------------------------

4.30.1.3 `const char* initHeap (char ** args, int numArgs)`

Initializes the heap to the provided size and creates a free mem block across it

Returns

True or false

4.30.1.4 `const char* isEmptyCom (char ** args, int numArgs)`

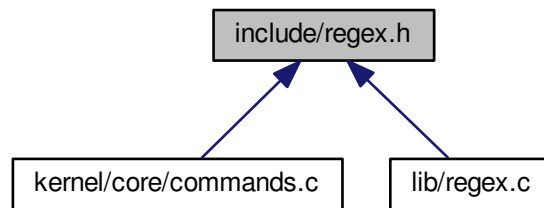
Check if memory is empty

4.30.1.5 void registerR5TempCommands ()

Registers the permanent commands in the command handler

4.31 include/regex.h File Reference

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



Functions

- int [testRegex](#) (const char *regex, const char *stringToCheck)

4.31.1 Function Documentation

4.31.1.1 int testRegex (const char * *regex*, const char * *stringToCheck*)

Tests if the stringToCheck adheres to the given regex string

The regex string is comprised of: d matches digits 0-9 c matches characters a-zA-Z u matches uppercase character, A-Z l matches lowercase character, a-z

- matches any char /char for a literal character, ex: "/a" matches 'a', /d matches 'd'

Example : regex "dcd/a" matches any string with the pattern "digit character digit 'a'", ex "1b3a", "6b9a"

Parameters

<i>regex</i>	
<i>stringToCheck</i>	

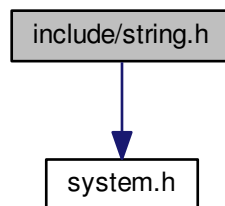
Returns

1 if adheres, 0 otherwise

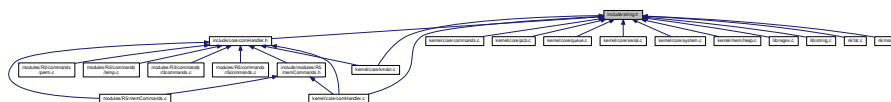
4.32 include/string.h File Reference

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

Include dependency graph for string.h:



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



Functions

- int [isspace](#) (const char *c)
- int [isdigit](#) (const char c)
- int [isChar](#) (const char c)
- int [isUpperChar](#) (const char c)
- int [isLowerChar](#) (const char c)
- char * [strcpy](#) (char *s1, const char *s2)
- char * [strcat](#) (char *s1, const char *s2)
- int [strlen](#) (const char *s)
- int [strcmp](#) (const char *s1, const char *s2)
- char * [strtok](#) (char *s1, const char *s2)
- int [atoi](#) (const char *s)
- void [itoa](#) (int num, char *str, int [base](#))
- void [reverse](#) (char *str, int len)

4.32.1 Function Documentation

4.32.1.1 int atoi (const char * s)

Convert an ASCII string to an integer

Parameters

<i>s</i>	The string to convert
----------	-----------------------

Returns

The integer value of the string, or the MAX/MIN value of an integer if the value is out of range.

4.32.1.2 int isChar (const char *c*)

Checks if the given char is a-z or A-Z

Parameters

<i>const</i>	char <i>c</i>
--------------	---------------

Returns

1 if *c* is a char, 0 otherwise

Checks if the given char is a-z or A-Z

Parameters

<i>const</i>	char <i>c</i>
--------------	---------------

Returns

1 if *c* is a char, 0 otherwise

4.32.1.3 int isdigit (const char *c*)

Determine if a character is a digit.

Parameters

<i>c</i>	The character to check
----------	------------------------

Returns

True if the character is a digit

4.32.1.4 int isLowerChar (const char *c*)

Checks if the given char is a-z

Parameters

<i>const</i>	char c
--------------	--------

Returns

1 if c is a lower char, 0 otherwise

4.32.1.5 int isspace (const char * c)

Determine if a character is whitespace.

Parameters

c	The character to check
---	------------------------

Returns

True if the character is a whitespace character

4.32.1.6 int isUpperChar (const char c)

Checks if the given char is A-Z

Parameters

<i>const</i>	char c
--------------	--------

Returns

1 if c is a upper char, 0 otherwise

4.32.1.7 void itoa (int num, char * str, int base)

Converts an integer to an ASCII string.

Parameters

<i>num</i>	The number to convert
<i>str</i>	The destination string
<i>base</i>	The radix

4.32.1.8 void reverse (char * *str*, int *len*)

Reverses a string.

Parameters

<i>str</i>	The string to reverse
<i>len</i>	The length of the string

4.32.1.9 char* strcat (char * *s1*, const char * *s2*)

Concatenate the contents of one string onto another.

Parameters

<i>s1</i>	The destination string
<i>s2</i>	The source string

Returns

A pointer to the destination string

4.32.1.10 int strcmp (const char * *s1*, const char * *s2*)

Compares two strings to each other

Parameters

<i>s1</i>	The first string
<i>s2</i>	The second string

Returns

The difference between the characters at the first index of indifference

4.32.1.11 char* strcpy (char * *cpy*, const char * *ori*)

Copy on string to another.

Parameters

<i>cpy</i>	The destination string
<i>ori</i>	The source string

Returns

A pointer to the destination string

4.32.1.12 int strlen (const char * s)

Returns the length of a string.

Parameters

s	The input string
---	------------------

Returns

The length of the string

4.32.1.13 char* strtok (char * s1, const char * s2)

Split string into tokens

Call this function multiple times (substituting NULL for s1) until NULL is returned to get all tokens.

Parameters

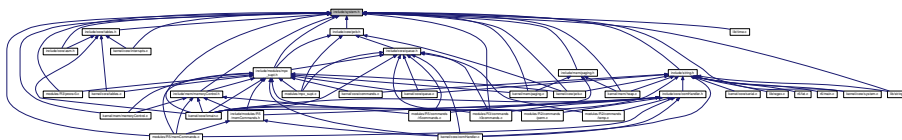
s1	The string to split
s2	The delimiter

Returns

A single token

4.33 include/system.h File Reference

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

**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

- void `klogv` (const char *msg)
- void `kpanic` (const char *msg)

4.33.1 Macro Definition Documentation

4.33.1.1 #define `asm __asm__`

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

Turn IRQs off.

4.33.1.3 #define `GDT_CS_ID` 0x01

Kernel code segment ID.

4.33.1.4 #define `GDT_DS_ID` 0x02

Kernel data segment ID.

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

Halt.

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

Interrupt return.

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

Suppress 'unused parameter' warnings/errors

Parameters

<i>p</i>	The parameter
----------	---------------

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

Skip cycle.

4.33.1.9 `#define NULL 0`

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

Turn IRQs on.

4.33.1.11 `#define volatile __volatile__`

4.33.2 Typedef Documentation

4.33.2.1 `typedef unsigned int size_t`

4.33.2.2 `typedef unsigned short u16int`

4.33.2.3 `typedef unsigned long u32int`

4.33.2.4 `typedef unsigned char u8int`

4.33.3 Function Documentation

4.33.3.1 `void klogv (const char * msg)`

Kernel log message. Sent to active serial device.

Parameters

<i>msg</i>	The message to log
------------	--------------------

4.33.3.2 `void kpanic (const char * msg)`

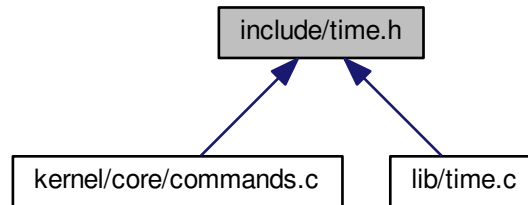
Kernel panic. Prints an error message and halts.

Parameters

<i>msg</i>	The error message to print
------------	----------------------------

4.34 include/time.h File Reference

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



Data Structures

- struct [date_time](#)

Macros

- #define [MON](#) ((const char*) "Monday")
- #define [TUE](#) ((const char*) "Tuesday")
- #define [WED](#) ((const char*) "Wednesday")
- #define [THU](#) ((const char*) "Thursday")
- #define [FRI](#) ((const char*) "Friday")
- #define [SAT](#) ((const char*) "Saturday")
- #define [SUN](#) ((const char*) "Sunday")
- #define [JAN](#) ((const char*) "January")
- #define [FEB](#) ((const char*) "February")
- #define [MAR](#) ((const char*) "March")
- #define [APR](#) ((const char*) "April")
- #define [MAY](#) ((const char*) "May")
- #define [JUN](#) ((const char*) "June")
- #define [JUL](#) ((const char*) "July")
- #define [AUG](#) ((const char*) "August")
- #define [SEP](#) ((const char*) "September")
- #define [OCT](#) ((const char*) "October")
- #define [NOV](#) ((const char*) "November")
- #define [DEC](#) ((const char*) "December")
- #define [SECONDS](#) 0x00
- #define [MINUTES](#) 0x02
- #define [HOURS](#) 0x04
- #define [DAY_WEEK](#) 0x06
- #define [DAY_MONTH](#) 0x07
- #define [MONTH](#) 0x08
- #define [YEAR](#) 0x09
- #define [CONTROL_PORT](#) 0x70
- #define [DATA_PORT](#) 0x71
- #define [TIME_DELIM](#) ':'
- #define [NMI_DISABLE](#) 0x80
- #define [NMI_ENABLE](#) 0x7F

Functions

- [date_time](#) [getDateTime](#) ()
- void [setDateTime](#) ([date_time](#))
- unsigned char [getSeconds](#) ()
- unsigned char [getMinutes](#) ()
- unsigned char [getHours](#) ()
- unsigned char [getDayOfWeek](#) ()
- unsigned char [getDayOfMonth](#) ()
- unsigned char [getMonth](#) ()
- unsigned char [getYear](#) ()
- void [setSeconds](#) (unsigned char seconds)
- void [setMinutes](#) (unsigned char minutes)
- void [setHours](#) (unsigned char hours)
- void [setDayOfWeek](#) (unsigned char dayOfWeek)
- void [setDayOfMonth](#) (unsigned char dayOfMonth)
- void [setMonth](#) (unsigned char month)
- void [setYear](#) (unsigned char year)
- void [updateDayOfWeek](#) ([date_time](#) *dateTime)
- void [updateDayOfYear](#) ([date_time](#) *dateTime)
- int [isLeapYear](#) (int year)

Variables

- const int [DAYS_IN_MONTH](#) [13]

4.34.1 Macro Definition Documentation

4.34.1.1 `#define APR ((const char*) "April")`

4.34.1.2 `#define AUG ((const char*) "August")`

4.34.1.3 `#define CONTROL_PORT 0x70`

Registers for reading/writing data

4.34.1.4 `#define DATA_PORT 0x71`

4.34.1.5 `#define DAY_MONTH 0x07`

4.34.1.6 `#define DAY_WEEK 0x06`

4.34.1.7 `#define DEC ((const char*) "December")`

4.34.1.8 `#define FEB ((const char*) "February")`

4.34.1.9 `#define FRI ((const char*) "Friday")`

4.34.1.10 `#define HOURS 0x04`

4.34.1.11 `#define JAN ((const char*) "January")`

Month Names

4.34.1.12 `#define JUL ((const char*) "July")`

4.34.1.13 `#define JUN ((const char*) "June")`

4.34.1.14 `#define MAR ((const char*) "March")`

4.34.1.15 `#define MAY ((const char*) "May")`

4.34.1.16 `#define MINUTES 0x02`

4.34.1.17 `#define MON ((const char*) "Monday")`

Day Names

4.34.1.18 `#define MONTH 0x08`

4.34.1.19 `#define NMI_DISABLE 0x80`

NMI Flags Disable - OR, Enable - AND

4.34.1.20 `#define NMI_ENABLE 0x7F`

4.34.1.21 `#define NOV ((const char*) "November")`

4.34.1.22 `#define OCT ((const char*) "October")`

4.34.1.23 `#define SAT ((const char*) "Saturday")`

4.34.1.24 `#define SECONDS 0x00`

Aliases for accessing time/date

4.34.1.25 `#define SEP ((const char*) "September")`

4.34.1.26 `#define SUN ((const char*) "Sunday")`

4.34.1.27 `#define THU ((const char*) "Thursday")`

4.34.1.28 `#define TIME_DELIM ':'`

The delimiter for the time

4.34.1.29 `#define TUE ((const char*) "Tuesday")`

4.34.1.30 `#define WED ((const char*) "Wednesday")`

4.34.1.31 `#define YEAR 0x09`

4.34.2 Function Documentation

4.34.2.1 `date_time getDateTime ()`

Gets the date and time from the RTC registers.

Returns

The date and time stored in the RTC.

4.34.2.2 `unsigned char getDayOfMonth ()`

Gets the day of the month (decimal-encoded) from the RTC.

Returns

The decimal-encoded day of the month.

4.34.2.3 `unsigned char getDayOfWeek ()`

Gets the day of the week (decimal-encoded) from the RTC.

Sunday - 1 Monday - 2 Tuesday - 3 Wednesday - 4 Thursday - 5 Friday - 6 Saturday - 7

Returns

The decimal-encoded day of the week.

4.34.2.4 `unsigned char getHours ()`

Gets the hours value (decimal-encoded) from the RTC.

Returns

The decimal-encoded number of hours.

4.34.2.5 unsigned char getMinutes ()

Gets the minutes value (decimal-encoded) from the RTC.

Returns

The decimal-encoded number of minutes.

4.34.2.6 unsigned char getMonth ()

Gets the month (decimal-encoded) from the RTC.

Returns

The decimal-encoded month.

4.34.2.7 unsigned char getSeconds ()

Gets the seconds value (decimal-encoded) from the RTC.

Returns

The decimal-encoded number of seconds.

4.34.2.8 unsigned char getYear ()

Gets the year (decimal-encoded) from the RTC.

Returns

The decimal-encoded year.

4.34.2.9 int isLeapYear (int *year*)

Determines if the given year is a leap year.

Parameters

<i>year</i>	The year to check
-------------	-------------------

Returns

True if the year is a leap year.

4.34.2.10 void setDateTime (date_time *dateTime*)

Sets the date and time to the specified values.

Day of month must be specified but day of week/year will be automatically calculated.

Parameters

<i>dateTime</i>	The values to set.
-----------------	--------------------

4.34.2.11 void setDayOfMonth (unsigned char *day*)

Sets the day of the month register in the RTC. This number should be decimal-encoded.

Parameters

<i>dayOfMonth</i>	The day of the month value to set
-------------------	-----------------------------------

4.34.2.12 void setDayOfWeek (unsigned char *day*)

Sets the day of the week register in the RTC. This number should be decimal-encoded.

Sunday - 1 Monday - 2 Tuesday - 3 Wednesday - 4 Thursday - 5 Friday - 6 Saturday - 7

Parameters

<i>dayOfWeek</i>	The day of the week value to set
------------------	----------------------------------

4.34.2.13 void setHours (unsigned char *hour*)

Sets the hours register in the RTC. This number should be decimal-encoded.

Parameters

<i>hours</i>	The hours value to set
--------------	------------------------

4.34.2.14 void setMinutes (unsigned char *min*)

Sets the minutes register in the RTC. This number should be decimal-encoded.

Parameters

<i>minutes</i>	The minutes value to set
----------------	--------------------------

4.34.2.15 void setMonth (unsigned char *mon*)

Sets the month register in the RTC. This number should be decimal-encoded.

Parameters

<i>month</i>	The month value to set
--------------	------------------------

4.34.2.16 void setSeconds (unsigned char *sec*)

Sets the seconds register in the RTC. This number should be decimal-encoded.

Parameters

<i>seconds</i>	The seconds value to set
----------------	--------------------------

4.34.2.17 void setYear (unsigned char *year*)

Sets the year register in the RTC. This number should be decimal-encoded.

Parameters

<i>year</i>	The year value to set
-------------	-----------------------

4.34.2.18 void updateDayOfWeek (*date_time* * *dateTime*)

Sets the day of week property of the [date_time](#) struct based on the year, month, and day of month values.

Sunday - 1 Monday - 2 Tuesday - 3 Wednesday - 4 Thursday - 5 Friday - 6 Saturday - 7

Parameters

<i>dateTime</i>	The date_time to update
-----------------	-----------------------------------------

Sets the day of week property of the [date_time](#) struct based on the year, month, and day of month values.

Sunday - 1 Monday - 2 Tuesday - 3 Wednesday - 4 Thursday - 5 Friday - 6 Saturday - 7

Parameters

<i>dateTime</i>	The date_time to update.
-----------------	------------------------------------------

4.34.2.19 void updateDayOfYear (*date_time* * *dateTime*)

Sets the day of year property of the [date_time](#) struct based on the year, month, and day of month values.

- void `addComHistory` (char *newCom)
- void `printStart` ()
- void `returnToInsertionPoint` (int endIndex, int insertionIndex)
- void `eraseCurrentRow` (int endIndex, int insertionIndex)
- char * `getInput` ()
- void `executeCommand` (char *commandString)
- const char * `help` (char **args, int numArgs)
- const char * `shutdown` (char **args, int numArgs)
- void `setupCommands` ()
- void `initCommandHandler` ()

Variables

- int `continueHandle` = 1
- char `buffer` [256]
- `functionDef` `functionDefs` [256]
- int `functionInsertPoint` = 0
- char `comHistory` [10][256]
- int `comHistoryPos` = 0

4.35.1 Function Documentation

4.35.1.1 void addComHistory (char * newCom)

Helper function to add a command to the command history array

Parameters

<i>newCom</i>	string to add to the command history
---------------	--------------------------------------

4.35.1.2 void addFunctionDef (char * name, const char * helpString, const char * funcPointer)(char **args, int numArgs)

Adds function definition struct, created from provided params to the functionDefs array This allows the function to be called in the command handler by its name

Parameters

<i>name</i>	- string representation of the function
<i>helpString</i>	- const string to be displayed for help
<i>funcPointer</i>	- pointer to the function, must return const char* and take in arguments: char** args and int numArgs

4.35.1.3 void eraseCurrentRow (int endIndex, int insertionIndex)

Helper function to remove all printed chars on the current line of input back to the >>

Parameters

<i>endIndex</i>	- index of last char printed
<i>insertionIndex</i>	- index of where insertion point should be

4.35.1.4 void executeCommand (char * *commandString*)

Gets the command in the given *commandString* param and executes it, printing the provided output string

Parameters

<i>commandString</i>	string containing the command name and any args
----------------------	-------------------------------------------------

4.35.1.5 char* getComHistory (int *isPrev*)

Helper function to get the next or previous command from the command history

Parameters

<i>isPrev</i>	integer denoting if to get the previous command
---------------	-------------------------------------------------

Returns

string of the command

4.35.1.6 functionDef getFunctionDef (char * *name*)

Gets the [functionDef](#) struct corresponding to the name provided, returns a [functionDef](#) with null funcPointer if none are found

Parameters

<i>name</i>	- name of the functionDef
-------------	-------------------------------------------

Returns

[functionDef](#)

4.35.1.7 const char* getHelpString (char * *name*)

Gets the help string from the struct for the function name provided

Parameters

<i>name</i>	- name associated with the struct from which to get the help string
-------------	---------------------------------------------------------------------

Returns

const char* help string

4.35.1.8 char* getInput ()

Polls the input for characters and handles special key strokes such as delete, backspace, arrows, etc. and returns the input string

Returns

string that was input

4.35.1.9 const char* help (char ** args, int numArgs)

Returns help for the specified commands.

Usage: help commandName

Args: [no args] - Returns the help for the help command commandName - The name of the command to get help for

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

The help string

4.35.1.10 void initCommandHandler ()

Main function of the comHandler that initializes the command handler, continually loops taking in input commands, manages the comHistory, and executes given commands

4.35.1.11 void printStart ()

Helper function to print out the beginning line tag: ">>"

4.35.1.12 void returnToInsertionPoint (int endIndex, int insertionIndex)

Helper function to move the insertion point from the end of the line to the correct placement

Parameters

<i>endIndex</i>	- index of last char printed
<i>insertionIndex</i>	- index of where insertion point should be

4.35.1.13 void setupCommands ()

Initialization function to add commands ot the functionDefs array

4.35.1.14 const char* shutdown (char ** args, int numArgs)

Shuts down the OS after asking for confirmation.

Usage: shutdown [-confirm]

Args: [no args] - Displays confirmation prompt –confirm - Auto-confirms shutdown

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

True if shutdown was confirmed

4.35.2 Variable Documentation

4.35.2.1 char buffer[256]

4.35.2.2 char comHistory[10][256]

4.35.2.3 int comHistoryPos = 0

4.35.2.4 int continueHandle = 1

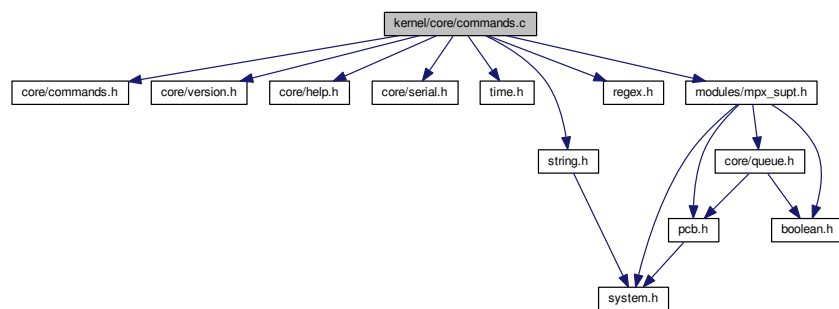
4.35.2.5 functionDef functionDefs[256]

4.35.2.6 int functionInsertPoint = 0

4.36 kernel/core/commands.c File Reference

```
#include <core/commands.h>
#include <core/version.h>
#include <core/help.h>
#include <core/serial.h>
#include <time.h>
#include <string.h>
#include <regex.h>
#include <modules/mpx_supt.h>
```

Include dependency graph for commands.c:



Functions

- `const char * version (char **args, int numArgs)`
- `const char * date (char **args, int numArgs)`

4.36.1 Function Documentation

4.36.1.1 `const char* date (char ** args, int numArgs)`

Returns the current date/time in ISO-8601 format. Improperly specified date/times are ignored.

Usage: `date [-date] [-time] [--setdate yyyy-MM-dd] [--settime hh:mm:ss]`

Args: [no args] - Return the date and time `-date` - Return the date `-time` - Return the time `-setdate` - Sets the date to the specified date (returns the new date) `-settime` - Sets the time to the specified time (returns the new time)

Parameters

<code>args</code>	The arguments to pass to the function
-------------------	---------------------------------------

Returns

The ISO-8601 formatted date

4.36.1.2 `const char* version (char ** args, int numArgs)`

Returns the current version of the OS.

Usage: version

Args: [no args] - Returns the version

Parameters

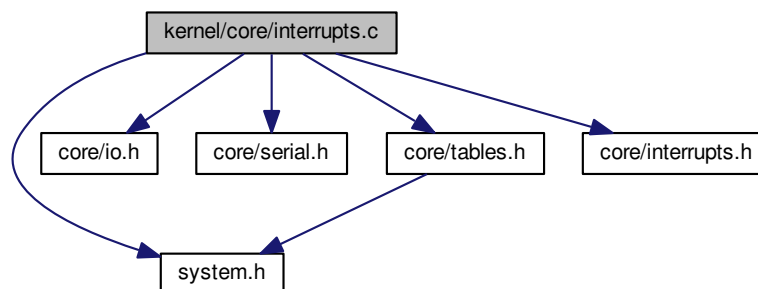
<code>args</code>	The arguments to pass to the function
-------------------	---------------------------------------

Returns

The version of the OS.

4.37 `kernel/core/interrupts.c` File Reference

```
#include <system.h>
#include <core/io.h>
#include <core/serial.h>
#include <core/tables.h>
#include <core/interrupts.h>
Include dependency graph for interrupts.c:
```



Macros

- `#define PIC1 0x20`
- `#define PIC2 0xA0`
- `#define ICW1 0x11`
- `#define ICW4 0x01`
- `#define io_wait() asm volatile ("outb $0x80")`

Functions

- void [divide_error](#) ()
- void [debug](#) ()
- void [nmi](#) ()
- void [breakpoint](#) ()
- void [overflow](#) ()
- void [bounds](#) ()
- void [invalid_op](#) ()
- void [device_not_available](#) ()
- void [double_fault](#) ()
- void [coprocessor_segment](#) ()
- void [invalid_tss](#) ()
- void [segment_not_present](#) ()
- void [stack_segment](#) ()
- void [general_protection](#) ()
- void [page_fault](#) ()
- void [reserved](#) ()
- void [coprocessor](#) ()
- void [rtc_isr](#) ()
- void [sys_call_isr](#) ()
- void [isr0](#) ()
- void [do_isr](#) ()
- void [init_irq](#) (void)
- void [init_pic](#) (void)
- void [do_divide_error](#) ()
- void [do_debug](#) ()
- void [do_nmi](#) ()
- void [do_breakpoint](#) ()
- void [do_overflow](#) ()
- void [do_bounds](#) ()
- void [do_invalid_op](#) ()
- void [do_device_not_available](#) ()
- void [do_double_fault](#) ()
- void [do_coprocessor_segment](#) ()
- void [do_invalid_tss](#) ()
- void [do_segment_not_present](#) ()
- void [do_stack_segment](#) ()
- void [do_general_protection](#) ()
- void [do_page_fault](#) ()
- void [do_reserved](#) ()
- void [do_coprocessor](#) ()

Variables

- idt_entry [idt_entries](#) [256]

4.37.1 Macro Definition Documentation

4.37.1.1 `#define ICW1 0x11`

4.37.1.2 `#define ICW4 0x01`

4.37.1.3 `#define io_wait() asm volatile ("outb $0x80")`

The i386 can do an io wait by accessing another port. Mainly used in initializing the PIC.

4.37.1.4 `#define PIC1 0x20`

4.37.1.5 `#define PIC2 0xA0`

4.37.2 Function Documentation

4.37.2.1 `void bounds ()`

4.37.2.2 `void breakpoint ()`

4.37.2.3 `void coprocessor ()`

4.37.2.4 `void coprocessor_segment ()`

4.37.2.5 `void debug ()`

4.37.2.6 `void device_not_available ()`

4.37.2.7 `void divide_error ()`

4.37.2.8 `void do_bounds ()`

4.37.2.9 `void do_breakpoint ()`

4.37.2.10 `void do_coprocessor ()`

4.37.2.11 `void do_coprocessor_segment ()`

4.37.2.12 `void do_debug ()`

4.37.2.13 `void do_device_not_available ()`

4.37.2.14 `void do_divide_error ()`

4.37.2.15 `void do_double_fault ()`

4.37.2.16 `void do_general_protection ()`

4.37.2.17 `void do_invalid_op ()`

4.37.2.18 `void do_invalid_tss ()`

4.37.2.19 `void do_isr ()`

4.37.2.20 `void do_nmi ()`

4.37.2.21 void do_overflow ()

4.37.2.22 void do_page_fault ()

4.37.2.23 void do_reserved ()

4.37.2.24 void do_segment_not_present ()

4.37.2.25 void do_stack_segment ()

4.37.2.26 void double_fault ()

4.37.2.27 void general_protection ()

4.37.2.28 void init_irq (void)

Installs the initial interrupt handlers for the first 32 irq lines. Most do a panic for now.

4.37.2.29 void init_pic (void)

Initializes the programmable interrupt controllers and performs the necessary remapping of IRQs. Leaves interrupts turned off.

4.37.2.30 void invalid_op ()

4.37.2.31 void invalid_tss ()

4.37.2.32 void isr0 ()

4.37.2.33 void nmi ()

4.37.2.34 void overflow ()

4.37.2.35 void page_fault ()

4.37.2.36 void reserved ()

4.37.2.37 void rtc_isr ()

4.37.2.38 void segment_not_present ()

4.37.2.39 void stack_segment ()

4.37.2.40 void sys_call_isr ()

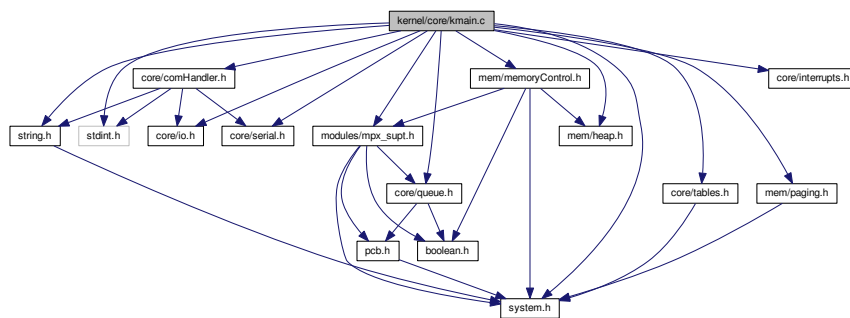
4.37.3 Variable Documentation

4.37.3.1 idt_entry idt_entries[256]

4.38 kernel/core/kmain.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <system.h>
#include <core/io.h>
#include <core/serial.h>
#include <core/tables.h>
#include <core/interrupts.h>
#include <core/queue.h>
#include <core/comHandler.h>
#include <mem/heap.h>
#include <mem/paging.h>
#include <mem/memoryControl.h>
#include <modules/mpx_supt.h>
```

Include dependency graph for kmain.c:



Functions

- void [kmain](#) (void)

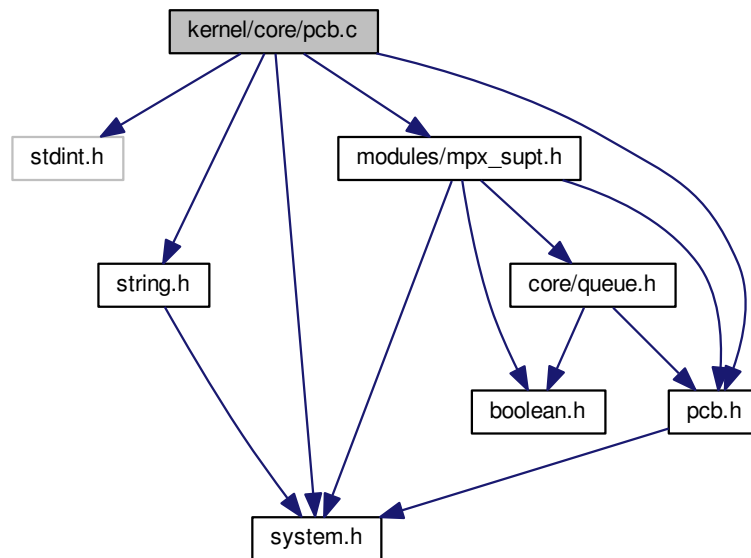
4.38.1 Function Documentation

4.38.1.1 void kmain (void)

4.39 kernel/core/pcb.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <modules/mpx_supt.h>
#include <system.h>
#include <core/pcb.h>
```

Include dependency graph for pcb.c:



Functions

- [pcb * allocatePCB \(\)](#)
- [int freePCB \(pcb *pcbPtr\)](#)
- [pcb * setupPCB \(const char *processName, int processClass, int priority\)](#)
- [int checkParamName \(const char *processName\)](#)
- [int checkParamClass \(int processClass\)](#)
- [int checkParamPriority \(int priority\)](#)

4.39.1 Function Documentation

4.39.1.1 `pcb* allocatePCB ()`

Allocates memory for a new PCB and returns a pointer to it

`freePCB` should be used when done using the `pcb` to free the memory in use

Returns

PCB pointer or Null if error occurs

4.39.1.2 `int checkParamClass (int processClass)`

Validates that the `processClass` is valid

Parameters

<i>processClass</i>	- int
---------------------	-------

Returns

integer 0 or 1 if valid

4.39.1.3 int checkParamName (const char * *processName*)

Validates that the processName is valid

Parameters

<i>processName</i>	- const char * processName
--------------------	----------------------------

Returns

integer 0 or 1 if valid

4.39.1.4 int checkParamPriority (int *priority*)

Validates that the priority is valid

Parameters

<i>priority</i>	- int
-----------------	-------

Returns

integer 0 or 1 if valid

4.39.1.5 int freePCB (pcb * *pcbPtr*)

Frees memory that is allocated for the pcb provided

Parameters

<i>pcbPtr</i>	pointer to pcb to be freed
---------------	----------------------------

Returns

integer code - 1 if successful, 0 otherwise

4.39.1.6 `pcb* setupPCB (const char * processName, int processClass, int priority)`

Allocates memory for a new PCB and sets it with given params

Parameters

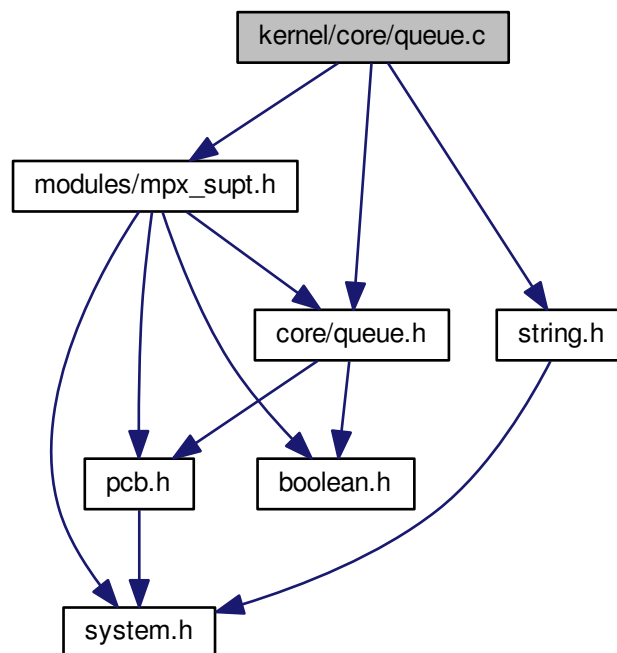
<i>processName</i>	- const string name
<i>processClass</i>	- integer identifying as system or application process (0, 1)
<i>priority</i>	- integer between 0 and 9 indicating priority

Returns

PCB pointer to new pcb or NULL if there were errors

4.40 kernel/core/queue.c File Reference

```
#include <core/queue.h>
#include <modules/mpx_supt.h>
#include <string.h>
Include dependency graph for queue.c:
```



Enumerations

- enum `queue` { `QUEUE_BLOCKED` = `BLOCKED`, `QUEUE_READY` = `READY`, `QUEUE_SUSPENDED_BLOCKED` = `BLOCKED + 0x02`, `QUEUE_SUSPENDED_READY` = `READY + 0x02` }

Functions

- `node * _newNode (pcb *p)`
- `node * _findNode (const char *processName)`
- `node * _findNodeInQueue (queue q, const char *processName)`
- `boolean _insertPriority (queue q, node *newNode)`
- `boolean _insertFIFO (queue q, node *newNode)`
- `node * getReadyQueue ()`
- `node * getBlockedQueue ()`
- `node * getSuspendedReadyQueue ()`
- `node * getSuspendedBlockedQueue ()`
- `pcb * popReady ()`
- `pcb * popBlocked ()`
- `pcb * popSuspendedReady ()`
- `pcb * popSuspendedBlocked ()`
- `boolean insertPCB (pcb *p)`
- `boolean removePCB (pcb *p)`
- `pcb * findPCB (const char *processName)`

Variables

- `node * queues [4]`

4.40.1 Enumeration Type Documentation

4.40.1.1 enum queue

Enumerator

QUEUE_BLOCKED
QUEUE_READY
QUEUE_SUSPENDED_BLOCKED
QUEUE_SUSPENDED_READY

4.40.2 Function Documentation

4.40.2.1 `node * _findNode (const char * processName)`

Internal function to find the node containing the PCB with the given process name.

Parameters

<i>processName</i>	The process name to search for
--------------------	--------------------------------

Returns

The node containing the PCB with the given process name, or null if not found

4.40.2.2 node * _findNodeInQueue (queue *q*, const char * *processName*)

Internal function for finding a node in a specific queue

Parameters

<i>q</i>	The queue to search in
<i>processName</i>	The process name to search for

Returns

The node containing the PCB with the given name, or null if not found

4.40.2.3 boolean _insertFIFO (queue *q*, node * *newNode*)

Inserts a node into a FIFO queue

Parameters

<i>q</i>	The queue to insert into
<i>newNode</i>	The node to insert

Returns

true if the node was inserted, false otherwise

4.40.2.4 boolean _insertPriority (queue *q*, node * *newNode*)

Internal function for inserting a node into a given queue in order by priority.

Parameters

<i>q</i>	The queue to insert into
<i>newNode</i>	The node to insert

Returns

true if the node was inserted, false otherwise

4.40.2.5 node * _newNode (pcb * *pcb*)

Internal function to create a new list node.

Parameters

<i>pcb</i>	The pcb to store in the node
------------	------------------------------

Returns

A pointer to the created node, or NULL if the node can't be created

4.40.2.6 `pcb* findPCB (const char * processName)`

Finds the PCB with the given process name.

Parameters

<i>processName</i>	The name of the process to search for
--------------------	---------------------------------------

Returns

A pointer to the PCB, or null if not found

4.40.2.7 `node* getBlockedQueue ()`

Gets the head node of the blocked queue.

Returns

The head node of the blocked queue

4.40.2.8 `node* getReadyQueue ()`

Gets the head node of the ready queue.

Returns

The head node of the ready queue

4.40.2.9 `node* getSuspendedBlockedQueue ()`

Gets the head node of the suspended-blocked queue.

Returns

The head node of the suspended-blocked queue

4.40.2.10 node* getSuspendedReadyQueue ()

Gets the head node of the suspended-ready queue.

Returns

The head node of the suspended-ready queue

4.40.2.11 boolean insertPCB (pcb * p)

Inserts the PCB into the appropriate queue.

Parameters

<i>p</i>	The PCB to insert.
----------	--------------------

Returns

true if the PCB was inserted, false otherwise

4.40.2.12 pcb* popBlocked ()

Pops the next node off of the blocked queue.

Returns

The next node of the blocked queue, or NULL if it is empty

4.40.2.13 pcb* popReady ()

Pops the next node off of the ready queue.

Returns

The next node of the ready queue, or NULL if it is empty

4.40.2.14 pcb* popSuspendedBlocked ()

Pops the next node off of the suspended-blocked queue.

Returns

The next node of the suspended-blocked queue, or NULL if it is empty

4.40.2.15 `pcb* popSuspendedReady ()`

Pops the next node off of the suspended-ready queue.

Returns

The next node of the suspended-ready queue, or NULL if it is empty

4.40.2.16 `boolean removePCB (pcb * p)`

Removes the given PCB from it's queue.

Parameters

<code>p</code>	The PCB to remove
----------------	-------------------

Returns

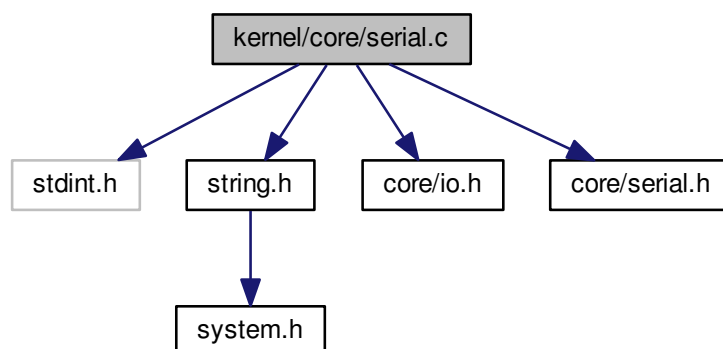
true if the PCB was removed, false otherwise

4.40.3 Variable Documentation

4.40.3.1 `node* queues[4]`

4.41 `kernel/core/serial.c` File Reference

```
#include <stdint.h>
#include <string.h>
#include <core/io.h>
#include <core/serial.h>
Include dependency graph for serial.c:
```



Macros

- `#define NO_ERROR 0`

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)`

Variables

- `int serial_port_out = 0`
- `int serial_port_in = 0`

4.41.1 Macro Definition Documentation

4.41.1.1 `#define NO_ERROR 0`

4.41.2 Function Documentation

4.41.2.1 `int init_serial (int device)`

Initializes devices for user interaction, logging, ...

Parameters

<i>device</i>	The device to initialize
---------------	--------------------------

Returns

The error code

4.41.2.2 `int serial_print (const char * msg)`

Writes a message to the active serial output device.

Parameters

<i>msg</i>	The message to write
------------	----------------------

Returns

The error code

4.41.2.3 int serial_println (const char * *msg*)

Writes a message to the active serial output device. Appends a newline character.

Parameters

<i>msg</i>	The message to write
------------	----------------------

Returns

The error code

4.41.2.4 int set_serial_in (int *device*)

Sets serial_port_in to the given device address. All serial input, such as console input via a virtual machine, QEMU/Bochs/etc, will be directed to the device.

Parameters

<i>device</i>	The device to set as input
---------------	----------------------------

Returns

The error code

4.41.2.5 int set_serial_out (int *device*)

Sets serial_port_out to the given device address. All serial output, such as that from serial_println, will be directed to this device.

Parameters

<i>device</i>	The device to set as output
---------------	-----------------------------

Returns

The error code

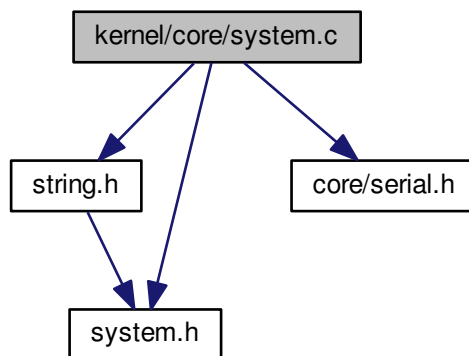
4.41.3 Variable Documentation**4.41.3.1 int serial_port_in = 0**

4.41.3.2 `int serial_port_out = 0`

4.42 kernel/core/system.c File Reference

```
#include <string.h>
#include <system.h>
#include <core/serial.h>
```

Include dependency graph for system.c:



Functions

- void `klogv` (const char *msg)
- void `kpanic` (const char *msg)

4.42.1 Function Documentation

4.42.1.1 void `klogv` (const char * *msg*)

Kernel log message. Sent to active serial device.

Parameters

<i>msg</i>	The message to log
------------	--------------------

4.42.1.2 void `kpanic` (const char * *msg*)

Kernel panic. Prints an error message and halts.

Parameters

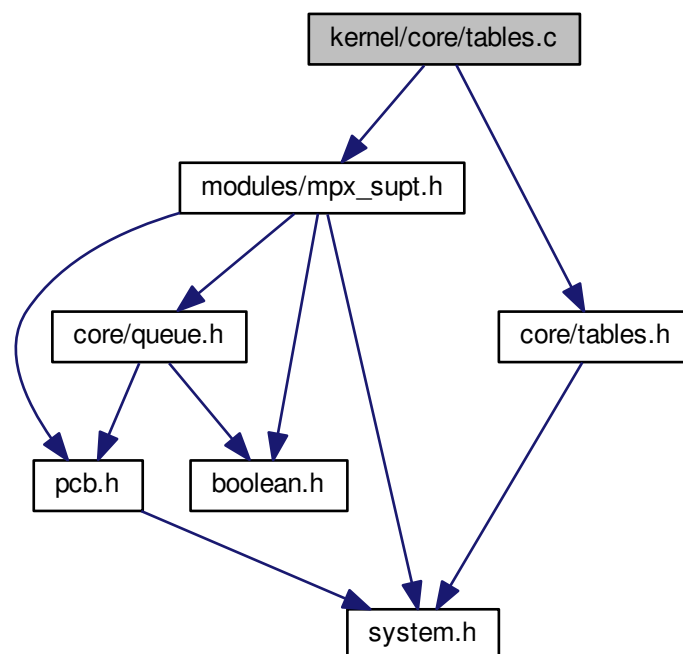
<i>msg</i>	The error mesage to print
------------	---------------------------

4.43 kernel/core/tables.c File Reference

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

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

Include dependency graph for tables.c:



Functions

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

Variables

- `gdt_descriptor` `gdt_ptr`
- `gdt_entry` `gdt_entries` [5]
- `idt_descriptor` `idt_ptr`
- `idt_entry` `idt_entries` [256]

4.43.1 Function Documentation

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

Installs a new table entry into the global descriptor table.

Parameters

<i>idx</i>	
<i>base</i>	
<i>limit</i>	
<i>access</i>	
<i>flags</i>	

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

Installs a new gate entry into the IDT.

Parameters

<i>idx</i>	
<i>base</i>	
<i>sel</i>	
<i>flags</i>	

4.43.1.3 void init_gdt ()

Creates the global descriptor table and installs it using the defined assembly routine.

4.43.1.4 void init_idt ()

Creates the interrupt descriptor table and writes the pointer using the defined assembly function.

4.43.1.5 void write_gdt_ptr (u32int , size_t)

4.43.1.6 void write_idt_ptr (u32int)

4.43.2 Variable Documentation

4.43.2.1 gdt_entry gdt_entries[5]

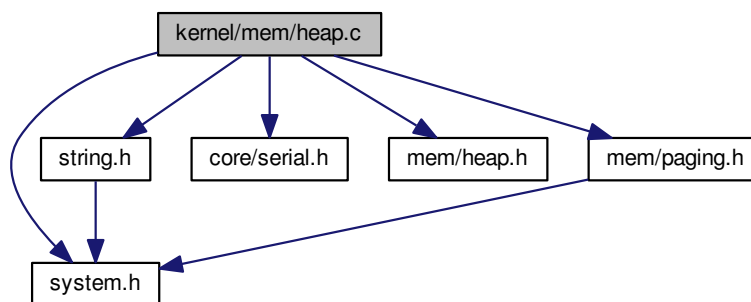
4.43.2.2 gdt_descriptor gdt_ptr

4.43.2.3 `idt_entry idt_entries[256]`

4.43.2.4 `idt_descriptor idt_ptr`

4.44 `kernel/mem/heap.c` File Reference

```
#include <system.h>
#include <string.h>
#include <core/serial.h>
#include <mem/heap.h>
#include <mem/paging.h>
Include dependency graph for heap.c:
```



Functions

- `u32int _kmalloc (u32int size, int page_align, u32int *phys_addr)`
- `u32int kmalloc (u32int size)`
- `u32int alloc (u32int size, heap *h, int align)`
- `heap * make_heap (u32int base, u32int max, u32int min)`

Variables

- `heap * kheap = 0`
- `heap * curr_heap = 0`
- `page_dir * kdir`
- `void * end`
- `void _end`
- `void __end`
- `u32int phys_alloc_addr = (u32int) &end`

4.44.1 Function Documentation

4.44.1.1 `u32int _kmalloc (u32int size, int page_align, u32int * phys_addr)`

Base-level kernel memory allocation routine. Used to provide page alignment and access physical addresses of allocations. Called by `kmalloc` with `align=0`, `physical_address=0`.

Parameters

<i>size</i>	The amount of memory to allocate
<i>align</i>	The page alignment
<i>phys_addr</i>	The physical address

Returns

The memory address

4.44.1.2 `u32int alloc (u32int size, heap * h, int align)`

Allocates some memory using the given heap. Can specify page-alignment.

Parameters

<i>size</i>	The amount of memory to allocate
<i>hp</i>	The heap to allocate on
<i>align</i>	The page alignment

Returns

The memory address

4.44.1.3 `u32int kmalloc (u32int size)`

Standard memory allocation routine. Use this unless you need to specify alignment or obtain a physical address. Calls `_kmalloc`.

Parameters

<i>size</i>	The amount of memory to allocate
-------------	----------------------------------

Returns

The memory address

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

Create a new heap.

Parameters

<i>base</i>	Physical start address of the heap
<i>max</i>	Maximum size the heap may grow to
<i>min</i>	Minium/Initial size

Returns

The address of the heap

4.44.2 Variable Documentation

4.44.2.1 `void __end`

4.44.2.2 `void _end`

4.44.2.3 `heap* curr_heap = 0`

4.44.2.4 `void* end`

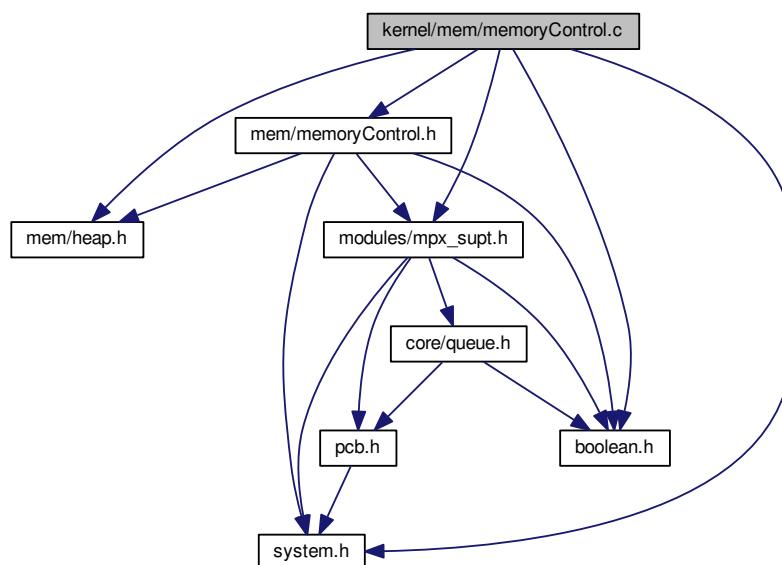
4.44.2.5 `page_dir* kdir`

4.44.2.6 `heap* kheap = 0`

4.44.2.7 `u32int phys_alloc_addr = (u32int) &end`

4.45 kernel/mem/memoryControl.c File Reference

```
#include <system.h>
#include <mem/heap.h>
#include <mem/memoryControl.h>
#include <modules/mpx_supt.h>
#include <boolean.h>
Include dependency graph for memoryControl.c:
```



Functions

- `cmcb * _placeStructs` (int size, void *pos, int type, `cmcb *prev`, `cmcb *next`)
- `void _mergeAdjacentFree` ()
- `boolean initializeHeap` (int size)
- `void * allocateMemory` (int size)
- `boolean deallocateMemory` (void *memPointer)
- `boolean isEmpty` ()
- `cmcb * getFreeHead` ()
- `cmcb * getAllocatedHead` ()

Variables

- `cmcb * freeHead`
- `cmcb * allocatedHead`
- `void * memHeap`
- `int isInitialized = false`
- `int memSize`
- `int memAllocated`

4.45.1 Function Documentation

4.45.1.1 `void _mergeAdjacentFree` ()

Private helper function to merge all adjacent memory blocks

4.45.1.2 `cmcb * _placeStructs` (int size, void * pos, int type, `cmcb * prev`, `cmcb * next`)

Private helper function to create structs to denote the beginning and end of a memory block

Parameters

<i>size</i>	- size of block in bytes
<i>pos</i>	- mem location of beginning
<i>type</i>	- type of mem block, either ALLOCATED or FREE
<i>prev</i>	- pointer to previous cmcb
<i>next</i>	- pointer to next cmcb

Returns

pointer to create cmcb at beginning of memory block

4.45.1.3 `void* allocateMemory` (int size)

Allocates a memory block if enough memory is available

Parameters

<i>size</i>	- size of memory to allocate in bytes
-------------	---------------------------------------

Returns

pointer to the me

4.45.1.4 boolean deallocateMemory (void * *memPointer*)

Deallocates the block of memory at the mempointer

Parameters

<i>memPointer</i>	- pointer to the mem block
-------------------	----------------------------

Returns

boolean - boolean telling whether succesful dealloc

4.45.1.5 cmcb* getAllocatedHead ()

Returns the head to the allocated list

Returns

cmcb * to the allocated list head

4.45.1.6 cmcb* getFreeHead ()

Returns the head of the free list

Returns

cmcb * to the free list head

4.45.1.7 boolean initializeHeap (int *size*)

Initializes the heap to the provided size and creates a free mem block across it

Parameters

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

Returns

boolean - boolean denoting if heap was initialized

4.45.1.8 boolean isEmpty ()

Returns a boolean telling if all the memory is empty

Returns

boolean

4.45.2 Variable Documentation

4.45.2.1 cmcb* allocatedHead

4.45.2.2 cmcb* freeHead

4.45.2.3 int isInitialized = false

4.45.2.4 int memAllocated

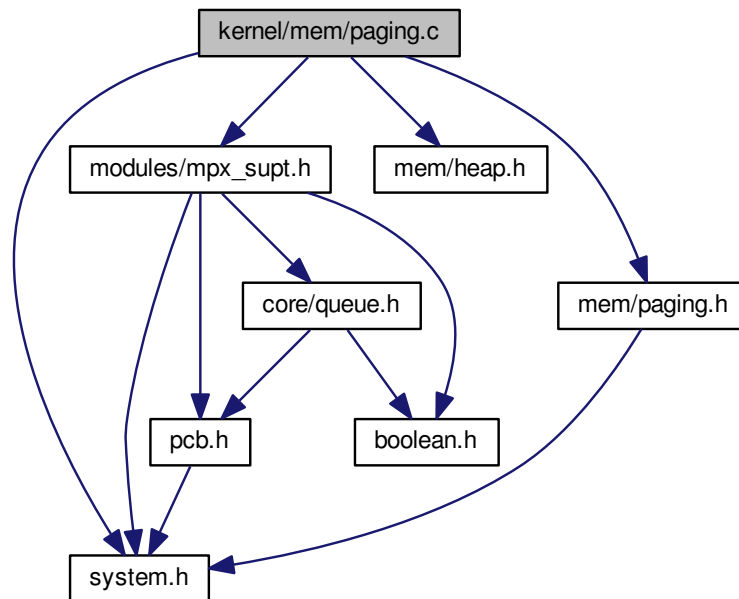
4.45.2.5 void* memHeap

4.45.2.6 int memSize

4.46 kernel/mem/paging.c File Reference

```
#include <system.h>
#include <modules/mpx_supt.h>
#include "mem/heap.h"
#include "mem/paging.h"
```

Include dependency graph for paging.c:



Functions

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

Variables

- u32int `mem_size` = 0x4000000
- u32int `page_size` = 0x1000
- u32int `nframes`
- u32int * `frames`
- page_dir * `kdir` = 0
- page_dir * `cdir` = 0
- u32int `phys_alloc_addr`
- heap * `kheap`

4.46.1 Function Documentation

4.46.1.1 void `clear_bit` (u32int *addr*)

Marks a page frame bit as free (0).

Parameters

<i>addr</i>	The address of the frame
-------------	--------------------------

4.46.1.2 u32int first_free ()

Finds the first free page frame.

Returns

The first free page frame

4.46.1.3 u32int get_bit (u32int *addr*)

Checks if page frame is in use.

Parameters

<i>addr</i>	The address of the frame
-------------	--------------------------

Returns

True if it is in use

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

Finds and returns a page, allocating a new page table if necessary.

Parameters

<i>addr</i>	The address of the page
<i>dir</i>	The page directory
<i>make_table</i>	Boolean to create a table if necessary

Returns

A pointer to the page

4.46.1.5 void init_paging ()

Initializes the kernel page directory and initial kernel heap area. Performs identity mapping of the kernel frames such that the virtual addresses are equivalent to the physical addresses.

4.46.1.6 void load_page_dir (page_dir * new_dir)

Sets a page directory as the current directory and enables paging via the CR0 register, The CR3 register enables address translation from linear to physical address.

http://en.wikipedia.org/wiki/Control_register#Control_registers_in_x86_↵series

Parameters

<i>new_page_dir</i>	The page directory to set as the current
---------------------	------------------------------------------

4.46.1.7 void new_frame (page_entry * page)

Marks a frame as in use un the frame bitmap, sets up the page, and saves* the frame index in the page.

Parameters

<i>page</i>	The page to create the frame in
-------------	---------------------------------

4.46.1.8 void set_bit (u32int addr)

Marks a page frame bit as in use (1).

Parameters

<i>addr</i>	The address of the frame
-------------	--------------------------

4.46.2 Variable Documentation

4.46.2.1 page_dir* cdir = 0

4.46.2.2 u32int* frames

4.46.2.3 page_dir* kdir = 0

4.46.2.4 heap* kheap

4.46.2.5 u32int mem_size = 0x4000000

4.46.2.6 u32int nframes

4.46.2.7 u32int page_size = 0x1000

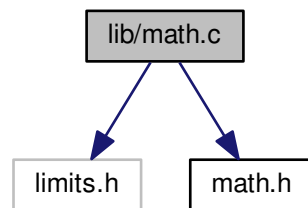
4.46.2.8 u32int phys_alloc_addr

4.47 lib/math.c File Reference

```
#include <limits.h>
```

```
#include <math.h>
```

Include dependency graph for math.c:



Functions

- unsigned char [bcdToDec](#) (unsigned char *bcd*)
- unsigned char [decToBcd](#) (unsigned char *dec*)

4.47.1 Function Documentation

4.47.1.1 unsigned char [bcdToDec](#) (unsigned char *bcd*)

Converts a BCD encoded byte to a decimal encoded byte

Parameters

<i>bcd</i>	The value to convert
------------	----------------------

Returns

The decimal value

4.47.1.2 unsigned char [decToBcd](#) (unsigned char *dec*)

Converts a decimal encoded byte to a BCD encoded byte

Parameters

<i>dec</i>	The value to convert
------------	----------------------

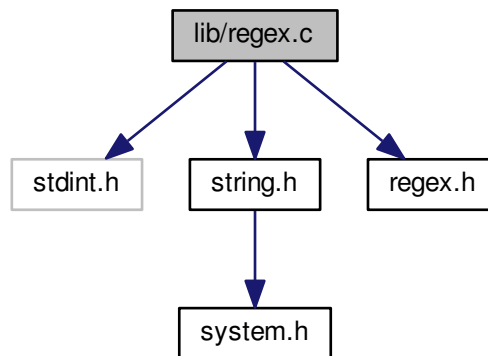
Returns

The BCD value

4.48 lib/regex.c File Reference

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

Include dependency graph for regex.c:



Functions

- int [testRegex](#) (const char **regex*, const char **stringToCheck*)

4.48.1 Function Documentation

4.48.1.1 int testRegex (const char * *regex*, const char * *stringToCheck*)

Tests if the *stringToCheck* adheres to the given *regex* string

The *regex* string is comprised of: d matches digits 0-9 c matches characters a-zA-Z u matches uppercase character, A-Z l matches lowercase character, a-z

- matches any char /char for a literal character, ex: "/a" matches 'a', /d matches 'd'

Example : *regex* "dcd/a" matches any string with the pattern "digit character digit 'a'", ex "1b3a", "6b9a"

Parameters

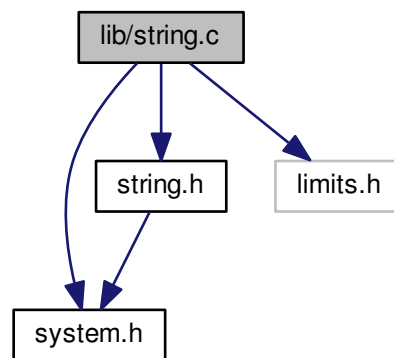
<i>regex</i>	
<i>stringToCheck</i>	

Returns

1 if adheres, 0 otherwise

4.49 lib/string.c File Reference

```
#include <system.h>
#include <string.h>
#include <limits.h>
Include dependency graph for string.c:
```



Functions

- int `strlen` (const char *s)
- char * `strcpy` (char *cpy, const char *ori)
- int `isdigit` (const char c)
- int `atoi` (const char *s)
- void `itoa` (int num, char *str, int base)
- void `reverse` (char *str, int len)
- int `strcmp` (const char *s1, const char *s2)
- char * `strcat` (char *s1, const char *s2)
- int `isspace` (const char *c)
- int `isChar` (const char c)
- int `isUpperChar` (const char c)
- int `isLowerChar` (const char c)
- char * `strtok` (char *s1, const char *s2)

4.49.1 Function Documentation

4.49.1.1 `int atoi (const char * s)`

Convert an ASCII string to an integer

Parameters

<i>s</i>	The string to convert
----------	-----------------------

Returns

The integer value of the string, or the MAX/MIN value of an integer if the value is out of range.

4.49.1.2 `int isChar (const char c)`

Checks if the given char is a-z or A-Z

Parameters

<i>const</i>	char c
--------------	--------

Returns

1 if c is a char, 0 otherwise

4.49.1.3 `int isdigit (const char c)`

Determine if a character is a digit.

Parameters

<i>c</i>	The character to check
----------	------------------------

Returns

True if the character is a digit

4.49.1.4 `int isLowerChar (const char c)`

Checks if the given char is a-z

Parameters

<i>const</i>	char c
--------------	--------

Returns

1 if *c* is a lower char, 0 otherwise

4.49.1.5 int isspace (const char * *c*)

Determine if a character is whitespace.

Parameters

<i>c</i>	The character to check
----------	------------------------

Returns

True if the character is a whitespace character

4.49.1.6 int isUpperChar (const char *c*)

Checks if the given char is A-Z

Parameters

<i>const</i>	char <i>c</i>
--------------	---------------

Returns

1 if *c* is a upper char, 0 otherwise

4.49.1.7 void itoa (int *num*, char * *str*, int *base*)

Converts an integer to an ASCII string.

Parameters

<i>num</i>	The number to convert
<i>str</i>	The destination string
<i>base</i>	The radix

4.49.1.8 void reverse (char * *str*, int *len*)

Reverses a string.

Parameters

<i>str</i>	The string to reverse
<i>len</i>	The length of the string

4.49.1.9 char* strcat (char * *s1*, const char * *s2*)

Concatenate the contents of one string onto another.

Parameters

<i>s1</i>	The destination string
<i>s2</i>	The source string

Returns

A pointer to the destination string

4.49.1.10 int strcmp (const char * *s1*, const char * *s2*)

Compares two strings to each other

Parameters

<i>s1</i>	The first string
<i>s2</i>	The second string

Returns

The difference between the characters at the first index of indifference

4.49.1.11 char* strcpy (char * *cpy*, const char * *ori*)

Copy on string to another.

Parameters

<i>cpy</i>	The destination string
<i>ori</i>	The source string

Returns

A pointer to the destination string

4.49.1.12 int strlen (const char * *s*)

Returns the length of a string.

Parameters

s	The input string
---	------------------

Returns

The length of the string

4.49.1.13 char* strtok (char * s1, const char * s2)

Split string into tokens

Call this function multiple times (substituting NULL for s1) until NULL is returned to get all tokens.

Parameters

s1	The string to split
s2	The delimiter

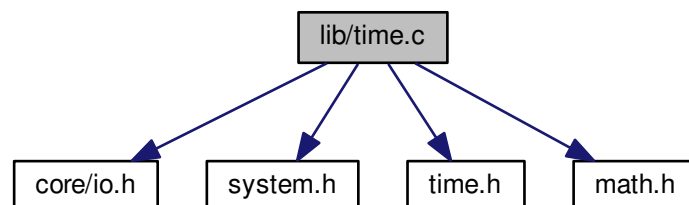
Returns

A single token

4.50 lib/time.c File Reference

```
#include <core/io.h>
#include <system.h>
#include <time.h>
#include <math.h>
```

Include dependency graph for time.c:



Functions

- [date_time getDateTime \(\)](#)

- void `setDateTime` (`date_time` dateTime)
- unsigned char `getSeconds` ()
- unsigned char `getMinutes` ()
- unsigned char `getHours` ()
- unsigned char `getDayOfWeek` ()
- unsigned char `getDayOfMonth` ()
- unsigned char `getMonth` ()
- unsigned char `getYear` ()
- void `setSeconds` (unsigned char sec)
- void `setMinutes` (unsigned char min)
- void `setHours` (unsigned char hour)
- void `setDayOfWeek` (unsigned char day)
- void `setDayOfMonth` (unsigned char day)
- void `setMonth` (unsigned char mon)
- void `setYear` (unsigned char year)
- void `updateDayOfWeek` (`date_time` *dateTime)
- void `updateDayOfYear` (`date_time` *dateTime)
- int `isLeapYear` (int year)

Variables

- const int `DAYS_IN_MONTH` [13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

4.50.1 Function Documentation

4.50.1.1 `date_time` `getDateTime` ()

Gets the date and time from the RTC registers.

Returns

The date and time stored in the RTC.

4.50.1.2 unsigned char `getDayOfMonth` ()

Gets the day of the month (decimal-encoded) from the RTC.

Returns

The decimal-encoded day of the month.

4.50.1.3 unsigned char `getDayOfWeek` ()

Gets the day of the week (decimal-encoded) from the RTC.

Sunday - 1 Monday - 2 Tuesday - 3 Wednesday - 4 Thursday - 5 Friday - 6 Saturday - 7

Returns

The decimal-encoded day of the week.

4.50.1.4 unsigned char getHours ()

Gets the hours value (decimal-encoded) from the RTC.

Returns

The decimal-encoded number of hours.

4.50.1.5 unsigned char getMinutes ()

Gets the minutes value (decimal-encoded) from the RTC.

Returns

The decimal-encoded number of minutes.

4.50.1.6 unsigned char getMonth ()

Gets the month (decimal-encoded) from the RTC.

Returns

The decimal-encoded month.

4.50.1.7 unsigned char getSeconds ()

Gets the seconds value (decimal-encoded) from the RTC.

Returns

The decimal-encoded number of seconds.

4.50.1.8 unsigned char getYear ()

Gets the year (decimal-encoded) from the RTC.

Returns

The decimal-encoded year.

4.50.1.9 int isLeapYear (int *year*)

Determines if the given year is a leap year.

Parameters

<i>year</i>	The year to check
-------------	-------------------

Returns

True if the year is a leap year.

4.50.1.10 void setDateTime (date_time *dateTime*)

Sets the date and time to the specified values.

Day of month must be specified but day of week/year will be automatically calculated.

Parameters

<i>dateTime</i>	The values to set.
-----------------	--------------------

4.50.1.11 void setDayOfMonth (unsigned char *day*)

Sets the day of the month register in the RTC. This number should be decimal-encoded.

Parameters

<i>dayOfMonth</i>	The day of the month value to set
-------------------	-----------------------------------

4.50.1.12 void setDayOfWeek (unsigned char *day*)

Sets the day of the week register in the RTC. This number should be decimal-encoded.

Sunday - 1 Monday - 2 Tuesday - 3 Wednesday - 4 Thursday - 5 Friday - 6 Saturday - 7

Parameters

<i>dayOfWeek</i>	The day of the week value to set
------------------	----------------------------------

4.50.1.13 void setHours (unsigned char *hour*)

Sets the hours register in the RTC. This number should be decimal-encoded.

Parameters

<i>hours</i>	The hours value to set
--------------	------------------------

4.50.1.14 void setMinutes (unsigned char *min*)

Sets the minutes register in the RTC. This number should be decimal-encoded.

Parameters

<i>minutes</i>	The minutes value to set
----------------	--------------------------

4.50.1.15 void setMonth (unsigned char *mon*)

Sets the month register in the RTC. This number should be decimal-encoded.

Parameters

<i>month</i>	The month value to set
--------------	------------------------

4.50.1.16 void setSeconds (unsigned char *sec*)

Sets the seconds register in the RTC. This number should be decimal-encoded.

Parameters

<i>seconds</i>	The seconds value to set
----------------	--------------------------

4.50.1.17 void setYear (unsigned char *year*)

Sets the year register in the RTC. This number should be decimal-encoded.

Parameters

<i>year</i>	The year value to set
-------------	-----------------------

4.50.1.18 void updateDayOfWeek ([date_time](#) * *dateTime*)

Sets the day of week property of the [date_time](#) struct based on the year, month, and day of month values.

Sunday - 1 Monday - 2 Tuesday - 3 Wednesday - 4 Thursday - 5 Friday - 6 Saturday - 7

Parameters

<i>dateTime</i>	The date_time to update.
-----------------	------------------------------------------

4.50.1.19 void updateDayOfYear (date_time * dateTime)

Sets the day of year property of the `date_time` struct based on the year, month, and day of month values.

Parameters

<code>dateTime</code>	The <code>date_time</code> to update.
-----------------------	---------------------------------------

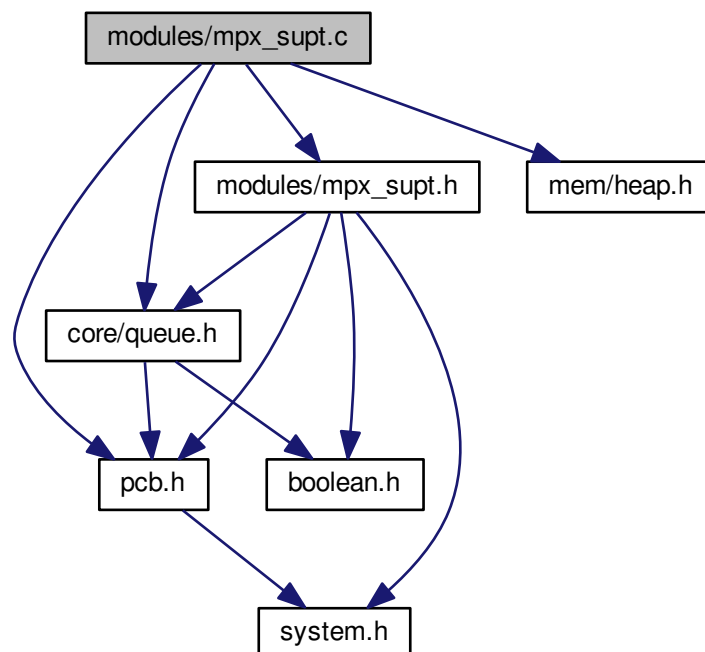
4.50.2 Variable Documentation

4.50.2.1 const int DAYS_IN_MONTH[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

4.51 modules/mpx_supt.c File Reference

```
#include <modules/mpx_supt.h>
#include <mem/heap.h>
#include <core/queue.h>
#include <core/pcb.h>
```

Include dependency graph for `mpx_supt.c`:



Functions

- `u32int * sys_call (context *registers)`
- `void * memset (void *s, int c, size_t n)`
- `int sys_req (int op_code)`
- `void mpx_init (int cur_mod)`
- `void sys_set_malloc (void *(*func)(int))`
- `void sys_set_free (boolean(func)(void *))`
- `void * sys_alloc_mem (u32int size)`
- `int sys_free_mem (void *ptr)`
- `void idle ()`
- `const char * getCOPName ()`

Variables

- `param params`
- `int current_module = -1`
- `void *(* student_malloc)(int)`
- `pcb * cop = NULL`
- `context * callerContext`
- `boolean(* student_free)(void *)`

4.51.1 Function Documentation

4.51.1.1 `const char* getCOPName ()`

Gets the name of the COP

Returns

const char pointer name

4.51.1.2 `void idle ()`

The idle process

4.51.1.3 `void* memset (void * s, int c, size_t n)`

Set a region of memory

Parameters

<i>s</i>	Destination
<i>c</i>	Byte to write
<i>n</i>	Count

Returns

s

4.51.1.4 void mpx_init (int *cur_mod*)

Initialize MPX support software

Parameters

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

4.51.1.5 void* sys_alloc_mem (u32int *size*)

Allocates a block of memory (similar to malloc)

Parameters

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

Returns

The allocated memory

4.51.1.6 u32int* sys_call (context * *registers*)

Changes the currently running process to that of the next ready process

Parameters

<i>registers</i>	- copy of register values
------------------	---------------------------

Returns

u32int position of stackTop

4.51.1.7 int sys_free_mem (void * *ptr*)

Frees memory

Parameters

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

Returns

4.51.1.8 `int sys_req (int op_code)`

Generates interrupt 60H

Parameters

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

Returns

0

4.51.1.9 `void sys_set_free (boolean(func)(void *))`

Sets the memory free function for `sys_free_mem`

Parameters

<i>func</i>	Function pointer to the memory free-er
-------------	----------------------------------------

4.51.1.10 `void sys_set_malloc (void (*)(int) func)`

Sets the memory allocation function for `sys_alloc_mem`

Parameters

<i>func</i>	Function pointer to the memory allocator
-------------	------------------------------------------

4.51.2 Variable Documentation

4.51.2.1 `context* callerContext`

4.51.2.2 `pcb* cop = NULL`

4.51.2.3 `int current_module = -1`

4.51.2.4 `param params`

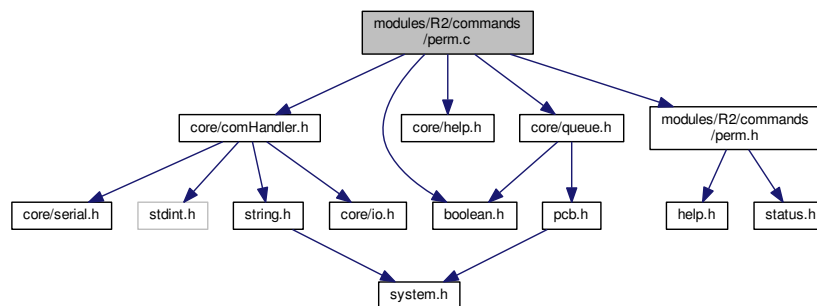
4.51.2.5 `boolean(* student_free) (void *)`

4.51.2.6 void>(* student_malloc)(int)

4.52 modules/R2/commands/perm.c File Reference

```
#include <boolean.h>
#include <core/comHandler.h>
#include <core/help.h>
#include <core/queue.h>
#include <modules/R2/commands/perm.h>
```

Include dependency graph for perm.c:



Functions

- void [printQueueInfo](#) (node *queue)
- void [printPcbInfo](#) (pcb *p)
- void [registerR2PermCommands](#) ()
- const char * [suspendPcb](#) (char **args, int numArgs)
- const char * [resumePcb](#) (char **args, int numArgs)
- const char * [setPriorityPcb](#) (char **args, int numArgs)
- const char * [showPcbInfo](#) (char **args, int numArgs)

4.52.1 Function Documentation

4.52.1.1 void printPcbInfo (pcb * p)

4.52.1.2 void printQueueInfo (node * queue)

4.52.1.3 void registerR2PermCommands ()

Registers the permanent commands in the command handler

4.52.1.4 const char* resumePcb (char ** args, int numArgs)

Places a PCB into the not suspended state and reinserts it into the appropriate queue.

Usage: rpcb name

Args: name - The name of the process to resume (must exist) –all - Resumes all processes

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.52.1.5 const char* setPriorityPcb (char ** args, int numArgs)

Sets a PCB's priority and reinserts the process into the correct place in the correct queue.

Usage: ppcb name priority

Args: name - The name of the process to set the priority on (must exist) priority - The new priority (between 0 and 9)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.52.1.6 const char* showPcbInfo (char ** args, int numArgs)

Displays the following information for the specified PCBs: Process Name: Class: State: Suspended Status↵: Priority:

Usage: showpcb [-all] [-ready] [-blocked] [-suspended] [-name pcbName]

Args: [no args] - Shows the help for this command -all - Displays information for all PCBs -ready - Displays information for ready PCBs -blocked - Displays information for blocked PCBs -suspended - Displays information for suspended PCBs -name - Displays information for the specified PCB

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.52.1.7 `const char* suspendPcb (char ** args, int numArgs)`

Places a PCB into the suspended state and reinserts it into the appropriate queue.

Usage: spcb name

Args: name - The name of the process to suspend (must exist) –all - Resumes all processes

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

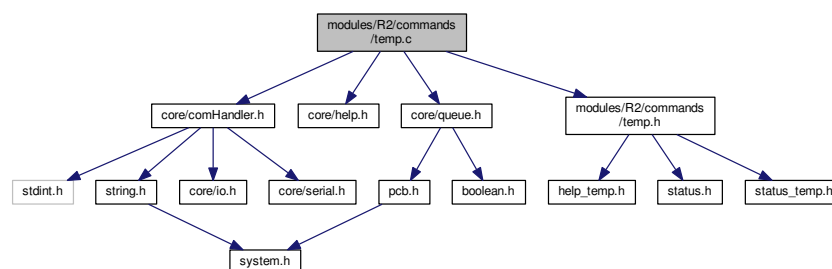
Returns

A status message indicating success/failure

4.53 modules/R2/commands/temp.c File Reference

```
#include <core/comHandler.h>
#include <core/help.h>
#include <core/queue.h>
#include <modules/R2/commands/temp.h>
```

Include dependency graph for temp.c:



Functions

- void [registerR2TempCommands](#) ()
- const char * [createPcb](#) (char **args, int numArgs)
- const char * [deletePcb](#) (char **args, int numArgs)
- const char * [blockPcb](#) (char **args, int numArgs)
- const char * [unblockPcb](#) (char **args, int numArgs)

4.53.1 Function Documentation

4.53.1.1 `const char* blockPcb (char ** args, int numArgs)`

Places a PCB into the blocked state and reinserts it into the appropriate queue.

Note: This command will be removed in module R3/R4

Usage: bpcb name

Args: name - The Process Name to place into the blocked state (must exist)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.53.1.2 `const char* createPcb (char ** args, int numArgs)`

Creates a PCB and inserts it into the appropriate queue.

Note: This command will be removed in module R3/R4

Usage: cpcb name class priority

Args: name - The Process Name (must be unique) class - The Process Class (either 0 (system) or 1 (application))
priority - The Process Priority (number between 0 and 9)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.53.1.3 `const char* deletePcb (char ** args, int numArgs)`

Removes a PCB from the appropriate queue and then frees all associated memory.

Note: This command will be removed in module R3/R4

Usage: dpcb name

Args: name - The Process Name to remove (must exist)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

Returns

A status message indicating success/failure

4.53.1.4 void registerR2TempCommands ()

Registers the temporary commands in the command handler

4.53.1.5 const char* unblockPcb (char ** args, int numArgs)

Places a PCB into the unblocked state and reinserts it into the appropriate queue.

Note: This command will be removed in module R3/R4

Usage: upcb name

Args: name - The Process Name to place into the unblocked state (must exist)

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

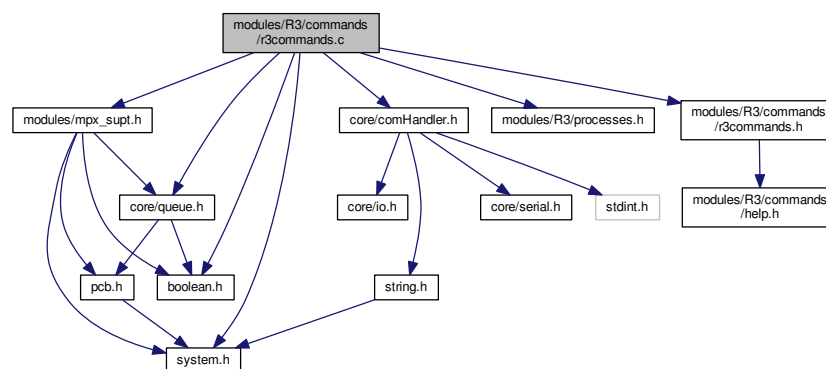
Returns

A status message indicating success/failure

4.54 modules/R3/commands/r3commands.c File Reference

```
#include <system.h>
#include <boolean.h>
#include <core/queue.h>
#include <core/comHandler.h>
#include <modules/R3/processes.h>
#include <modules/R3/commands/r3commands.h>
#include <modules/mpx_supt.h>
```

Include dependency graph for r3commands.c:



Macros

- `#define P1_NAME ((const char*) "r3p1")`
- `#define P2_NAME ((const char*) "r3p2")`
- `#define P3_NAME ((const char*) "r3p3")`
- `#define P4_NAME ((const char*) "r3p4")`
- `#define P5_NAME ((const char*) "r3p5")`

Functions

- `void registerR3Commands ()`
- `const char * yield (char **args, int numArgs)`
- `const char * loadr3 (char **args, int numArgs)`

4.54.1 Macro Definition Documentation

4.54.1.1 `#define P1_NAME ((const char*) "r3p1")`

4.54.1.2 `#define P2_NAME ((const char*) "r3p2")`

4.54.1.3 `#define P3_NAME ((const char*) "r3p3")`

4.54.1.4 `#define P4_NAME ((const char*) "r3p4")`

4.54.1.5 `#define P5_NAME ((const char*) "r3p5")`

4.54.2 Function Documentation

4.54.2.1 `const char* loadr3 (char ** args, int numArgs)`

Loads the r3 processes to the queue.
")

Usage: loadr3

Args: [no args] - loads processes

Parameters

<i>args</i>	The arguments to pass to the function
-------------	---------------------------------------

Returns

""

4.54.2.2 `void registerR3Commands ()`

Registers commands in command handler

4.54.2.3 `const char* yield (char ** args, int numArgs)`

Yields command handler to allow other processes to run.

Usage: yield

Args: [no args] - yields command handler

Parameters

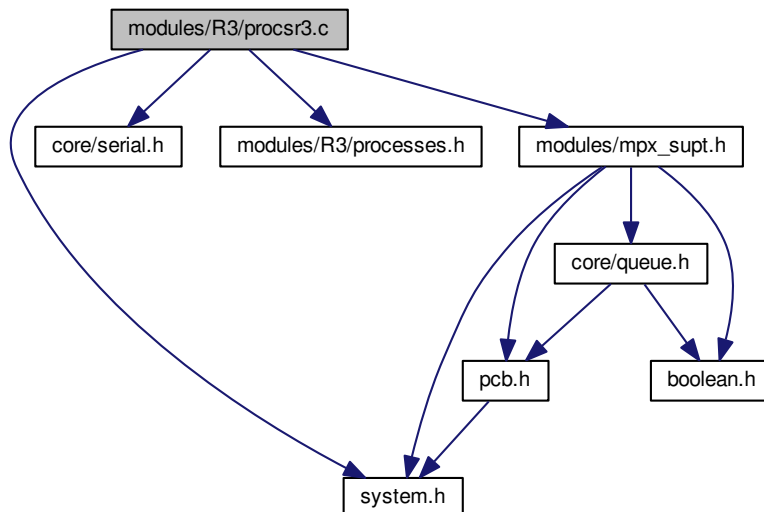
<code>args</code>	The arguments to pass to the function
-------------------	---------------------------------------

Returns

""

4.55 modules/R3/procsr3.c File Reference

```
#include <system.h>
#include <core/serial.h>
#include <modules/R3/processes.h>
#include <modules/mpx_supt.h>
Include dependency graph for procsr3.c:
```



Macros

- `#define RC_1 1`
- `#define RC_2 2`
- `#define RC_3 3`
- `#define RC_4 4`
- `#define RC_5 5`

Functions

- void [proc1](#) ()
- void [proc2](#) ()
- void [proc3](#) ()
- void [proc4](#) ()
- void [proc5](#) ()

4.55.1 Macro Definition Documentation

4.55.1.1 `#define RC_1 1`

4.55.1.2 `#define RC_2 2`

4.55.1.3 `#define RC_3 3`

4.55.1.4 `#define RC_4 4`

4.55.1.5 `#define RC_5 5`

4.55.2 Function Documentation

4.55.2.1 `void proc1 ()`

4.55.2.2 `void proc2 ()`

4.55.2.3 `void proc3 ()`

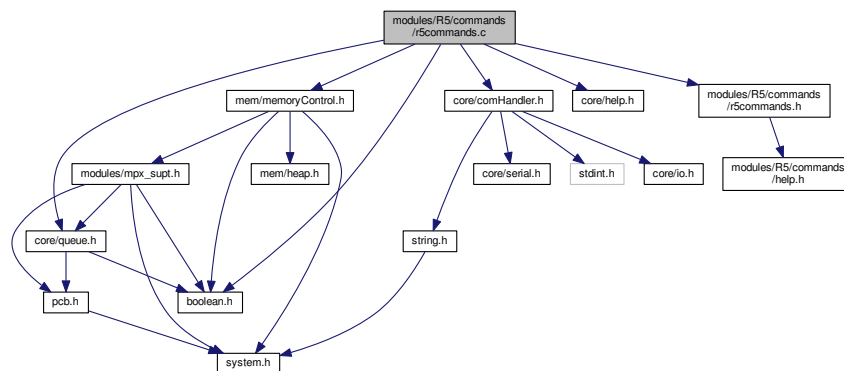
4.55.2.4 `void proc4 ()`

4.55.2.5 `void proc5 ()`

4.56 modules/R5/commands/r5commands.c File Reference

```
#include <boolean.h>
#include <core/comHandler.h>
#include <core/help.h>
#include <core/queue.h>
#include <modules/R5/commands/r5commands.h>
#include <mem/memoryControl.h>
```

Include dependency graph for `r5commands.c`:



Functions

- void [printBlockInfo](#) ([cmcb](#) *blockList)
- void [printCmcbInfo](#) ([cmcb](#) *block)
- void [registerR5PermCommands](#) ()
- const char * [showMemory](#) (char **args, int numArgs)

4.56.1 Function Documentation

4.56.1.1 void [printBlockInfo](#) ([cmcb](#) * *blockList*)

4.56.1.2 void [printCmcbInfo](#) ([cmcb](#) * *block*)

4.56.1.3 void [registerR5PermCommands](#) ()

Registers the permanent commands in the command handler

4.56.1.4 const char* [showMemory](#) (char ** *args*, int *numArgs*)

Displays the following information for the specified CMCB's: CMCB Type: Begining Memory Address: Block Size: Memory Size: Process Name:

Usage: [showMemory](#) [-all] [-free] [-allocated]

Args: [no args] - Shows the help for this command -all - Displays both free and allocated memory -free - Displays free memory -allocated - Displays allocated memory

Parameters

<i>args</i>	The arguments to pass to the function
<i>numArgs</i>	The number of arguments

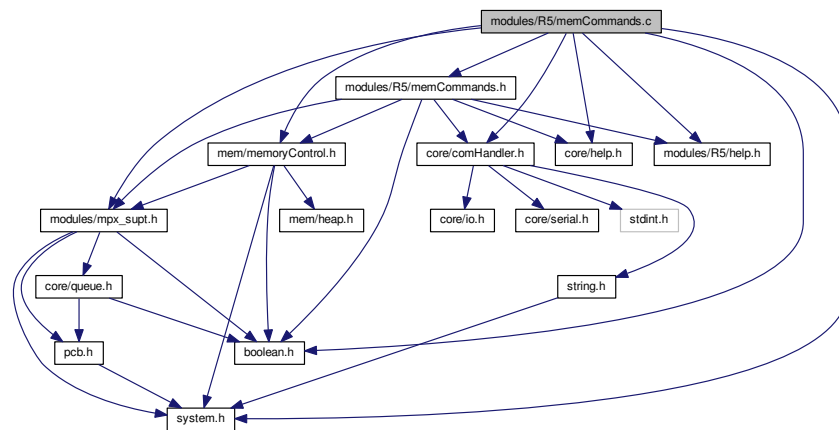
Returns

A status message indicating success/failure

4.57 modules/R5/memCommands.c File Reference

```
#include <mem/memoryControl.h>
#include <boolean.h>
#include <core/comHandler.h>
#include <core/help.h>
#include <modules/mpx_supt.h>
#include <modules/R5/memCommands.h>
#include <modules/R5/help.h>
#include <system.h>
```

Include dependency graph for memCommands.c:



Functions

- void [registerR5TempCommands](#) ()
- const char * [initHeap](#) (char **args, int numArgs)
- const char * [allocateMem](#) (char **args, int numArgs)
- const char * [freeMemory](#) (char **args, int numArgs)
- const char * [isEmptyCom](#) (char **args, int numArgs)

4.57.1 Function Documentation

4.57.1.1 const char* allocateMem (char ** args, int numArgs)

Allocates a memory block if enough memory is available

Parameters

<i>size</i>	- size of memory to allocate in bytes
-------------	---------------------------------------

Returns

pointer to the me

4.57.1.2 const char* freeMemory (char ** args, int numArgs)

Deallocates the block of memory at the mempointer

Parameters

<i>memPointer</i>	- pointer to the mem block
-------------------	----------------------------

4.57.1.3 `const char* initHeap (char ** args, int numArgs)`

Initializes the heap to the provided size and creates a free mem block across it

Returns

True or false

4.57.1.4 `const char* isEmptyCom (char ** args, int numArgs)`

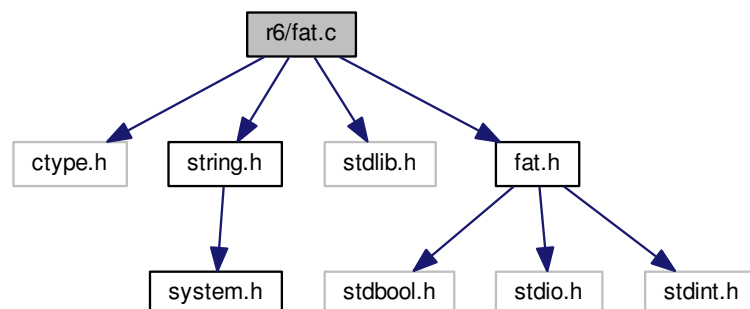
Check if memory is empty

4.57.1.5 `void registerR5TempCommands ()`

Registers the permanent commands in the command handler

4.58 r6/fat.c File Reference

```
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#include "fat.h"
Include dependency graph for fat.c:
```



Functions

- void [_loadBootSectorInfo](#) ()
- void [_loadFATTables](#) ()
- void [_loadRootDirectory](#) ()
- [dir_entry](#) * [_loadSectorAsDirectoryEntries](#) (uint16_t sector)
- void [_readDirectoryEntry](#) ([dir_entry](#) *dir)
- long [_getDiskOffsetForDirEntry](#) (int idx)
- void [_saveFATTables](#) ()
- void [_saveDirEntry](#) ([dir_entry](#) *dir)
- void [_refreshDirectory](#) ()
- void [_prepNewDirSector](#) (uint16_t sector)
- int [_getFirstFreeIndexInSector](#) ([dir_entry](#) *dirs)
- int [_getFirstFreeIndexInDirs](#) ([dir_entry](#) *dirs, int maxSize)
- uint16_t [_getFirstOpenSector](#) ()
- void [initialize](#) (FILE *diskImage)
- void [destroy](#) ()
- [boot_sector](#) * [getBootSector](#) ()
- [fat_tables](#) * [getFATTables](#) ()
- [dir_entry](#) * [getCurrentDirectory](#) ()
- int [getCurrentDirectoryMaxSize](#) ()
- unsigned char * [getFileFromSector](#) (uint16_t sector, int size)
- int [changeToDirectory](#) (uint16_t sector)
- int [changeToParentDirectory](#) ()
- void [setFilename](#) (int idx, const char *filename, const char *fileExt)
- bool [moveFile](#) (int idx, uint16_t destSector)

Variables

- FILE * [_DiskImage](#)
- [boot_sector](#) [_BootSector](#)
- [fat_tables](#) [_FATTables](#)
- [dir_entry](#) * [_CurrentDirectory](#)
- int [_CurrDirSize](#) = 0
- bool [_isCurrentRoot](#) = false

4.58.1 Function Documentation

4.58.1.1 long [_getDiskOffsetForDirEntry](#) (int *idx*)

Gets the disk offset (in bytes) of the file at the given index of the current directory.

Parameters

<i>idx</i>	The index of the current directory
------------	------------------------------------

Returns

The disk offset in byte

4.58.1.2 `int _getFirstFreeIndexInDirs (dir_entry * dirs, int maxSize)`

Gets the first free directory entry index in the given array of entries. Supports a variable max size.

Parameters

<i>dirs</i>	The array on entries to search through
<i>maxSize</i>	The size of the array

Returns

The index of the first free directory entry

4.58.1.3 `int _getFirstFreeIndexInSector (dir_entry * dirs)`

Gets the first free directory entry index in the given array of entries.

Parameters

<i>dirs</i>	The array on entries to search through
-------------	----------------------------------------

Returns

The index of the first free directory entry

4.58.1.4 `uint16_t _getFirstOpenSector ()`

Gets the first open sector.

Returns

The ID of the first empty sector

4.58.1.5 `void _loadBootSectorInfo ()`

Loads boot sector information.

4.58.1.6 `void _loadFATTables ()`

Loads the FAT table information

4.58.1.7 `void _loadRootDirectroy ()`

Loads the root directory as the current directory.

4.58.1.8 `dir_entry * _loadSectorAsDirectoryEntries (uint16_t sector)`

Loads a sector as an array of (bytesPerSector / DIR_ENTRY_SIZE) directory entries.

Parameters

<i>sector</i>	The sector to load from.
---------------	--------------------------

Returns

An array of directory entries.

4.58.1.9 void _prepNewDirSector (uint16_t *sector*)

Preps the given sector to be a new directory sector.

Parameters

<i>sector</i>	The sector ID to set
---------------	----------------------

4.58.1.10 void _readDirectoryEntry (dir_entry * *dir*)

Reads a single directory entry into the passed entry.

Parameters

<i>The</i>	directory entry to store the information in
------------	---------------------------------------------

4.58.1.11 void _refreshDirectory ()

Refreshes (reloads) the current directory.

4.58.1.12 void _saveDirEntry (dir_entry * *dir*)

Saves the passed directory entry to the disc.

4.58.1.13 void _saveFATTables ()

Saves the FAT Tables to the disc.

4.58.1.14 int changeToDirectory (uint16_t *sector*)

Changes to the directory specified by the given logical sector.

Parameters

<code>sector</code>	The starting logical sector of the directory.
---------------------	-----------------------------------------------

Returns

The maximum number of entries that can appear in this directory.

4.58.1.15 `int changeToParentDirectory ()`

Changes the current directory to the parent of the current.

Returns

The maximum size of the new current directory, or -1 if the directory didn't switch.

4.58.1.16 `void destroy ()`

Destroys the FAT abstraction, freeing any memory used internally.

4.58.1.17 `boot_sector* getBootSector ()`

Gets the boot sector of the FAT File System.

Returns

The boot sector of the FAT File System

4.58.1.18 `dir_entry* getCurrentDirectory ()`

Gets the current directory of the FAT File System.

Returns

The current directory as an array of directory entries.

4.58.1.19 `int getCurrentDirectoryMaxSize ()`

Gets the maximum size of the current directory.

Returns

The maximum size of the current directory.

4.58.1.20 `fat_tables* getFATTables ()`

Gets the FAT Tables for the FAT File System.

Returns

The FAT Tables for the FAT File System

4.58.1.21 `unsigned char* getFileFromSector (uint16_t sector, int size)`

Gets the file specified by the given sector.

Parameters

<i>sector</i>	The starting logical sector of the file
<i>size</i>	The size of the file (in bytes)

Returns

An array of bytes representing the file.

4.58.1.22 void initialize (FILE * *diskImage*)

Initializes the FAT abstraction with the given disk image.

Parameters

<i>diskImage</i>	The pointer to an opened file that is a FAT12 disk image.
------------------	-----------------------------------------------------------

4.58.1.23 bool moveFile (int *idx*, uint16_t *destSector*)

Moves a file at the specified index of the current directory to the directory at the destination sector.

Parameters

<i>idx</i>	The index of the listing in the current direcorey
<i>destSector</i>	The beginning sector of the target directory

4.58.1.24 void setFilename (int *idx*, const char * *filename*, const char * *fileExt*)

Sets the name of the file at the specified index of the current list of directory entries.

Parameters

<i>idx</i>	The index of the listing in the current directory
<i>filename</i>	The name of the file (max 8 characters)
<i>fileExt</i>	The extension of the file (max 3 characters)

4.58.2 Variable Documentation

4.58.2.1 boot_sector _BootSector

4.58.2.2 int _CurrDirSize = 0

4.58.2.3 `dir_entry* _CurrentDirectory`

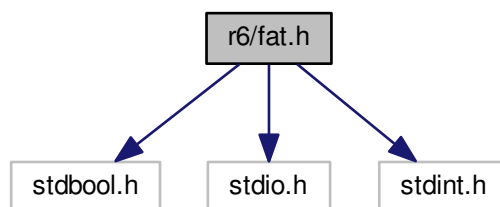
4.58.2.4 `FILE* _DiskImage`

4.58.2.5 `fat_tables _FATTables`

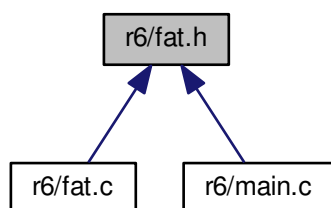
4.58.2.6 `bool _isCurrentRoot = false`

4.59 r6/fat.h File Reference

```
#include <stdbool.h>
#include <stdio.h>
#include <stdint.h>
Include dependency graph for fat.h:
```



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



Data Structures

- struct [boot_sector](#)
- struct [dir_entry](#)
- struct [fat_tables](#)

Macros

- `#define BOOT_SECTOR_OFFSET 0`
- `#define FAT1_OFFSET 1`
- `#define FAT2_OFFSET 10`
- `#define ROOT_DIRECTORY_OFFSET 19`
- `#define DATA_AREA_OFFSET 33`
- `#define UNUSED 0x00`
- `#define RESERVED_CLUSTER_BEGIN 0xFF0`
- `#define RESERVED_CLUSTER_END 0xFF6`
- `#define BAD_CLUSTER 0xFF7`
- `#define LAST_CLUSTER_BEGIN 0xFF8`
- `#define LAST_CLUSTER_END 0xFF`
- `#define READ_ONLY 0x01`
- `#define HIDDEN 0x02`
- `#define SYSTEM 0x04`
- `#define VOLUME_LABEL 0x08`
- `#define SUBDIRECTORY 0x10`
- `#define ARCHIVE 0x20`
- `#define DIR_ENTRY_SIZE 32`
- `#define MAX_FILENAME_LENGTH 8`
- `#define MAX_EXT_LENGTH 3`
- `#define DELETED 0xE5`
- `#define REMAINING_FREE 0x00`

Functions

- `void initialize (FILE *diskImage)`
- `void destroy ()`
- `boot_sector * getBootSector ()`
- `fat_tables * getFATTables ()`
- `dir_entry * getCurrentDirectory ()`
- `int getCurrentDirectoryMaxSize ()`
- `unsigned char * getFileFromSector (uint16_t cluster, int size)`
- `int changeToDirectory (uint16_t cluster)`
- `int changeToParentDirectory ()`
- `void setFilename (int idx, const char *filename, const char *fileExt)`
- `bool moveFile (int idx, uint16_t destSector)`

4.59.1 Macro Definition Documentation

4.59.1.1 `#define ARCHIVE 0x20`

4.59.1.2 `#define BAD_CLUSTER 0xFF7`

4.59.1.3 `#define BOOT_SECTOR_OFFSET 0`

4.59.1.4 `#define DATA_AREA_OFFSET 33`

4.59.1.5 `#define DELETED 0xE5`

4.59.1.6 `#define DIR_ENTRY_SIZE 32`

4.59.1.7 `#define FAT1_OFFSET 1`

4.59.1.8 `#define FAT2_OFFSET 10`

4.59.1.9 `#define HIDDEN 0x02`

4.59.1.10 `#define LAST_CLUSTER_BEGIN 0xFF8`

4.59.1.11 `#define LAST_CLUSTER_END 0xFFF`

4.59.1.12 `#define MAX_EXT_LENGTH 3`

4.59.1.13 `#define MAX_FILENAME_LENGTH 8`

4.59.1.14 `#define READ_ONLY 0x01`

4.59.1.15 `#define REMAINING_FREE 0x00`

4.59.1.16 `#define RESERVED_CLUSTER_BEGIN 0xFF0`

4.59.1.17 `#define RESERVED_CLUSTER_END 0xFF6`

4.59.1.18 `#define ROOT_DIRECTORY_OFFSET 19`

4.59.1.19 `#define SUBDIRECTORY 0x10`

4.59.1.20 `#define SYSTEM 0x04`

4.59.1.21 `#define UNUSED 0x00`

4.59.1.22 `#define VOLUME_LABEL 0x08`

4.59.2 Function Documentation

4.59.2.1 `int changeToDirectory (uint16_t sector)`

Changes to the directory specified by the given logical cluster.

Parameters

<i>cluster</i>	The starting logical cluster of the directory.
----------------	------------------------------------------------

Returns

The maximum number of entries that can appear in this directory.

Changes to the directory specified by the given logical sector.

Parameters

<i>sector</i>	The starting logical sector of the directory.
---------------	-----------------------------------------------

Returns

The maximum number of entries that can appear in this directory.

4.59.2.2 int changeToParentDirectory ()

Changes the current directory to the parent of the current.

Returns

The maximum size of the new current directory.

Changes the current directory to the parent of the current.

Returns

The maximum size of the new current directory, or -1 if the directory didn't switch.

4.59.2.3 void destroy ()

Destroys the FAT abstraction, freeing any memory used internally.

4.59.2.4 boot_sector* getBootSector ()

Gets the boot sector of the FAT File System.

Returns

The boot sector of the FAT File System

4.59.2.5 dir_entry* getCurrentDirectory ()

Gets the current directory of the FAT File System.

Returns

The current directory as an array of directory entries.

4.59.2.6 `int getCurrentDirectoryMaxSize ()`

Gets the maximum size of the current directory.

Returns

The maximum size of the current directory.

4.59.2.7 `fat_tables* getFATTables ()`

Gets the FAT Tables for the FAT File System.

Returns

The FAT Tables for the FAT File System

4.59.2.8 `unsigned char* getFileFromSector (uint16_t sector, int size)`

Gets the file specified by the given cluster.

Parameters

<i>cluster</i>	The starting logical cluster of the file
<i>size</i>	The size of the file (in bytes)

Returns

An array of bytes representing the file.

Gets the file specified by the given sector.

Parameters

<i>sector</i>	The starting logical sector of the file
<i>size</i>	The size of the file (in bytes)

Returns

An array of bytes representing the file.

4.59.2.9 `void initialize (FILE * diskImage)`

Initializes the FAT abstraction with the given disk image.

Parameters

<i>diskImage</i>	The pointer to an opened file that is a FAT12 disk image.
------------------	-----------------------------------------------------------

4.59.2.10 `bool moveFile (int idx, uint16_t destSector)`

Moves a file at the specified index of the current directory to the directory at the destination sector.

Parameters

<i>idx</i>	The index of the listing in the current direcorey
<i>destSector</i>	The beginning sector of the target directory

4.59.2.11 `void setFilename (int idx, const char * filename, const char * fileExt)`

Sets the name of the file at the specified index of the current list of directory entries.

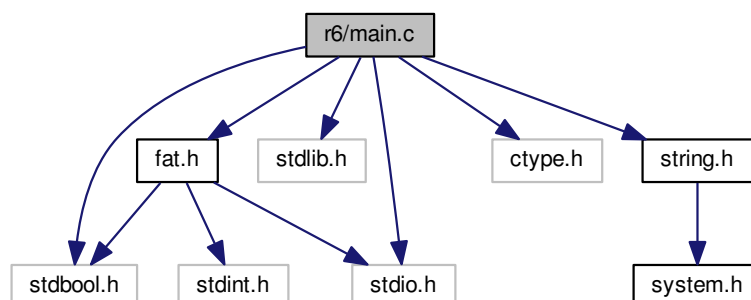
Parameters

<i>idx</i>	The index of the listing in the current directory
<i>filename</i>	The name of the file (max 8 characters)
<i>fileExt</i>	The extension of the file (max 3 characters)

4.60 `r6/main.c` File Reference

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "fat.h"
```

Include dependency graph for main.c:



Functions

- void [_launchCommandInterface](#) ()
- void [_printBootSectorInfo](#) ()
- void [_printFATTableInfo](#) ()
- void [_printDirectoryEntries](#) ([dir_entry](#) *entries, int maxEntries)
- void [_printDirectoryEntriesByType](#) ([dir_entry](#) *entries, int maxEntries, char *ext)
- void [_printDirectoryEntriesByFileName](#) ([dir_entry](#) *entries, int maxEntries, char *name, char *fileExt)
- int [_getClusterOfFileWithName](#) (const char *name, const char *ext)
- int [_getSizeOfFileWithName](#) (const char *name, const char *ext)
- void [_printFile](#) (uint16_t sector, int fileSize, bool pag)
- int [_getIndexOfFileWithName](#) (const char *name, const char *ext)
- bool [_fncmp](#) (const char *n1, const char *n2)
- bool [_extcmp](#) (const char *n1, const char *n2)
- bool [_nameCmpHelper](#) (const char *n1, const char *n2, int maxElements)
- void [_callCommand](#) (char *command)
- int [main](#) (int numArgs, char *args[])

Variables

- char * [paths](#) [256]
- int [depth](#) = 0
- const char * [imageName](#) = ""
- bool [printFileFlag](#) = false
- char * [filename](#) = ""
- FILE * [diskImage](#)

4.60.1 Function Documentation

4.60.1.1 void [_callCommand](#) (char * *command*)

Handles parsing of command and calling of correct operations

4.60.1.2 bool [_extcmp](#) (const char * *n1*, const char * *n2*)

Compares the extension with the given extension to determine equality. // TODO: Words are hard

Parameters

<i>n1</i>	The first name of the comparison (no null terminator)
<i>n2</i>	The second name of the comparison (null terminated)

Returns

True, if the names are equal

4.60.1.3 bool [_fncmp](#) (const char * *n1*, const char * *n2*)

Compares the filename with the given filename to determine equality. // TODO: Words are hard

Parameters

<i>n1</i>	The first name of the comparison (no null terminator)
<i>n2</i>	The second name of the comparison (null terminated)

Returns

True, if the names are equal

4.60.1.4 `int _getClusterOfFileWithName (const char * name, const char * ext)`

Gets the cluster of the file with the given name

4.60.1.5 `int _getIndexOfFileWithName (const char * name, const char * ext)`

Gets the index of the file with the specified name and extension

4.60.1.6 `int _getSizeOfFileWithName (const char * name, const char * ext)`

Gets the size of the file with the given name

4.60.1.7 `void _launchCommandInterface ()`

Starts the interactive shell session.

4.60.1.8 `bool _nameCmpHelper (const char * n1, const char * n2, int maxElements)`

Compares two string name with up to 'maxElements' number of characters. // TODO: Words are hard

Parameters

<i>n1</i>	The first name of the comparison (no null terminator)
<i>n2</i>	The second name of the comparison (null terminated)
<i>maxElements</i>	The number of elements to check

Returns

True, if the names are equal

4.60.1.9 `void _printBootSectorInfo ()`

Prints information for the boot sector

4.60.1.10 void _printDirectoryEntries (*dir_entry* * *entries*, int *maxEntries*)

Prints all directory entries in current dir

4.60.1.11 void _printDirectoryEntriesByFileName (*dir_entry* * *entries*, int *maxEntries*, char * *name*, char * *fileExt*)

Prints any files with given name

4.60.1.12 void _printDirectoryEntriesByType (*dir_entry* * *entries*, int *maxEntries*, char * *ext*)

Prints all directory entries in the current dir with the provided extension

4.60.1.13 void _printFATTableInfo ()

Prints fat table information

4.60.1.14 void _printFile (uint16_t *sector*, int *fileSize*, bool *pag*)

Prints contents of given file

4.60.1.15 int main (int *numArgs*, char * *args*[])

4.60.2 Variable Documentation

4.60.2.1 int *depth* = 0

4.60.2.2 FILE* *diskImage*

4.60.2.3 char* *filename* = ""

4.60.2.4 const char* *imageName* = ""

4.60.2.5 char* *paths*[256]

4.60.2.6 bool *printFileFlag* = false

Index

- `_BootSector`
 - `fat.c`, [162](#)
- `_CurrDirSize`
 - `fat.c`, [162](#)
- `_CurrentDirectory`
 - `fat.c`, [162](#)
- `_DiskImage`
 - `fat.c`, [163](#)
- `_FATTables`
 - `fat.c`, [163](#)
- `__attribute__`
 - `heap.h`, [54](#)
 - `tables.h`, [49](#)
- `__end`
 - `heap.c`, [124](#)
- `_callCommand`
 - `main.c`, [169](#)
- `_end`
 - `heap.c`, [124](#)
- `_extcmp`
 - `main.c`, [169](#)
- `_findNode`
 - `queue.c`, [112](#)
- `_findNodeInQueue`
 - `queue.c`, [112](#)
- `_fncmp`
 - `main.c`, [169](#)
- `_getClusterOfFileWithName`
 - `main.c`, [170](#)
- `_getDiskOffsetForDirEntry`
 - `fat.c`, [158](#)
- `_getFirstFreeIndexInDirs`
 - `fat.c`, [158](#)
- `_getFirstFreeIndexInSector`
 - `fat.c`, [159](#)
- `_getFirstOpenSector`
 - `fat.c`, [159](#)
- `_getIndexOfFileWithName`
 - `main.c`, [170](#)
- `_getSizeOfFileWithName`
 - `main.c`, [170](#)
- `_insertFIFO`
 - `queue.c`, [113](#)
- `_insertPriority`
 - `queue.c`, [113](#)
- `_isCurrentRoot`
 - `fat.c`, [163](#)
- `_kmalloc`
 - `heap.c`, [122](#)
- `heap.h`, [53](#)
- `_launchCommandInterface`
 - `main.c`, [170](#)
- `_loadBootSectorInfo`
 - `fat.c`, [159](#)
- `_loadFATTables`
 - `fat.c`, [159](#)
- `_loadRootDirectory`
 - `fat.c`, [159](#)
- `_loadSectorAsDirectoryEntries`
 - `fat.c`, [159](#)
- `_mergeAdjacentFree`
 - `memoryControl.c`, [125](#)
- `_nameCmpHelper`
 - `main.c`, [170](#)
- `_newNode`
 - `queue.c`, [113](#)
- `_placeStructs`
 - `memoryControl.c`, [125](#)
- `_prepNewDirSector`
 - `fat.c`, [160](#)
- `_printBootSectorInfo`
 - `main.c`, [170](#)
- `_printDirectoryEntries`
 - `main.c`, [170](#)
- `_printDirectoryEntriesByFileName`
 - `main.c`, [171](#)
- `_printDirectoryEntriesByType`
 - `main.c`, [171](#)
- `_printFATTableInfo`
 - `main.c`, [171](#)
- `_printFile`
 - `main.c`, [171](#)
- `_readDirectoryEntry`
 - `fat.c`, [160](#)
- `_refreshDirectory`
 - `fat.c`, [160](#)
- `_saveDirEntry`
 - `fat.c`, [160](#)
- `_saveFATTables`
 - `fat.c`, [160](#)
- ALLOCATED
 - `memoryControl.h`, [56](#)
- APPLICATION
 - `pcb.h`, [40](#)
- APR
 - `time.h`, [92](#)
- ARCHIVE
 - `fat.h`, [164](#)

AUG
 time.h, 92
 access
 gdt_entry_struct, 13
 tables.h, 50
 accessed
 page_entry, 20
 addComHistory
 comHandler.c, 99
 comHandler.h, 25
 addFunctionDef
 comHandler.c, 99
 comHandler.h, 25
 alloc
 heap.c, 123
 heap.h, 53
 allocateMem
 memCommands.c, 156
 memCommands.h, 82
 allocateMemory
 memoryControl.c, 125
 memoryControl.h, 56
 allocatePCB
 pcb.c, 109
 pcb.h, 40
 allocatedHead
 memoryControl.c, 127
 asm
 system.h, 89
 atoi
 string.c, 134
 string.h, 84
 attributes
 dir_entry, 10

 BAD_CLUSTER
 fat.h, 164
 BLOCK_PCB_SUCCESS
 status_temp.h, 73
 BLOCKED
 pcb.h, 40
 BOOT_SECTOR_OFFSET
 fat.h, 164
 base
 gdt_descriptor_struct, 13
 heap, 15
 idt_struct, 16
 tables.h, 50
 base_high
 gdt_entry_struct, 13
 idt_entry_struct, 15
 tables.h, 50
 base_low
 gdt_entry_struct, 13
 idt_entry_struct, 15
 tables.h, 50
 base_mid
 gdt_entry_struct, 13
 tables.h, 50

 bcdToDec
 math.c, 131
 math.h, 51
 beginningAddr
 cmcb, 7
 block
 index_entry, 16
 blockPcb
 temp.c, 149
 temp.h, 75
 boolean
 boolean.h, 23
 boolean.h
 boolean, 23
 false, 23
 true, 23
 boot_sector, 5
 bootSig, 6
 bytesPerSector, 6
 fileSystemType, 6
 ignore_1, 6
 ignore_2, 6
 ignore_3, 6
 ignore_4, 6
 ignore_5, 6
 maxRootDirEntries, 6
 numFATCopies, 6
 numHeads, 6
 numReservedSectors, 6
 numSectors, 6
 sectorCountFAT32, 6
 sectorsPerCluster, 6
 sectorsPerFAT, 6
 sectorsPerTrack, 6
 volId, 6
 volName, 6
 bootSig
 boot_sector, 6
 bounds
 interrupts.c, 106
 breakpoint
 interrupts.c, 106
 buffer
 comHandler.c, 102
 bytesPerSector
 boot_sector, 6

 COM1
 serial.h, 46
 COM2
 serial.h, 46
 COM3
 serial.h, 46
 COM4
 serial.h, 46
 CONTROL_PORT
 time.h, 92
 CREATE_PCB_SUCCESS
 status_temp.h, 73

- callerContext
 - mpx_supt.c, 145
- cdir
 - paging.c, 130
- changeToDirectory
 - fat.c, 160
 - fat.h, 165
- changeToParentDirectory
 - fat.c, 161
 - fat.h, 166
- checkParamClass
 - pcb.c, 109
 - pcb.h, 40
- checkParamName
 - pcb.c, 110
 - pcb.h, 41
- checkParamPriority
 - pcb.c, 110
 - pcb.h, 41
- clear_bit
 - paging.c, 128
 - paging.h, 59
- cli
 - system.h, 89
- cmcb, 7
 - beginningAddr, 7
 - memSize, 7
 - memoryControl.h, 56
 - name, 7
 - next, 7
 - prev, 7
 - size, 7
 - type, 7
- comHandler.c
 - addComHistory, 99
 - addFunctionDef, 99
 - buffer, 102
 - comHistory, 102
 - comHistoryPos, 102
 - continueHandle, 102
 - eraseCurrentRow, 99
 - executeCommand, 100
 - functionDefs, 102
 - functionInsertPoint, 102
 - getComHistory, 100
 - getFunctionDef, 100
 - getHelpString, 100
 - getInput, 101
 - help, 101
 - initCommandHandler, 101
 - printStart, 101
 - returnToInsertionPoint, 101
 - setupCommands, 102
 - shutdown, 102
- comHandler.h
 - addComHistory, 25
 - addFunctionDef, 25
 - eraseCurrentRow, 26
 - executeCommand, 26
 - getComHistory, 26
 - getFunctionDef, 26
 - getHelpString, 27
 - getInput, 27
 - initCommandHandler, 27
 - printStart, 27
 - returnToInsertionPoint, 27
 - setupCommands, 27
- comHistory
 - comHandler.c, 102
- comHistoryPos
 - comHandler.c, 102
- commands.c
 - date, 103
 - version, 103
- commands.h
 - date, 28
 - help, 29
 - shutdown, 29
 - version, 30
- context, 8
 - cs, 8
 - ds, 8
 - eax, 8
 - ebp, 8
 - ebx, 8
 - ecx, 8
 - edi, 8
 - edx, 8
 - eflags, 8
 - eip, 8
 - es, 8
 - esi, 8
 - esp, 8
 - fs, 8
 - gs, 8
 - mpx_supt.h, 63
- continueHandle
 - comHandler.c, 102
- cop
 - mpx_supt.c, 145
- coprocessor
 - interrupts.c, 106
- coprocessor_segment
 - interrupts.c, 106
- core/help.h
 - HELP_COMMAND_DATE, 31
 - HELP_COMMAND_HELP, 31
 - HELP_COMMAND_SHUTDOWN, 31
 - HELP_COMMAND_VERSION, 31
 - HELP_INVALID_ARGUMENTS, 32
 - HELP_UNKNOWN_COMMAND, 32
- createPcb
 - temp.c, 150
 - temp.h, 75
- creationDate
 - dir_entry, 10

- creationTime
 - dir_entry, 10
- cs
 - context, 8
- curr_heap
 - heap.c, 124
- current_module
 - mpx_supt.c, 145
- DATA_AREA_OFFSET
 - fat.h, 164
- DATA_PORT
 - time.h, 92
- DAY_MONTH
 - time.h, 92
- DAY_WEEK
 - time.h, 92
- DAYS_IN_MONTH
 - time.c, 142
 - time.h, 98
- DELETE_PCB_SUCCESS
 - status_temp.h, 73
- DELETED
 - fat.h, 164
- DEC
 - time.h, 92
- DIR_ENTRY_SIZE
 - fat.h, 164
- data
 - node, 19
- date
 - commands.c, 103
 - commands.h, 28
- date_time, 9
 - day_m, 9
 - day_w, 9
 - day_y, 9
 - hour, 9
 - min, 9
 - mon, 9
 - sec, 9
 - year, 9
- day_m
 - date_time, 9
- day_w
 - date_time, 9
- day_y
 - date_time, 9
- deallocateMemory
 - memoryControl.c, 126
 - memoryControl.h, 56
- debug
 - interrupts.c, 106
- decToBcd
 - math.c, 131
 - math.h, 51
- deletePcb
 - temp.c, 150
 - temp.h, 75
- depth
 - main.c, 171
- destroy
 - fat.c, 161
 - fat.h, 166
- device_id
 - param, 22
- device_not_available
 - interrupts.c, 106
- dir_entry, 9
 - attributes, 10
 - creationDate, 10
 - creationTime, 10
 - extension, 10
 - fileSize, 10
 - filename, 10
 - firstLogicalCluster, 10
 - ignore, 10
 - lastAccess, 10
 - lastWriteDate, 10
 - lastWriteTime, 10
 - reserved, 10
- dirty
 - page_entry, 20
- diskImage
 - main.c, 171
- divide_error
 - interrupts.c, 106
- do_bounds
 - interrupts.c, 106
- do_breakpoint
 - interrupts.c, 106
- do_coprocessor
 - interrupts.c, 106
- do_coprocessor_segment
 - interrupts.c, 106
- do_debug
 - interrupts.c, 106
- do_device_not_available
 - interrupts.c, 106
- do_divide_error
 - interrupts.c, 106
- do_double_fault
 - interrupts.c, 106
- do_general_protection
 - interrupts.c, 106
- do_invalid_op
 - interrupts.c, 106
- do_invalid_tss
 - interrupts.c, 106
- do_isr
 - interrupts.c, 106
- do_nmi
 - interrupts.c, 106
- do_overflow
 - interrupts.c, 106
- do_page_fault
 - interrupts.c, 107

- do_reserved
 - interrupts.c, [107](#)
- do_segment_not_present
 - interrupts.c, [107](#)
- do_stack_segment
 - interrupts.c, [107](#)
- double_fault
 - interrupts.c, [107](#)
- ds
 - context, [8](#)
- EXIT
 - mpx_supt.h, [63](#)
- eax
 - context, [8](#)
- ebp
 - context, [8](#)
- ebx
 - context, [8](#)
- ecx
 - context, [8](#)
- edi
 - context, [8](#)
- edx
 - context, [8](#)
- eflags
 - context, [8](#)
- eip
 - context, [8](#)
- empty
 - index_entry, [16](#)
- end
 - heap.c, [124](#)
- eraseCurrentRow
 - comHandler.c, [99](#)
 - comHandler.h, [26](#)
- es
 - context, [8](#)
- esi
 - context, [8](#)
- esp
 - context, [8](#)
- executeCommand
 - comHandler.c, [100](#)
 - comHandler.h, [26](#)
- extension
 - dir_entry, [10](#)
- FAT1_OFFSET
 - fat.h, [165](#)
- FAT2_OFFSET
 - fat.h, [165](#)
- FEB
 - time.h, [92](#)
- FREE
 - memoryControl.h, [56](#)
- FRI
 - time.h, [92](#)
- false
 - boolean.h, [23](#)
- fat.c
 - _BootSector, [162](#)
 - _CurrDirSize, [162](#)
 - _CurrentDirectory, [162](#)
 - _DiskImage, [163](#)
 - _FATTables, [163](#)
 - _getDiskOffsetForDirEntry, [158](#)
 - _getFirstFreeIndexInDirs, [158](#)
 - _getFirstFreeIndexInSector, [159](#)
 - _getFirstOpenSector, [159](#)
 - _isCurrentRoot, [163](#)
 - _loadBootSectorInfo, [159](#)
 - _loadFATTables, [159](#)
 - _loadRootDirectroy, [159](#)
 - _loadSectorAsDirectoryEntries, [159](#)
 - _prepNewDirSector, [160](#)
 - _readDirectoryEntry, [160](#)
 - _refreshDirectory, [160](#)
 - _saveDirEntry, [160](#)
 - _saveFATTables, [160](#)
 - changeToDirectory, [160](#)
 - changeToParentDirectory, [161](#)
 - destroy, [161](#)
 - getBootSector, [161](#)
 - getCurrentDirectory, [161](#)
 - getCurrentDirectoryMaxSize, [161](#)
 - getFATTables, [161](#)
 - getFileFromSector, [161](#)
 - initialize, [162](#)
 - moveFile, [162](#)
 - setFilename, [162](#)
- fat.h
 - ARCHIVE, [164](#)
 - BAD_CLUSTER, [164](#)
 - BOOT_SECTOR_OFFSET, [164](#)
 - changeToDirectory, [165](#)
 - changeToParentDirectory, [166](#)
 - DATA_AREA_OFFSET, [164](#)
 - DELETED, [164](#)
 - DIR_ENTRY_SIZE, [164](#)
 - destroy, [166](#)
 - FAT1_OFFSET, [165](#)
 - FAT2_OFFSET, [165](#)
 - getBootSector, [166](#)
 - getCurrentDirectory, [166](#)
 - getCurrentDirectoryMaxSize, [166](#)
 - getFATTables, [167](#)
 - getFileFromSector, [167](#)
 - HIDDEN, [165](#)
 - initialize, [167](#)
 - LAST_CLUSTER_BEGIN, [165](#)
 - LAST_CLUSTER_END, [165](#)
 - MAX_EXT_LENGTH, [165](#)
 - MAX_FILENAME_LENGTH, [165](#)
 - moveFile, [168](#)
 - READ_ONLY, [165](#)
 - REMAINING_FREE, [165](#)

- RESERVED_CLUSTER_BEGIN, 165
- RESERVED_CLUSTER_END, 165
- ROOT_DIRECTORY_OFFSET, 165
- SUBDIRECTORY, 165
- SYSTEM, 165
- setFilename, 168
- UNUSED, 165
- VOLUME_LABEL, 165
- fat1
 - fat_tables, 11
- fat2
 - fat_tables, 11
- fat_tables, 11
 - fat1, 11
 - fat2, 11
 - numEntries, 11
- fileSize
 - dir_entry, 10
- fileSystemType
 - boot_sector, 6
- filename
 - dir_entry, 10
 - main.c, 171
- findPCB
 - queue.c, 114
 - queue.h, 43
- first_free
 - paging.c, 129
 - paging.h, 59
- firstLogicalCluster
 - dir_entry, 10
- flags
 - gdt_entry_struct, 13
 - idt_entry_struct, 15
 - tables.h, 50
- footer, 11
 - head, 12
- frameaddr
 - page_entry, 20
- frames
 - paging.c, 130
- freeHead
 - memoryControl.c, 127
- freeMemory
 - memCommands.c, 156
 - memCommands.h, 82
- freePCB
 - pcb.c, 110
 - pcb.h, 41
- fs
 - context, 8
- funcPointer
 - functionDef, 12
- functionDef, 12
 - funcPointer, 12
 - helpString, 12
 - name, 12
- functionDefs
 - comHandler.c, 102
- functionInsertPoint
 - comHandler.c, 102
- GDT_CS_ID
 - system.h, 89
- GDT_DS_ID
 - system.h, 89
- gdt_descriptor_struct, 12
 - base, 13
 - limit, 13
- gdt_entries
 - tables.c, 121
- gdt_entry_struct, 13
 - access, 13
 - base_high, 13
 - base_low, 13
 - base_mid, 13
 - flags, 13
 - limit_low, 13
- gdt_init_entry
 - tables.c, 121
 - tables.h, 49
- gdt_ptr
 - tables.c, 121
- general_protection
 - interrupts.c, 107
- get_bit
 - paging.c, 129
 - paging.h, 59
- get_page
 - paging.c, 129
 - paging.h, 60
- getAllocatedHead
 - memoryControl.c, 126
 - memoryControl.h, 57
- getBlockedQueue
 - queue.c, 114
 - queue.h, 44
- getBootSector
 - fat.c, 161
 - fat.h, 166
- getCOPName
 - mpx_supt.c, 143
 - mpx_supt.h, 63
- getComHistory
 - comHandler.c, 100
 - comHandler.h, 26
- getCurrentDirectory
 - fat.c, 161
 - fat.h, 166
- getCurrentDirectoryMaxSize
 - fat.c, 161
 - fat.h, 166
- getDateTime
 - time.c, 138
 - time.h, 94
- getDayOfMonth
 - time.c, 138

- time.h, [94](#)
- getDayOfWeek
 - time.c, [138](#)
 - time.h, [94](#)
- getFATTables
 - fat.c, [161](#)
 - fat.h, [167](#)
- getFileFromSector
 - fat.c, [161](#)
 - fat.h, [167](#)
- getFreeHead
 - memoryControl.c, [126](#)
 - memoryControl.h, [57](#)
- getFunctionDef
 - comHandler.c, [100](#)
 - comHandler.h, [26](#)
- getHelpString
 - comHandler.c, [100](#)
 - comHandler.h, [27](#)
- getHours
 - time.c, [138](#)
 - time.h, [94](#)
- getInput
 - comHandler.c, [101](#)
 - comHandler.h, [27](#)
- getMinutes
 - time.c, [139](#)
 - time.h, [94](#)
- getMonth
 - time.c, [139](#)
 - time.h, [95](#)
- getReadyQueue
 - queue.c, [114](#)
 - queue.h, [44](#)
- getSeconds
 - time.c, [139](#)
 - time.h, [95](#)
- getSuspendedBlockedQueue
 - queue.c, [114](#)
 - queue.h, [44](#)
- getSuspendedReadyQueue
 - queue.c, [114](#)
 - queue.h, [44](#)
- getYear
 - time.c, [139](#)
 - time.h, [95](#)
- gs
 - context, [8](#)
- HELP_COMMAND_DATE
 - core/help.h, [31](#)
- HELP_COMMAND_HELP
 - core/help.h, [31](#)
- HELP_COMMAND_SHUTDOWN
 - core/help.h, [31](#)
- HELP_COMMAND_VERSION
 - core/help.h, [31](#)
- HELP_INVALID_ARGUMENTS
 - core/help.h, [32](#)
- HELP_R2_COMMAND_BPCB
 - help_temp.h, [67](#)
- HELP_R2_COMMAND_CPCB
 - help_temp.h, [67](#)
- HELP_R2_COMMAND_DPCB
 - help_temp.h, [68](#)
- HELP_R2_COMMAND_PPCB
 - modules/R2/commands/help.h, [33](#)
- HELP_R2_COMMAND_RPCB
 - modules/R2/commands/help.h, [33](#)
- HELP_R2_COMMAND_SHOWPCB
 - modules/R2/commands/help.h, [33](#)
- HELP_R2_COMMAND_SPCB
 - modules/R2/commands/help.h, [33](#)
- HELP_R2_COMMAND_UPCB
 - help_temp.h, [68](#)
- HELP_R3_COMMAND_LOAD
 - modules/R3/commands/help.h, [34](#)
- HELP_R3_COMMAND_YIELD
 - modules/R3/commands/help.h, [34](#)
- HELP_R5_COMMAND_ALLOC
 - modules/R5/help.h, [36](#)
- HELP_R5_COMMAND_EMPTY
 - modules/R5/help.h, [36](#)
- HELP_R5_COMMAND_FREE
 - modules/R5/help.h, [36](#)
- HELP_R5_COMMAND_HEAP
 - modules/R5/help.h, [37](#)
- HELP_R5_COMMAND_SHOWMEMORY
 - modules/R5/commands/help.h, [35](#)
- HELP_UNKNOWN_COMMAND
 - core/help.h, [32](#)
- HIDDEN
 - fat.h, [165](#)
- HOURS
 - time.h, [92](#)
- head
 - footer, [12](#)
- header, [14](#)
 - index_id, [14](#)
 - size, [14](#)
- heap, [14](#)
 - base, [15](#)
 - index, [15](#)
 - max_size, [15](#)
 - min_size, [15](#)
- heap.c
 - __end, [124](#)
 - _end, [124](#)
 - _kmallocc, [122](#)
 - alloc, [123](#)
 - curr_heap, [124](#)
 - end, [124](#)
 - kdir, [124](#)
 - kheap, [124](#)
 - kmallocc, [123](#)
 - make_heap, [123](#)
 - phys_alloc_addr, [124](#)

- heap.h
 - __attribute__, 54
 - _kmalloc, 53
 - alloc, 53
 - init_kheap, 54
 - KHEAP_BASE, 53
 - KHEAP_MIN, 53
 - KHEAP_SIZE, 53
 - kfree, 54
 - kmalloc, 54
 - make_heap, 54
 - TABLE_SIZE, 53
- help
 - comHandler.c, 101
 - commands.h, 29
- help_temp.h
 - HELP_R2_COMMAND_BPCB, 67
 - HELP_R2_COMMAND_CPCB, 67
 - HELP_R2_COMMAND_DPCB, 68
 - HELP_R2_COMMAND_UPCB, 68
- helpString
 - functionDef, 12
- hlt
 - system.h, 89
- hour
 - date_time, 9
- ICW1
 - interrupts.c, 105
- ICW4
 - interrupts.c, 105
- IDLE
 - mpx_supt.h, 63
- id
 - index_table, 17
- idle
 - mpx_supt.c, 143
 - mpx_supt.h, 63
- idt_entries
 - interrupts.c, 108
 - tables.c, 121
- idt_entry_struct, 15
 - base_high, 15
 - base_low, 15
 - flags, 15
 - sselect, 15
 - zero, 15
- idt_ptr
 - tables.c, 122
- idt_set_gate
 - tables.c, 121
 - tables.h, 49
- idt_struct, 16
 - base, 16
 - limit, 16
- ignore
 - dir_entry, 10
- ignore_1
 - boot_sector, 6
- ignore_2
 - boot_sector, 6
- ignore_3
 - boot_sector, 6
- ignore_4
 - boot_sector, 6
- ignore_5
 - boot_sector, 6
- imageName
 - main.c, 171
- inb
 - io.h, 38
- include/boolean.h, 23
- include/core/asm.h, 24
- include/core/comHandler.h, 24
- include/core/commands.h, 28
- include/core/help.h, 30
- include/core/interrupts.h, 37
- include/core/io.h, 38
- include/core/pcb.h, 39
- include/core/queue.h, 42
- include/core/serial.h, 46
- include/core/tables.h, 48
- include/core/version.h, 50
- include/math.h, 51
- include/mem/heap.h, 52
- include/mem/memoryControl.h, 55
- include/mem/paging.h, 58
- include/modules/R2/commands/help.h, 32
- include/modules/R2/commands/help_temp.h, 67
- include/modules/R2/commands/perm.h, 68
- include/modules/R2/commands/status.h, 72
- include/modules/R2/commands/status_temp.h, 73
- include/modules/R2/commands/temp.h, 74
- include/modules/R3/commands/help.h, 34
- include/modules/R3/commands/r3commands.h, 76
- include/modules/R3/processes.h, 78
- include/modules/R5/commands/help.h, 35
- include/modules/R5/commands/r5commands.h, 79
- include/modules/R5/help.h, 36
- include/modules/R5/memCommands.h, 81
- include/modules/mpx_supt.h, 61
- include/regex.h, 83
- include/string.h, 84
- include/system.h, 88
- include/time.h, 91
- index
 - heap, 15
- index_entry, 16
 - block, 16
 - empty, 16
 - size, 16
- index_id
 - header, 14
- index_table, 17
 - id, 17
 - table, 17
- init_gdt

- tables.c, 121
- tables.h, 49
- init_idt
 - tables.c, 121
 - tables.h, 50
- init_irq
 - interrupts.c, 107
 - interrupts.h, 37
- init_kheap
 - heap.h, 54
- init_paging
 - paging.c, 129
 - paging.h, 60
- init_pic
 - interrupts.c, 107
 - interrupts.h, 37
- init_serial
 - serial.c, 117
 - serial.h, 46
- initCommandHandler
 - comHandler.c, 101
 - comHandler.h, 27
- initHeap
 - memCommands.c, 157
 - memCommands.h, 82
- initialize
 - fat.c, 162
 - fat.h, 167
- initializeHeap
 - memoryControl.c, 126
 - memoryControl.h, 57
- insertPCB
 - queue.c, 115
 - queue.h, 44
- interrupts.c
 - bounds, 106
 - breakpoint, 106
 - coprocessor, 106
 - coprocessor_segment, 106
 - debug, 106
 - device_not_available, 106
 - divide_error, 106
 - do_bounds, 106
 - do_breakpoint, 106
 - do_coprocessor, 106
 - do_coprocessor_segment, 106
 - do_debug, 106
 - do_device_not_available, 106
 - do_divide_error, 106
 - do_double_fault, 106
 - do_general_protection, 106
 - do_invalid_op, 106
 - do_invalid_tss, 106
 - do_isr, 106
 - do_nmi, 106
 - do_overflow, 106
 - do_page_fault, 107
 - do_reserved, 107
 - do_segment_not_present, 107
 - do_stack_segment, 107
 - double_fault, 107
 - general_protection, 107
 - ICW1, 105
 - ICW4, 105
 - idt_entries, 108
 - init_irq, 107
 - init_pic, 107
 - invalid_op, 107
 - invalid_tss, 107
 - io_wait, 105
 - isr0, 107
 - nmi, 107
 - overflow, 107
 - PIC1, 105
 - PIC2, 106
 - page_fault, 107
 - reserved, 107
 - rtc_isr, 107
 - segment_not_present, 107
 - stack_segment, 107
 - sys_call_isr, 107
- interrupts.h
 - init_irq, 37
 - init_pic, 37
- invalid_op
 - interrupts.c, 107
- invalid_tss
 - interrupts.c, 107
- io.h
 - inb, 38
 - outb, 38
- io_wait
 - interrupts.c, 105
- iret
 - system.h, 89
- isChar
 - string.c, 134
 - string.h, 85
- isEmpty
 - memoryControl.c, 127
 - memoryControl.h, 58
- isEmptyCom
 - memCommands.c, 157
 - memCommands.h, 82
- isInitialized
 - memoryControl.c, 127
- isLeapYear
 - time.c, 139
 - time.h, 95
- isLowerChar
 - string.c, 134
 - string.h, 85
- isSuspended
 - pcb, 22
- isUpperChar
 - string.c, 135

- string.h, 86
- isdigit
 - string.c, 134
 - string.h, 85
- isr0
 - interrupts.c, 107
- isspace
 - string.c, 135
 - string.h, 86
- itoa
 - string.c, 135
 - string.h, 86
- JAN
 - time.h, 92
- JUL
 - time.h, 92
- JUN
 - time.h, 93
- KHEAP_BASE
 - heap.h, 53
- KHEAP_MIN
 - heap.h, 53
- KHEAP_SIZE
 - heap.h, 53
- kdir
 - heap.c, 124
 - paging.c, 130
- kernel/core/comHandler.c, 98
- kernel/core/commands.c, 103
- kernel/core/interrupts.c, 104
- kernel/core/kmain.c, 108
- kernel/core/pcb.c, 108
- kernel/core/queue.c, 111
- kernel/core/serial.c, 116
- kernel/core/system.c, 119
- kernel/core/tables.c, 120
- kernel/mem/heap.c, 122
- kernel/mem/memoryControl.c, 124
- kernel/mem/paging.c, 127
- kfree
 - heap.h, 54
- kheap
 - heap.c, 124
 - paging.c, 130
- klogv
 - system.c, 119
 - system.h, 90
- kmain
 - kmain.c, 108
- kmain.c
 - kmain, 108
- kmalloc
 - heap.c, 123
 - heap.h, 54
- kpanic
 - system.c, 119
 - system.h, 90
- LAST_CLUSTER_BEGIN
 - fat.h, 165
- LAST_CLUSTER_END
 - fat.h, 165
- lastAccess
 - dir_entry, 10
- lastWriteDate
 - dir_entry, 10
- lastWriteTime
 - dir_entry, 10
- lib/math.c, 131
- lib/regex.c, 132
- lib/string.c, 133
- lib/time.c, 137
- limit
 - gdt_descriptor_struct, 13
 - idt_struct, 16
 - tables.h, 50
- limit_low
 - gdt_entry_struct, 13
 - tables.h, 50
- lmcb, 17
 - memSize, 18
 - memoryControl.h, 56
 - size, 18
 - type, 18
- load_page_dir
 - paging.c, 129
 - paging.h, 60
- loadr3
 - r3commands.c, 152
 - r3commands.h, 77
- MAX_EXT_LENGTH
 - fat.h, 165
- MAX_FILENAME_LENGTH
 - fat.h, 165
- MAR
 - time.h, 93
- MAY
 - time.h, 93
- MINUTES
 - time.h, 93
- MODULE_R1
 - mpx_supt.h, 63
- MODULE_R2
 - mpx_supt.h, 63
- MODULE_R3
 - mpx_supt.h, 63
- MODULE_R4
 - mpx_supt.h, 63
- MODULE_R5
 - mpx_supt.h, 63
- MONTH
 - time.h, 93
- MON
 - time.h, 93
- main
 - main.c, 171

- main.c
 - _callCommand, 169
 - _extcmp, 169
 - _fncmp, 169
 - _getClusterOfFileWithName, 170
 - _getIndexOfFileWithName, 170
 - _getSizeOfFileWithName, 170
 - _launchCommandInterface, 170
 - _nameCmpHelper, 170
 - _printBootSectorInfo, 170
 - _printDirectoryEntries, 170
 - _printDirectoryEntriesByFileName, 171
 - _printDirectoryEntriesByType, 171
 - _printFATTableInfo, 171
 - _printFile, 171
 - depth, 171
 - diskImage, 171
 - filename, 171
 - imageName, 171
 - main, 171
 - paths, 171
 - printFileFlag, 171
- make_heap
 - heap.c, 123
 - heap.h, 54
- math.c
 - bcdToDec, 131
 - decToBcd, 131
- math.h
 - bcdToDec, 51
 - decToBcd, 51
- max_size
 - heap, 15
- maxRootDirEntries
 - boot_sector, 6
- mem_size
 - paging.c, 130
- memAllocated
 - memoryControl.c, 127
- memCommands.c
 - allocateMem, 156
 - freeMemory, 156
 - initHeap, 157
 - isEmptyCom, 157
 - registerR5TempCommands, 157
- memCommands.h
 - allocateMem, 82
 - freeMemory, 82
 - initHeap, 82
 - isEmptyCom, 82
 - registerR5TempCommands, 82
- memHeap
 - memoryControl.c, 127
- memSize
 - cmcb, 7
 - lmcb, 18
 - memoryControl.c, 127
- memoryControl.c
 - _mergeAdjacentFree, 125
 - _placeStructs, 125
 - allocateMemory, 125
 - allocatedHead, 127
 - deallocateMemory, 126
 - freeHead, 127
 - getAllocatedHead, 126
 - getFreeHead, 126
 - initializeHeap, 126
 - isEmpty, 127
 - isInitialized, 127
 - memAllocated, 127
 - memHeap, 127
 - memSize, 127
- memoryControl.h
 - ALLOCATED, 56
 - allocateMemory, 56
 - cmcb, 56
 - deallocateMemory, 56
 - FREE, 56
 - getAllocatedHead, 57
 - getFreeHead, 57
 - initializeHeap, 57
 - isEmpty, 58
 - lmcb, 56
- memset
 - mpx_supt.c, 143
 - mpx_supt.h, 64
- min
 - date_time, 9
- min_size
 - heap, 15
- modules/R2/commands/help.h
 - HELP_R2_COMMAND_PPCB, 33
 - HELP_R2_COMMAND_RPCB, 33
 - HELP_R2_COMMAND_SHOWPCB, 33
 - HELP_R2_COMMAND_SPCB, 33
- modules/R2/commands/perm.c, 146
- modules/R2/commands/temp.c, 148
- modules/R3/commands/help.h
 - HELP_R3_COMMAND_LOAD, 34
 - HELP_R3_COMMAND_YIELD, 34
- modules/R3/commands/r3commands.c, 151
- modules/R3/procsr3.c, 153
- modules/R5/commands/help.h
 - HELP_R5_COMMAND_SHOWMEMORY, 35
- modules/R5/commands/r5commands.c, 154
- modules/R5/help.h
 - HELP_R5_COMMAND_ALLOC, 36
 - HELP_R5_COMMAND_EMPTY, 36
 - HELP_R5_COMMAND_FREE, 36
 - HELP_R5_COMMAND_HEAP, 37
- modules/R5/memCommands.c, 155
- modules/mpx_supt.c, 142
- mon
 - date_time, 9
- moveFile
 - fat.c, 162

- fat.h, 168
- mpx_init
 - mpx_supt.c, 144
 - mpx_supt.h, 65
- mpx_supt.c
 - callerContext, 145
 - cop, 145
 - current_module, 145
 - getCOPName, 143
 - idle, 143
 - memset, 143
 - mpx_init, 144
 - params, 145
 - student_free, 145
 - student_malloc, 145
 - sys_alloc_mem, 144
 - sys_call, 144
 - sys_free_mem, 144
 - sys_req, 145
 - sys_set_free, 145
 - sys_set_malloc, 145
- mpx_supt.h
 - context, 63
 - EXIT, 63
 - getCOPName, 63
 - IDLE, 63
 - idle, 63
 - MODULE_R1, 63
 - MODULE_R2, 63
 - MODULE_R3, 63
 - MODULE_R4, 63
 - MODULE_R5, 63
 - memset, 64
 - mpx_init, 65
 - READ, 63
 - sys_alloc_mem, 65
 - sys_call, 65
 - sys_free_mem, 65
 - sys_req, 66
 - sys_set_free, 66
 - sys_set_malloc, 66
 - WRITE, 63
- NMI_DISABLE
 - time.h, 93
- NMI_ENABLE
 - time.h, 93
- NO_ERROR
 - serial.c, 117
- NOV
 - time.h, 93
- NULL
 - system.h, 90
- name
 - cmcb, 7
 - functionDef, 12
- new_frame
 - paging.c, 130
 - paging.h, 60
- next
 - cmcb, 7
 - node, 19
- nframes
 - paging.c, 130
- nmi
 - interrupts.c, 107
- no_warn
 - system.h, 89
- node, 18
 - data, 19
 - next, 19
 - prev, 19
 - queue.h, 43
- nop
 - system.h, 90
- numEntries
 - fat_tables, 11
- numFATCopies
 - boot_sector, 6
- numHeads
 - boot_sector, 6
- numReservedSectors
 - boot_sector, 6
- numSectors
 - boot_sector, 6
- OCT
 - time.h, 93
- OS_VERSION
 - version.h, 51
- op_code
 - param, 22
- outb
 - io.h, 38
- overflow
 - interrupts.c, 107
- P1_NAME
 - r3commands.c, 152
- P2_NAME
 - r3commands.c, 152
- P3_NAME
 - r3commands.c, 152
- P4_NAME
 - r3commands.c, 152
- P5_NAME
 - r3commands.c, 152
- PAGE_SIZE
 - paging.h, 59
- PIC1
 - interrupts.c, 105
- PIC2
 - interrupts.c, 106
- PROCESS_NAME_ALREADY_EXISTS
 - status_temp.h, 73
- page_dir, 19
 - tables, 20
 - tables_phys, 20

- page_entry, 20
 - accessed, 20
 - dirty, 20
 - frameaddr, 20
 - present, 20
 - reserved, 20
 - usermode, 20
 - writable, 20
- page_fault
 - interrupts.c, 107
- page_size
 - paging.c, 130
- page_table, 21
 - pages, 21
- pages
 - page_table, 21
- paging.c
 - cdir, 130
 - clear_bit, 128
 - first_free, 129
 - frames, 130
 - get_bit, 129
 - get_page, 129
 - init_paging, 129
 - kdir, 130
 - kheap, 130
 - load_page_dir, 129
 - mem_size, 130
 - new_frame, 130
 - nframes, 130
 - page_size, 130
 - phys_alloc_addr, 130
 - set_bit, 130
- paging.h
 - clear_bit, 59
 - first_free, 59
 - get_bit, 59
 - get_page, 60
 - init_paging, 60
 - load_page_dir, 60
 - new_frame, 60
 - PAGE_SIZE, 59
 - set_bit, 61
- param, 21
 - device_id, 22
 - op_code, 22
- params
 - mpx_supt.c, 145
- paths
 - main.c, 171
- pcb, 22
 - isSuspended, 22
 - pcb.h, 40
 - priority, 22
 - processClass, 22
 - processName, 22
 - stackBottom, 22
 - stackTop, 22
 - state, 22
- pcb.c
 - allocatePCB, 109
 - checkParamClass, 109
 - checkParamName, 110
 - checkParamPriority, 110
 - freePCB, 110
 - setupPCB, 110
- pcb.h
 - APPLICATION, 40
 - allocatePCB, 40
 - BLOCKED, 40
 - checkParamClass, 40
 - checkParamName, 41
 - checkParamPriority, 41
 - freePCB, 41
 - pcb, 40
 - READY, 40
 - RUNNING, 40
 - SYSTEM, 40
 - setupPCB, 42
- perm.c
 - printPcbInfo, 146
 - printQueueInfo, 146
 - registerR2PermCommands, 146
 - resumePcb, 146
 - setPriorityPcb, 147
 - showPcbInfo, 147
 - suspendPcb, 147
- perm.h
 - registerR2PermCommands, 69
 - resumePcb, 69
 - setPriorityPcb, 70
 - showPcbInfo, 70
 - suspendPcb, 71
- phys_alloc_addr
 - heap.c, 124
 - paging.c, 130
- popBlocked
 - queue.c, 115
 - queue.h, 45
- popReady
 - queue.c, 115
 - queue.h, 45
- popSuspendedBlocked
 - queue.c, 115
 - queue.h, 45
- popSuspendedReady
 - queue.c, 115
 - queue.h, 45
- present
 - page_entry, 20
- prev
 - cmcb, 7
 - node, 19
- printBlockInfo
 - r5commands.c, 155
- printCmcbInfo

- r5commands.c, 155
- printFileFlag
 - main.c, 171
- printPcbInfo
 - perm.c, 146
- printQueueInfo
 - perm.c, 146
- printStart
 - comHandler.c, 101
 - comHandler.h, 27
- priority
 - pcb, 22
- proc1
 - processes.h, 79
 - procsr3.c, 154
- proc2
 - processes.h, 79
 - procsr3.c, 154
- proc3
 - processes.h, 79
 - procsr3.c, 154
- proc4
 - processes.h, 79
 - procsr3.c, 154
- proc5
 - processes.h, 79
 - procsr3.c, 154
- processClass
 - pcb, 22
- processName
 - pcb, 22
- processes.h
 - proc1, 79
 - proc2, 79
 - proc3, 79
 - proc4, 79
 - proc5, 79
- procsr3.c
 - proc1, 154
 - proc2, 154
 - proc3, 154
 - proc4, 154
 - proc5, 154
 - RC_1, 154
 - RC_2, 154
 - RC_3, 154
 - RC_4, 154
 - RC_5, 154
- QUEUE_BLOCKED
 - queue.c, 112
- QUEUE_READY
 - queue.c, 112
- QUEUE_SUSPENDED_BLOCKED
 - queue.c, 112
- QUEUE_SUSPENDED_READY
 - queue.c, 112
- queue
 - queue.c, 112
- queue.c
 - _findNode, 112
 - _findNodeInQueue, 112
 - _insertFIFO, 113
 - _insertPriority, 113
 - _newNode, 113
 - findPCB, 114
 - getBlockedQueue, 114
 - getReadyQueue, 114
 - getSuspendedBlockedQueue, 114
 - getSuspendedReadyQueue, 114
 - insertPCB, 115
 - popBlocked, 115
 - popReady, 115
 - popSuspendedBlocked, 115
 - popSuspendedReady, 115
 - QUEUE_BLOCKED, 112
 - QUEUE_READY, 112
 - QUEUE_SUSPENDED_BLOCKED, 112
 - QUEUE_SUSPENDED_READY, 112
 - queue, 112
 - queues, 116
 - removePCB, 116
- queue.h
 - findPCB, 43
 - getBlockedQueue, 44
 - getReadyQueue, 44
 - getSuspendedBlockedQueue, 44
 - getSuspendedReadyQueue, 44
 - insertPCB, 44
 - node, 43
 - popBlocked, 45
 - popReady, 45
 - popSuspendedBlocked, 45
 - popSuspendedReady, 45
 - removePCB, 45
- queues
 - queue.c, 116
- r3commands.c
 - loadr3, 152
 - P1_NAME, 152
 - P2_NAME, 152
 - P3_NAME, 152
 - P4_NAME, 152
 - P5_NAME, 152
 - registerR3Commands, 152
 - yield, 152
- r3commands.h
 - loadr3, 77
 - registerR3Commands, 78
 - yield, 78
- r5commands.c
 - printBlockInfo, 155
 - printCmcbInfo, 155
 - registerR5PermCommands, 155
 - showMemory, 155
- r5commands.h
 - registerR5PermCommands, 80

- showMemory, 80
- r6/fat.c, 157
- r6/fat.h, 163
- r6/main.c, 168
- RC_1
 - procsr3.c, 154
- RC_2
 - procsr3.c, 154
- RC_3
 - procsr3.c, 154
- RC_4
 - procsr3.c, 154
- RC_5
 - procsr3.c, 154
- READ_ONLY
 - fat.h, 165
- READY
 - pcb.h, 40
- READ
 - mpx_supt.h, 63
- REMAINING_FREE
 - fat.h, 165
- RESERVED_CLUSTER_BEGIN
 - fat.h, 165
- RESERVED_CLUSTER_END
 - fat.h, 165
- RESUME_PCB_SUCCESS
 - status.h, 72
- RESUME_PCBS_SUCCESS
 - status.h, 72
- ROOT_DIRECTORY_OFFSET
 - fat.h, 165
- RUNNING
 - pcb.h, 40
- regex.c
 - testRegex, 132
- regex.h
 - testRegex, 83
- registerR2PermCommands
 - perm.c, 146
 - perm.h, 69
- registerR2TempCommands
 - temp.c, 150
 - temp.h, 76
- registerR3Commands
 - r3commands.c, 152
 - r3commands.h, 78
- registerR5PermCommands
 - r5commands.c, 155
 - r5commands.h, 80
- registerR5TempCommands
 - memCommands.c, 157
 - memCommands.h, 82
- removePCB
 - queue.c, 116
 - queue.h, 45
- reserved
 - dir_entry, 10
- interrupts.c, 107
- page_entry, 20
- resumePcb
 - perm.c, 146
 - perm.h, 69
- returnToInsertionPoint
 - comHandler.c, 101
 - comHandler.h, 27
- reverse
 - string.c, 135
 - string.h, 86
- rtc_isr
 - interrupts.c, 107
- SAT
 - time.h, 93
- SECONDS
 - time.h, 93
- SEP
 - time.h, 93
- SUBDIRECTORY
 - fat.h, 165
- SUSPEND_PCB_SUCCESS
 - status.h, 72
- SUN
 - time.h, 93
- SYSTEM
 - fat.h, 165
 - pcb.h, 40
- sec
 - date_time, 9
- sectorCountFAT32
 - boot_sector, 6
- sectorsPerCluster
 - boot_sector, 6
- sectorsPerFAT
 - boot_sector, 6
- sectorsPerTrack
 - boot_sector, 6
- segment_not_present
 - interrupts.c, 107
- serial.c
 - init_serial, 117
 - NO_ERROR, 117
 - serial_port_in, 118
 - serial_port_out, 118
 - serial_print, 117
 - serial_println, 118
 - set_serial_in, 118
 - set_serial_out, 118
- serial.h
 - COM1, 46
 - COM2, 46
 - COM3, 46
 - COM4, 46
 - init_serial, 46
 - serial_print, 47
 - serial_println, 47
 - set_serial_in, 47

- set_serial_out, 47
- serial_port_in
 - serial.c, 118
- serial_port_out
 - serial.c, 118
- serial_print
 - serial.c, 117
 - serial.h, 47
- serial_println
 - serial.c, 118
 - serial.h, 47
- set_bit
 - paging.c, 130
 - paging.h, 61
- set_serial_in
 - serial.c, 118
 - serial.h, 47
- set_serial_out
 - serial.c, 118
 - serial.h, 47
- setDateTime
 - time.c, 140
 - time.h, 95
- setDayOfMonth
 - time.c, 140
 - time.h, 96
- setDayOfWeek
 - time.c, 140
 - time.h, 96
- setFilename
 - fat.c, 162
 - fat.h, 168
- setHours
 - time.c, 140
 - time.h, 96
- setMinutes
 - time.c, 140
 - time.h, 96
- setMonth
 - time.c, 141
 - time.h, 96
- setPriorityPcb
 - perm.c, 147
 - perm.h, 70
- setSeconds
 - time.c, 141
 - time.h, 97
- setYear
 - time.c, 141
 - time.h, 97
- setupCommands
 - comHandler.c, 102
 - comHandler.h, 27
- setupPCB
 - pcb.c, 110
 - pcb.h, 42
- showMemory
 - r5commands.c, 155
 - r5commands.h, 80
- showPcbInfo
 - perm.c, 147
 - perm.h, 70
- shutdown
 - comHandler.c, 102
 - commands.h, 29
- size
 - cmcb, 7
 - header, 14
 - index_entry, 16
 - lmcb, 18
- size_t
 - system.h, 90
- sselect
 - idt_entry_struct, 15
 - tables.h, 50
- stack_segment
 - interrupts.c, 107
- stackBottom
 - pcb, 22
- stackTop
 - pcb, 22
- state
 - pcb, 22
- status.h
 - RESUME_PCB_SUCCESS, 72
 - RESUME_PCBS_SUCCESS, 72
 - SUSPEND_PCB_SUCCESS, 72
 - UNKNOWN_PCB_NAME, 72
 - UPDATE_PRIORITY_SUCCESS, 72
- status_temp.h
 - BLOCK_PCB_SUCCESS, 73
 - CREATE_PCB_SUCCESS, 73
 - DELETE_PCB_SUCCESS, 73
 - PROCESS_NAME_ALREADY_EXISTS, 73
 - UNBLOCK_PCB_SUCCESS, 73
- sti
 - system.h, 90
- strcat
 - string.c, 136
 - string.h, 87
- strcmp
 - string.c, 136
 - string.h, 87
- strcpy
 - string.c, 136
 - string.h, 87
- string.c
 - atoi, 134
 - isChar, 134
 - isLowerChar, 134
 - isUpperChar, 135
 - isdigit, 134
 - isspace, 135
 - itoa, 135
 - reverse, 135
 - strcat, 136

- strcmp, 136
 - strcpy, 136
 - strlen, 136
 - strtok, 137
- string.h
 - atoi, 84
 - isChar, 85
 - isLowerChar, 85
 - isUpperChar, 86
 - isdigit, 85
 - isspace, 86
 - itoa, 86
 - reverse, 86
 - strcat, 87
 - strcmp, 87
 - strcpy, 87
 - strlen, 88
 - strtok, 88
- strlen
 - string.c, 136
 - string.h, 88
- strtok
 - string.c, 137
 - string.h, 88
- student_free
 - mpx_supt.c, 145
- student_malloc
 - mpx_supt.c, 145
- suspendPcb
 - perm.c, 147
 - perm.h, 71
- sys_alloc_mem
 - mpx_supt.c, 144
 - mpx_supt.h, 65
- sys_call
 - mpx_supt.c, 144
 - mpx_supt.h, 65
- sys_call_isr
 - interrupts.c, 107
- sys_free_mem
 - mpx_supt.c, 144
 - mpx_supt.h, 65
- sys_req
 - mpx_supt.c, 145
 - mpx_supt.h, 66
- sys_set_free
 - mpx_supt.c, 145
 - mpx_supt.h, 66
- sys_set_malloc
 - mpx_supt.c, 145
 - mpx_supt.h, 66
- system.c
 - klogv, 119
 - kpanic, 119
- system.h
 - asm, 89
 - cli, 89
 - GDT_CS_ID, 89
 - GDT_DS_ID, 89
 - hlt, 89
 - iret, 89
 - klogv, 90
 - kpanic, 90
 - NULL, 90
 - no_warn, 89
 - nop, 90
 - size_t, 90
 - sti, 90
 - u16int, 90
 - u32int, 90
 - u8int, 90
 - volatile, 90
- TABLE_SIZE
 - heap.h, 53
- THU
 - time.h, 93
- TIME_DELIM
 - time.h, 93
- TUE
 - time.h, 93
- table
 - index_table, 17
- tables
 - page_dir, 20
- tables.c
 - gdt_entries, 121
 - gdt_init_entry, 121
 - gdt_ptr, 121
 - idt_entries, 121
 - idt_ptr, 122
 - idt_set_gate, 121
 - init_gdt, 121
 - init_idt, 121
 - write_gdt_ptr, 121
 - write_idt_ptr, 121
- tables.h
 - __attribute__, 49
 - access, 50
 - base, 50
 - base_high, 50
 - base_low, 50
 - base_mid, 50
 - flags, 50
 - gdt_init_entry, 49
 - idt_set_gate, 49
 - init_gdt, 49
 - init_idt, 50
 - limit, 50
 - limit_low, 50
 - sselect, 50
 - zero, 50
- tables_phys
 - page_dir, 20
- temp.c
 - blockPcb, 149
 - createPcb, 150

- deletePcb, [150](#)
- registerR2TempCommands, [150](#)
- unlockPcb, [151](#)
- temp.h
 - blockPcb, [75](#)
 - createPcb, [75](#)
 - deletePcb, [75](#)
 - registerR2TempCommands, [76](#)
 - unlockPcb, [76](#)
- testRegex
 - regex.c, [132](#)
 - regex.h, [83](#)
- time.c
 - DAYS_IN_MONTH, [142](#)
 - getDateTime, [138](#)
 - getDayOfMonth, [138](#)
 - getDayOfWeek, [138](#)
 - getHours, [138](#)
 - getMinutes, [139](#)
 - getMonth, [139](#)
 - getSeconds, [139](#)
 - getYear, [139](#)
 - isLeapYear, [139](#)
 - setDateTime, [140](#)
 - setDayOfMonth, [140](#)
 - setDayOfWeek, [140](#)
 - setHours, [140](#)
 - setMinutes, [140](#)
 - setMonth, [141](#)
 - setSeconds, [141](#)
 - setYear, [141](#)
 - updateDayOfWeek, [141](#)
 - updateDayOfYear, [141](#)
- time.h
 - APR, [92](#)
 - AUG, [92](#)
 - CONTROL_PORT, [92](#)
 - DATA_PORT, [92](#)
 - DAY_MONTH, [92](#)
 - DAY_WEEK, [92](#)
 - DAYS_IN_MONTH, [98](#)
 - DEC, [92](#)
 - FEB, [92](#)
 - FRI, [92](#)
 - getDateTime, [94](#)
 - getDayOfMonth, [94](#)
 - getDayOfWeek, [94](#)
 - getHours, [94](#)
 - getMinutes, [94](#)
 - getMonth, [95](#)
 - getSeconds, [95](#)
 - getYear, [95](#)
 - HOURS, [92](#)
 - isLeapYear, [95](#)
 - JAN, [92](#)
 - JUL, [92](#)
 - JUN, [93](#)
 - MAR, [93](#)
 - MAY, [93](#)
 - MINUTES, [93](#)
 - MONTH, [93](#)
 - MON, [93](#)
 - NMI_DISABLE, [93](#)
 - NMI_ENABLE, [93](#)
 - NOV, [93](#)
 - OCT, [93](#)
 - SAT, [93](#)
 - SECONDS, [93](#)
 - SEP, [93](#)
 - SUN, [93](#)
 - setDateTime, [95](#)
 - setDayOfMonth, [96](#)
 - setDayOfWeek, [96](#)
 - setHours, [96](#)
 - setMinutes, [96](#)
 - setMonth, [96](#)
 - setSeconds, [97](#)
 - setYear, [97](#)
 - THU, [93](#)
 - TIME_DELIM, [93](#)
 - TUE, [93](#)
 - updateDayOfWeek, [97](#)
 - updateDayOfYear, [97](#)
 - WED, [94](#)
 - YEAR, [94](#)
- true
 - boolean.h, [23](#)
- type
 - cmcb, [7](#)
 - lmcb, [18](#)
- u16int
 - system.h, [90](#)
- u32int
 - system.h, [90](#)
- u8int
 - system.h, [90](#)
- UNBLOCK_PCB_SUCCESS
 - status_temp.h, [73](#)
- UNKNOWN_PCB_NAME
 - status.h, [72](#)
- UNUSED
 - fat.h, [165](#)
- UPDATE_PRIORITY_SUCCESS
 - status.h, [72](#)
- unlockPcb
 - temp.c, [151](#)
 - temp.h, [76](#)
- updateDayOfWeek
 - time.c, [141](#)
 - time.h, [97](#)
- updateDayOfYear
 - time.c, [141](#)
 - time.h, [97](#)
- usermode
 - page_entry, [20](#)

VOLUME_LABEL
 fat.h, [165](#)
version
 commands.c, [103](#)
 commands.h, [30](#)
version.h
 OS_VERSION, [51](#)
volld
 boot_sector, [6](#)
volName
 boot_sector, [6](#)
volatile
 system.h, [90](#)

WED
 time.h, [94](#)
WRITE
 mpx_supt.h, [63](#)
write_gdt_ptr
 tables.c, [121](#)
write_idt_ptr
 tables.c, [121](#)
writeable
 page_entry, [20](#)

YEAR
 time.h, [94](#)
year
 date_time, [9](#)
yield
 r3commands.c, [152](#)
 r3commands.h, [78](#)

zero
 idt_entry_struct, [15](#)
 tables.h, [50](#)