

MPX-Fall2020-Group9

1

Generated by Doxygen 1.9.0

1 MPX-Fall2020-Group9	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 date_time Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Member Data Documentation	7
4.1.2.1 day_m	7
4.1.2.2 day_w	8
4.1.2.3 day_y	8
4.1.2.4 hour	8
4.1.2.5 min	8
4.1.2.6 mon	8
4.1.2.7 sec	8
4.1.2.8 year	9
4.2 footer Struct Reference	9
4.2.1 Detailed Description	9
4.2.2 Member Data Documentation	9
4.2.2.1 head	9
4.3 gdt_descriptor_struct Struct Reference	9
4.3.1 Detailed Description	10
4.3.2 Member Data Documentation	10
4.3.2.1 base	10
4.3.2.2 limit	10
4.4 gdt_entry_struct Struct Reference	10
4.4.1 Detailed Description	10
4.4.2 Member Data Documentation	11
4.4.2.1 access	11
4.4.2.2 base_high	11
4.4.2.3 base_low	11
4.4.2.4 base_mid	11
4.4.2.5 flags	11
4.4.2.6 limit_low	12
4.5 header Struct Reference	12
4.5.1 Detailed Description	12
4.5.2 Member Data Documentation	12
4.5.2.1 index_id	12
4.5.2.2 size	12

4.6 heap Struct Reference	13
4.6.1 Detailed Description	13
4.6.2 Member Data Documentation	13
4.6.2.1 base	13
4.6.2.2 index	13
4.6.2.3 max_size	13
4.6.2.4 min_size	14
4.7 idt_entry_struct Struct Reference	14
4.7.1 Detailed Description	14
4.7.2 Member Data Documentation	14
4.7.2.1 base_high	14
4.7.2.2 base_low	14
4.7.2.3 flags	15
4.7.2.4 sselect	15
4.7.2.5 zero	15
4.8 idt_struct Struct Reference	15
4.8.1 Detailed Description	15
4.8.2 Member Data Documentation	15
4.8.2.1 base	16
4.8.2.2 limit	16
4.9 index_entry Struct Reference	16
4.9.1 Detailed Description	16
4.9.2 Member Data Documentation	16
4.9.2.1 block	16
4.9.2.2 empty	17
4.9.2.3 size	17
4.10 index_table Struct Reference	17
4.10.1 Detailed Description	17
4.10.2 Member Data Documentation	17
4.10.2.1 id	17
4.10.2.2 table	18
4.11 page_dir Struct Reference	18
4.11.1 Detailed Description	18
4.11.2 Member Data Documentation	18
4.11.2.1 tables	18
4.11.2.2 tables_phys	18
4.12 page_entry Struct Reference	19
4.12.1 Detailed Description	19
4.12.2 Member Data Documentation	19
4.12.2.1 accessed	19
4.12.2.2 dirty	19
4.12.2.3 frameaddr	19

4.12.2.4 present	20
4.12.2.5 reserved	20
4.12.2.6 usermode	20
4.12.2.7 writeable	20
4.13 page_table Struct Reference	20
4.13.1 Detailed Description	20
4.13.2 Member Data Documentation	21
4.13.2.1 pages	21
4.14 param Struct Reference	21
4.14.1 Detailed Description	21
4.14.2 Member Data Documentation	21
4.14.2.1 buffer_ptr	21
4.14.2.2 count_ptr	22
4.14.2.3 device_id	22
4.14.2.4 op_code	22
5 File Documentation	23
5.1 include/core/asm.h File Reference	23
5.2 include/core/interrupts.h File Reference	23
5.2.1 Function Documentation	23
5.2.1.1 init_irq()	23
5.2.1.2 init_pic()	24
5.3 include/core/io.h File Reference	24
5.3.1 Macro Definition Documentation	24
5.3.1.1 inb	25
5.3.1.2 outb	25
5.4 include/core/serial.h File Reference	25
5.4.1 Macro Definition Documentation	25
5.4.1.1 COM1	26
5.4.1.2 COM2	26
5.4.1.3 COM3	26
5.4.1.4 COM4	26
5.4.2 Function Documentation	26
5.4.2.1 init_serial()	26
5.4.2.2 polling()	27
5.4.2.3 serial_print()	29
5.4.2.4 serial_println()	29
5.4.2.5 set_serial_in()	29
5.4.2.6 set_serial_out()	30
5.5 include/core/tables.h File Reference	30
5.5.1 Function Documentation	30
5.5.1.1 __attribute__()	31

5.5.1.2 gdt_init_entry()	31
5.5.1.3 idt_set_gate()	31
5.5.1.4 init_gdt()	31
5.5.1.5 init_idt()	32
5.5.2 Variable Documentation	32
5.5.2.1 access	32
5.5.2.2 base	32
5.5.2.3 base_high	32
5.5.2.4 base_low	32
5.5.2.5 base_mid	33
5.5.2.6 flags	33
5.5.2.7 limit	33
5.5.2.8 limit_low	33
5.5.2.9 sselect	33
5.5.2.10 zero	33
5.6 include/mem/heap.h File Reference	34
5.6.1 Macro Definition Documentation	34
5.6.1.1 KHEAP_BASE	34
5.6.1.2 KHEAP_MIN	34
5.6.1.3 KHEAP_SIZE	35
5.6.1.4 TABLE_SIZE	35
5.6.2 Function Documentation	35
5.6.2.1 _kmalloc()	35
5.6.2.2 alloc()	36
5.6.2.3 init_kheap()	36
5.6.2.4 kfree()	36
5.6.2.5 kmalloc()	36
5.6.2.6 make_heap()	36
5.7 include/mem/paging.h File Reference	37
5.7.1 Macro Definition Documentation	37
5.7.1.1 PAGE_SIZE	37
5.7.2 Function Documentation	37
5.7.2.1 clear_bit()	38
5.7.2.2 first_free()	38
5.7.2.3 get_bit()	38
5.7.2.4 get_page()	38
5.7.2.5 init_paging()	39
5.7.2.6 load_page_dir()	39
5.7.2.7 new_frame()	39
5.7.2.8 set_bit()	40
5.8 include/string.h File Reference	40
5.8.1 Function Documentation	40

5.8.1.1 atoi()	41
5.8.1.2 isspace()	41
5.8.1.3 memset()	41
5.8.1.4 strcat()	42
5.8.1.5 strcmp()	42
5.8.1.6 strcpy()	42
5.8.1.7 strlen()	42
5.8.1.8 strtok()	43
5.9 include/system.h File Reference	43
5.9.1 Macro Definition Documentation	44
5.9.1.1 asm	44
5.9.1.2 cli	44
5.9.1.3 GDT_CS_ID	44
5.9.1.4 GDT_DS_ID	45
5.9.1.5 hlt	45
5.9.1.6 iret	45
5.9.1.7 no_warn	45
5.9.1.8 nop	45
5.9.1.9 NULL	45
5.9.1.10 sti	46
5.9.1.11 volatile	46
5.9.2 Typedef Documentation	46
5.9.2.1 size_t	46
5.9.2.2 u16int	46
5.9.2.3 u32int	46
5.9.2.4 u8int	46
5.9.3 Function Documentation	47
5.9.3.1 klogv()	47
5.9.3.2 kpanic()	47
5.10 kernel/core/interrupts.c File Reference	47
5.10.1 Macro Definition Documentation	48
5.10.1.1 ICW1	49
5.10.1.2 ICW4	49
5.10.1.3 io_wait	49
5.10.1.4 PIC1	49
5.10.1.5 PIC2	49
5.10.2 Function Documentation	49
5.10.2.1 bounds()	49
5.10.2.2 breakpoint()	50
5.10.2.3 coprocessor()	50
5.10.2.4 coprocessor_segment()	50
5.10.2.5 debug()	50

5.10.2.6 device_not_available()	50
5.10.2.7 divide_error()	50
5.10.2.8 do_bounds()	50
5.10.2.9 do_breakpoint()	51
5.10.2.10 do_coprocessor()	51
5.10.2.11 do_coprocessor_segment()	51
5.10.2.12 do_debug()	51
5.10.2.13 do_device_not_available()	51
5.10.2.14 do_divide_error()	52
5.10.2.15 do_double_fault()	52
5.10.2.16 do_general_protection()	52
5.10.2.17 do_invalid_op()	52
5.10.2.18 do_invalid_tss()	52
5.10.2.19 do_isr()	53
5.10.2.20 do_nmi()	53
5.10.2.21 do_overflow()	53
5.10.2.22 do_page_fault()	53
5.10.2.23 do_reserved()	53
5.10.2.24 do_segment_not_present()	54
5.10.2.25 do_stack_segment()	54
5.10.2.26 double_fault()	54
5.10.2.27 general_protection()	54
5.10.2.28 init_irq()	54
5.10.2.29 init_pic()	55
5.10.2.30 invalid_op()	55
5.10.2.31 invalid_tss()	55
5.10.2.32 isr0()	55
5.10.2.33 nmi()	55
5.10.2.34 overflow()	56
5.10.2.35 page_fault()	56
5.10.2.36 reserved()	56
5.10.2.37 rtc_isr()	56
5.10.2.38 segment_not_present()	56
5.10.2.39 stack_segment()	56
5.10.3 Variable Documentation	56
5.10.3.1 idt_entries	56
5.11 kernel/core/kmain.c File Reference	57
5.11.1 Function Documentation	57
5.11.1.1 kmain()	57
5.12 kernel/core/serial.c File Reference	58
5.12.1 Macro Definition Documentation	59
5.12.1.1 NO_ERROR	59

5.12.2 Function Documentation	59
5.12.2.1 init_serial()	59
5.12.2.2 polling()	59
5.12.2.3 serial_print()	61
5.12.2.4 serial_println()	62
5.12.2.5 set_serial_in()	62
5.12.2.6 set_serial_out()	62
5.12.3 Variable Documentation	62
5.12.3.1 serial_port_in	62
5.12.3.2 serial_port_out	63
5.13 kernel/core/system.c File Reference	63
5.13.1 Function Documentation	63
5.13.1.1 klogv()	63
5.13.1.2 kpanic()	63
5.14 kernel/core/tables.c File Reference	64
5.14.1 Function Documentation	64
5.14.1.1 gdt_init_entry()	64
5.14.1.2 idt_set_gate()	65
5.14.1.3 init_gdt()	65
5.14.1.4 init_idt()	65
5.14.1.5 write_gdt_ptr()	65
5.14.1.6 write_idt_ptr()	66
5.14.2 Variable Documentation	66
5.14.2.1 gdt_entries	66
5.14.2.2 gdt_ptr	66
5.14.2.3 idt_entries	66
5.14.2.4 idt_ptr	66
5.15 kernel/mem/heap.c File Reference	66
5.15.1 Function Documentation	67
5.15.1.1 _kmalloc()	67
5.15.1.2 alloc()	68
5.15.1.3 kmalloc()	68
5.15.1.4 make_heap()	68
5.15.2 Variable Documentation	68
5.15.2.1 __end	68
5.15.2.2 _end	69
5.15.2.3 curr_heap	69
5.15.2.4 end	69
5.15.2.5 kdir	69
5.15.2.6 kheap	69
5.15.2.7 phys_alloc_addr	69
5.16 kernel/mem/paging.c File Reference	70

5.16.1 Function Documentation	70
5.16.1.1 clear_bit()	70
5.16.1.2 find_free()	71
5.16.1.3 get_bit()	71
5.16.1.4 get_page()	71
5.16.1.5 init_paging()	72
5.16.1.6 load_page_dir()	72
5.16.1.7 new_frame()	72
5.16.1.8 set_bit()	73
5.16.2 Variable Documentation	73
5.16.2.1 cdir	73
5.16.2.2 frames	73
5.16.2.3 kdir	73
5.16.2.4 kheap	74
5.16.2.5 mem_size	74
5.16.2.6 nframes	74
5.16.2.7 page_size	74
5.16.2.8 phys_alloc_addr	74
5.17 lib/string.c File Reference	74
5.17.1 Function Documentation	75
5.17.1.1 atoi()	75
5.17.1.2 isspace()	75
5.17.1.3 memset()	76
5.17.1.4 strcat()	76
5.17.1.5 strcmp()	76
5.17.1.6 strcpy()	77
5.17.1.7 strlen()	77
5.17.1.8 strtok()	77
5.18 modules/mpx_supt.c File Reference	78
5.18.1 Function Documentation	78
5.18.1.1 idle()	78
5.18.1.2 mpx_init()	79
5.18.1.3 sys_alloc_mem()	79
5.18.1.4 sys_free_mem()	79
5.18.1.5 sys_req()	79
5.18.1.6 sys_set_free()	80
5.18.1.7 sys_set_malloc()	80
5.18.2 Variable Documentation	80
5.18.2.1 current_module	81
5.18.2.2 params	81
5.18.2.3 student_free	81
5.18.2.4 student_malloc	81

5.19 modules/mpx_supt.h File Reference	81
5.19.1 Macro Definition Documentation	82
5.19.1.1 COM_PORT	82
5.19.1.2 DEFAULT_DEVICE	82
5.19.1.3 EXIT	83
5.19.1.4 FALSE	83
5.19.1.5 IDLE	83
5.19.1.6 INVALID_BUFFER	83
5.19.1.7 INVALID_COUNT	83
5.19.1.8 INVALID_OPERATION	83
5.19.1.9 IO_MODULE	84
5.19.1.10 MEM_MODULE	84
5.19.1.11 MODULE_F	84
5.19.1.12 MODULE_R1	84
5.19.1.13 MODULE_R2	84
5.19.1.14 MODULE_R3	84
5.19.1.15 MODULE_R4	85
5.19.1.16 MODULE_R5	85
5.19.1.17 READ	85
5.19.1.18 TRUE	85
5.19.1.19 WRITE	85
5.19.2 Function Documentation	85
5.19.2.1 idle()	86
5.19.2.2 mpx_init()	86
5.19.2.3 sys_alloc_mem()	86
5.19.2.4 sys_free_mem()	86
5.19.2.5 sys_req()	87
5.19.2.6 sys_set_free()	87
5.19.2.7 sys_set_malloc()	87
5.20 modules/R1/commhand.c File Reference	88
5.20.1 Function Documentation	88
5.20.1.1 commhand()	88
5.21 modules/R1/commhand.h File Reference	89
5.21.1 Function Documentation	89
5.21.1.1 commhand()	89
5.22 modules/R1/R1commands.c File Reference	90
5.22.1 Function Documentation	90
5.22.1.1 BCDtoChar()	91
5.22.1.2 change_int_to_binary()	91
5.22.1.3 getDate()	91
5.22.1.4 getTime()	92
5.22.1.5 help()	92

5.22.1.6 intToBCD()	93
5.22.1.7 setDate()	93
5.22.1.8 setTime()	95
5.22.1.9 version()	97
5.23 modules/R1/R1commands.h File Reference	97
5.23.1 Function Documentation	97
5.23.1.1 BCDtoChar()	98
5.23.1.2 change_int_to_binary()	98
5.23.1.3 getDate()	98
5.23.1.4 getTime()	99
5.23.1.5 help()	99
5.23.1.6 setDate()	100
5.23.1.7 setTime()	102
5.23.1.8 version()	104
5.24 README.md File Reference	104
Index	105

Chapter 1

MPX-Fall2020-Group9

WVU CS 450 MPX Project files Making operating system// test message

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

date_time	7
footer	9
gdt_descriptor_struct	9
gdt_entry_struct	10
header	12
heap	13
idt_entry_struct	14
idt_struct	15
index_entry	16
index_table	17
page_dir	18
page_entry	19
page_table	20
param	21

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

include/string.h	40
include/system.h	43
include/core/asm.h	23
include/core/interrupts.h	23
include/core/io.h	24
include/core/serial.h	25
include/core/tables.h	30
include/mem/heap.h	34
include/mem/paging.h	37
kernel/core/interrupts.c	47
kernel/core/kmain.c	57
kernel/core/serial.c	58
kernel/core/system.c	63
kernel/core/tables.c	64
kernel/mem/heap.c	66
kernel/mem/paging.c	70
lib/string.c	74
modules/mpx_supt.c	78
modules/mpx_supt.h	81
modules/R1/commhand.c	88
modules/R1/commhand.h	89
modules/R1/R1commands.c	90
modules/R1/R1commands.h	97

Chapter 4

Class Documentation

4.1 date_time Struct Reference

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

Public Attributes

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

4.1.1 Detailed Description

Definition at line 30 of file system.h.

4.1.2 Member Data Documentation

4.1.2.1 day_m

```
int date_time::day_m
```

Definition at line 35 of file system.h.

4.1.2.2 day_w

```
int date_time::day_w
```

Definition at line 34 of file system.h.

4.1.2.3 day_y

```
int date_time::day_y
```

Definition at line 36 of file system.h.

4.1.2.4 hour

```
int date_time::hour
```

Definition at line 33 of file system.h.

4.1.2.5 min

```
int date_time::min
```

Definition at line 32 of file system.h.

4.1.2.6 mon

```
int date_time::mon
```

Definition at line 37 of file system.h.

4.1.2.7 sec

```
int date_time::sec
```

Definition at line 31 of file system.h.

4.1.2.8 year

```
int date_time::year
```

Definition at line 38 of file system.h.

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

- include/[system.h](#)

4.2 footer Struct Reference

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

Public Attributes

- [header head](#)

4.2.1 Detailed Description

Definition at line 16 of file heap.h.

4.2.2 Member Data Documentation

4.2.2.1 head

```
header footer::head
```

Definition at line 17 of file heap.h.

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

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

4.3 gdt_descriptor_struct Struct Reference

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

Public Attributes

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

4.3.1 Detailed Description

Definition at line 23 of file tables.h.

4.3.2 Member Data Documentation

4.3.2.1 base

```
u32int gdt_descriptor_struct::base
```

Definition at line 26 of file tables.h.

4.3.2.2 limit

```
u16int gdt_descriptor_struct::limit
```

Definition at line 25 of file tables.h.

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

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

4.4 gdt_entry_struct Struct Reference

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

Public Attributes

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

4.4.1 Detailed Description

Definition at line 30 of file tables.h.

4.4.2 Member Data Documentation

4.4.2.1 access

```
u8int gdt_entry_struct::access
```

Definition at line 35 of file tables.h.

4.4.2.2 base_high

```
u8int gdt_entry_struct::base_high
```

Definition at line 37 of file tables.h.

4.4.2.3 base_low

```
u16int gdt_entry_struct::base_low
```

Definition at line 33 of file tables.h.

4.4.2.4 base_mid

```
u8int gdt_entry_struct::base_mid
```

Definition at line 34 of file tables.h.

4.4.2.5 flags

```
u8int gdt_entry_struct::flags
```

Definition at line 36 of file tables.h.

4.4.2.6 limit_low

```
ul6int gdt_entry_struct::limit_low
```

Definition at line 32 of file tables.h.

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

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

4.5 header Struct Reference

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

Public Attributes

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

4.5.1 Detailed Description

Definition at line 11 of file heap.h.

4.5.2 Member Data Documentation

4.5.2.1 index_id

```
int header::index_id
```

Definition at line 13 of file heap.h.

4.5.2.2 size

```
int header::size
```

Definition at line 12 of file heap.h.

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

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

4.6 heap Struct Reference

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

Public Attributes

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

4.6.1 Detailed Description

Definition at line 33 of file heap.h.

4.6.2 Member Data Documentation

4.6.2.1 base

```
u32int heap::base
```

Definition at line 35 of file heap.h.

4.6.2.2 index

```
index\_table heap::index
```

Definition at line 34 of file heap.h.

4.6.2.3 max_size

```
u32int heap::max_size
```

Definition at line 36 of file heap.h.

4.6.2.4 min_size

```
u32int heap::min_size
```

Definition at line 37 of file heap.h.

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

- include/mem/heap.h

4.7 idt_entry_struct Struct Reference

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

Public Attributes

- u16int base_low
- u16int sselect
- u8int zero
- u8int flags
- u16int base_high

4.7.1 Detailed Description

Definition at line 6 of file tables.h.

4.7.2 Member Data Documentation

4.7.2.1 base_high

```
u16int idt_entry_struct::base_high
```

Definition at line 12 of file tables.h.

4.7.2.2 base_low

```
u16int idt_entry_struct::base_low
```

Definition at line 8 of file tables.h.

4.7.2.3 flags

```
u8int idt_entry_struct::flags
```

Definition at line 11 of file tables.h.

4.7.2.4 sselect

```
u16int idt_entry_struct::sselect
```

Definition at line 9 of file tables.h.

4.7.2.5 zero

```
u8int idt_entry_struct::zero
```

Definition at line 10 of file tables.h.

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

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

4.8 idt_struct Struct Reference

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

Public Attributes

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

4.8.1 Detailed Description

Definition at line 16 of file tables.h.

4.8.2 Member Data Documentation

4.8.2.1 base

```
u32int idt_struct::base
```

Definition at line 19 of file tables.h.

4.8.2.2 limit

```
ul6int idt_struct::limit
```

Definition at line 18 of file tables.h.

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

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

4.9 index_entry Struct Reference

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

Public Attributes

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

4.9.1 Detailed Description

Definition at line 20 of file heap.h.

4.9.2 Member Data Documentation

4.9.2.1 block

```
u32int index_entry::block
```

Definition at line 23 of file heap.h.

4.9.2.2 empty

```
int index_entry::empty
```

Definition at line 22 of file heap.h.

4.9.2.3 size

```
int index_entry::size
```

Definition at line 21 of file heap.h.

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

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

4.10 index_table Struct Reference

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

Public Attributes

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

4.10.1 Detailed Description

Definition at line 27 of file heap.h.

4.10.2 Member Data Documentation

4.10.2.1 id

```
int index_table::id
```

Definition at line 29 of file heap.h.

4.10.2.2 table

```
index_entry index_table::table[TABLE\_SIZE]
```

Definition at line 28 of file heap.h.

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

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

4.11 page_dir Struct Reference

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

Public Attributes

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

4.11.1 Detailed Description

Definition at line 34 of file paging.h.

4.11.2 Member Data Documentation

4.11.2.1 tables

```
page\_table* page\_dir::tables[1024]
```

Definition at line 35 of file paging.h.

4.11.2.2 tables_phys

```
u32int page\_dir::tables\_phys[1024]
```

Definition at line 36 of file paging.h.

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

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

4.12 page_entry Struct Reference

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

Public Attributes

- `u32int present`: 1
- `u32int writeable`: 1
- `u32int usermode`: 1
- `u32int accessed`: 1
- `u32int dirty`: 1
- `u32int reserved`: 7
- `u32int frameaddr`: 20

4.12.1 Detailed Description

Definition at line 12 of file `paging.h`.

4.12.2 Member Data Documentation

4.12.2.1 accessed

```
u32int page_entry::accessed
```

Definition at line 16 of file `paging.h`.

4.12.2.2 dirty

```
u32int page_entry::dirty
```

Definition at line 17 of file `paging.h`.

4.12.2.3 frameaddr

```
u32int page_entry::frameaddr
```

Definition at line 19 of file `paging.h`.

4.12.2.4 present

```
u32int page_entry::present
```

Definition at line 13 of file paging.h.

4.12.2.5 reserved

```
u32int page_entry::reserved
```

Definition at line 18 of file paging.h.

4.12.2.6 usermode

```
u32int page_entry::usermode
```

Definition at line 15 of file paging.h.

4.12.2.7 writeable

```
u32int page_entry::writeable
```

Definition at line 14 of file paging.h.

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

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

4.13 page_table Struct Reference

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

Public Attributes

- [page_entry pages](#) [1024]

4.13.1 Detailed Description

Definition at line 26 of file paging.h.

4.13.2 Member Data Documentation

4.13.2.1 pages

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

Definition at line 27 of file paging.h.

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

- include/mem/paging.h

4.14 param Struct Reference

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

Public Attributes

- int [op_code](#)
- int [device_id](#)
- char * [buffer_ptr](#)
- int * [count_ptr](#)

4.14.1 Detailed Description

Definition at line 31 of file mpx_supt.h.

4.14.2 Member Data Documentation

4.14.2.1 buffer_ptr

```
char* param::buffer_ptr
```

Definition at line 34 of file mpx_supt.h.

4.14.2.2 `count_ptr`

```
int* param::count_ptr
```

Definition at line 35 of file `mpx_supt.h`.

4.14.2.3 `device_id`

```
int param::device_id
```

Definition at line 33 of file `mpx_supt.h`.

4.14.2.4 `op_code`

```
int param::op_code
```

Definition at line 32 of file `mpx_supt.h`.

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

- [modules/mpx_supt.h](#)

Chapter 5

File Documentation

5.1 include/core/asm.h File Reference

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

5.2 include/core/interrupts.h File Reference

Functions

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

5.2.1 Function Documentation

5.2.1.1 [init_irq\(\)](#)

```
void init_irq (
    void )
```

Definition at line 66 of file interrupts.c.

```
67 {
68     int i;
69
70     // Necessary interrupt handlers for protected mode
71     u32int isrs[17] = {
72         (u32int)divide_error,
73         (u32int)debug,
74         (u32int)nmi,
75         (u32int)breakpoint,
76         (u32int)overflow,
77         (u32int)bounds,
78         (u32int)invalid_op,
79         (u32int)device_not_available,
80         (u32int)double_fault,
81         (u32int)coprocessor_segment,
```

```

82     (u32int)invalid_tss,
83     (u32int)segment_not_present,
84     (u32int)stack_segment,
85     (u32int)general_protection,
86     (u32int)page_fault,
87     (u32int)reserved,
88     (u32int)coprocessor
89 };
90
91 // Install handlers; 0x08=sel, 0x8e=flags
92 for(i=0; i<32; i++){
93     if (i<17) idt_set_gate(i, isrs[i], 0x08, 0x8e);
94     else idt_set_gate(i, (u32int)reserved, 0x08, 0x8e);
95 }
96 // Ignore interrupts from the real time clock
97 idt_set_gate(0x08, (u32int)rtc_isr, 0x08, 0x8e);
98 }

```

5.2.1.2 init_pic()

```

void init_pic (
    void )

```

Definition at line 106 of file interrupts.c.

```

107 {
108     outb(PIC1,ICW1);    //send initialization code words 1 to PIC1
109     io_wait();
110     outb(PIC2,ICW1);    //send icw1 to PIC2
111     io_wait();
112     outb(PIC1+1,0x20);  //icw2: remap irq0 to 32
113     io_wait();
114     outb(PIC2+1,0x28);  //icw2: remap irq8 to 40
115     io_wait();
116     outb(PIC1+1,4);     //icw3
117     io_wait();
118     outb(PIC2+1,2);     //icw3
119     io_wait();
120     outb(PIC1+1,ICW4);  //icw4: 80x86, automatic handling
121     io_wait();
122     outb(PIC2+1,ICW4);  //icw4: 80x86, automatic handling
123     io_wait();
124     outb(PIC1+1,0xFF);  //disable irqs for PIC1
125     io_wait();
126     outb(PIC2+1,0xFF);  //disable irqs for PIC2
127 }

```

5.3 include/core/io.h File Reference

Macros

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

5.3.1 Macro Definition Documentation

5.3.1.1 inb

```
#define inb(  
    port )
```

Value:

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

Definition at line 15 of file io.h.

5.3.1.2 outb

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

Definition at line 8 of file io.h.

5.4 include/core/serial.h File Reference

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)
- int * [polling](#) (char *buffer, int *count)

5.4.1 Macro Definition Documentation

5.4.1.1 COM1

```
#define COM1 0x3f8
```

Definition at line 4 of file serial.h.

5.4.1.2 COM2

```
#define COM2 0x2f8
```

Definition at line 5 of file serial.h.

5.4.1.3 COM3

```
#define COM3 0x3e8
```

Definition at line 6 of file serial.h.

5.4.1.4 COM4

```
#define COM4 0x2e8
```

Definition at line 7 of file serial.h.

5.4.2 Function Documentation

5.4.2.1 init_serial()

```
int init_serial (  
    int device )
```

Definition at line 23 of file serial.c.

```
24 {  
25     outb(device + 1, 0x00);           //disable interrupts  
26     outb(device + 3, 0x80);           //set line control register  
27     outb(device + 0, 115200 / 9600); //set bsd least sig bit  
28     outb(device + 1, 0x00);           //brd most significant bit  
29     outb(device + 3, 0x03);           //lock divisor; 8bits, no parity, one stop  
30     outb(device + 2, 0xC7);           //enable fifo, clear, 14byte threshold  
31     outb(device + 4, 0x0B);           //enable interrupts, rts/dsr set  
32     (void)inb(device);                //read bit to reset port  
33     return NO_ERROR;  
34 }
```

5.4.2.2 polling()

```
int* polling (
    char * buffer,
    int * count )
```

Definition at line 93 of file serial.c.

```
94 {
95     // insert your code to gather keyboard input via the technique of polling.
96
97     char keyboard_character;
98
99     int cursor = 0;
100
101     char log[] = {'\0', '\0', '\0', '\0'};
102
103     int characters_in_buffer = 0;
104
105     while (1)
106     {
107
108         if (inb(COM1 + 5) & 1)
109         {
110             // is there input char?
111             keyboard_character = inb(COM1); //read the char from COM1
112
113             if (keyboard_character == '\n' || keyboard_character == '\r')
114             { // HANDLING THE CARRIAGE RETURN AND NEW LINE CHARACTERS
115
116                 buffer[characters_in_buffer] = '\0';
117                 break;
118             }
119             else if ((keyboard_character == 127 || keyboard_character == 8) && cursor > 0)
120             { // HANDLING THE BACKSPACE CHARACTER
121
122                 //serial_println("Handleing backspace character.");
123                 serial_print("\033[K");
124
125                 buffer[cursor - 1] = '\0';
126                 serial_print("\b \b");
127                 serial_print(buffer + cursor);
128                 cursor--;
129
130                 int temp_cursor = cursor;
131
132                 while (buffer[temp_cursor + 1] != '\0')
133                 {
134                     buffer[temp_cursor] = buffer[temp_cursor + 1];
135                     buffer[temp_cursor + 1] = '\0';
136                     temp_cursor++;
137                 }
138
139                 characters_in_buffer--;
140                 cursor = characters_in_buffer;
141             }
142             else if (keyboard_character == '~' && cursor < 99)
143             { //HANDLING THE DELETE KEY
144                 // \033[3~
145
146                 serial_print("\033[K");
147
148                 buffer[cursor + 1] = '\0';
149                 serial_print("\b \b");
150                 serial_print(buffer + cursor);
151
152                 int temp_cursor = cursor + 1;
153
154                 while (buffer[temp_cursor + 1] != '\0')
155                 {
156                     buffer[temp_cursor] = buffer[temp_cursor + 1];
157                     buffer[temp_cursor + 1] = '\0';
158                     temp_cursor++;
159                 }
160
161                 characters_in_buffer--;
162                 cursor = characters_in_buffer;
163             }
164             else if (keyboard_character == '\033')
165             { // HANDLING FIRST CHARACTER FOR ARROW KEYS
166
167                 log[0] = keyboard_character;
168             }
169             else if (keyboard_character == '[' && log[0] == '\033')
170             { // HANDLING SECOND CHARACTER FOR ARROW KEYS
```

```

171     log[1] = keyboard_character;
172 }
173 else if (log[0] == '\033' && log[1] == '[')
174 { // HANDLING LAST CHARACTER FOR ARROW KEYS
175     log[2] = keyboard_character;
176
177     if (keyboard_character == 'A')
178     { //Up arrow
179         //Call a history function from the commhand or do nothing
180     }
181     else if (keyboard_character == 'B')
182     { //Down arrow
183         //Call a history command from the commhand or do nothing
184     }
185     else if (keyboard_character == 'C' && cursor != 99)
186     { //Right arrow
187
188         serial_print("\033[C");
189         cursor++;
190     }
191     else if (keyboard_character == 'D' && cursor != 0)
192     { //Left arrow
193
194         serial_print("\033[D");
195         cursor--;
196     }
197
198     memset(log, '\0', 4);
199 }
200 else
201 {
202
203     if (cursor == 0 && buffer[cursor] == '\0') //Adding character at beginning of buffer
204     {
205         buffer[cursor] = keyboard_character;
206         serial_print(&keyboard_character);
207         cursor++;
208     }
209     else if (buffer[cursor] == '\0') //Adding character at the end of the buffer
210     {
211         buffer[cursor] = keyboard_character;
212         serial_print(&keyboard_character);
213         cursor++;
214     }
215     else //Inserting character to the middle of the buffer
216     {
217         char temp_buffer[strlen(buffer)];
218         memset(temp_buffer, '\0', strlen(buffer));
219
220         int temp_cursor = 0;
221         while (temp_cursor <= characters_in_buffer) //Filling the temp_buffer with all of the
characters from buffer, and inserting the new character.
222         {
223             if (temp_cursor < cursor)
224             {
225                 temp_buffer[temp_cursor] = buffer[temp_cursor];
226             }
227             else if (temp_cursor > cursor)
228             {
229                 temp_buffer[temp_cursor] = buffer[temp_cursor - 1];
230             }
231             else
232             { //temp_cursor == cursor
233                 temp_buffer[temp_cursor] = keyboard_character;
234             }
235             temp_cursor++;
236         }
237
238         temp_cursor = 0;
239         int temp_buffer_size = strlen(temp_buffer);
240         while (temp_cursor <= temp_buffer_size) //Setting the contents of the buffer equal to the
temp_buffer.
241         {
242             buffer[temp_cursor] = temp_buffer[temp_cursor];
243             temp_cursor++;
244         }
245
246         serial_print("\033[K");
247         serial_print(&keyboard_character);
248         serial_print(buffer + cursor + 1);
249         cursor++;
250     }
251     characters_in_buffer++;
252 }
253 }
254 }
255

```



```
256  *count = characters_in_buffer; // buffer count
257
258  return count;
259 }
```

5.4.2.3 serial_print()

```
int serial_print (
    const char * msg )
```

Definition at line 57 of file serial.c.

```
58 {
59     int i;
60     for (i = 0; *(i + msg) != '\0'; i++)
61     {
62         outb(serial_port_out, *(i + msg));
63     }
64     if (*msg == '\r')
65         outb(serial_port_out, '\n');
66     return NO_ERROR;
67 }
```

5.4.2.4 serial_println()

```
int serial_println (
    const char * msg )
```

Definition at line 41 of file serial.c.

```
42 {
43     int i;
44     for (i = 0; *(i + msg) != '\0'; i++)
45     {
46         outb(serial_port_out, *(i + msg));
47     }
48     outb(serial_port_out, '\r');
49     outb(serial_port_out, '\n');
50     return NO_ERROR;
51 }
```

5.4.2.5 set_serial_in()

```
int set_serial_in (
    int device )
```

Definition at line 87 of file serial.c.

```
88 {
89     serial_port_in = device;
90     return NO_ERROR;
91 }
```

5.4.2.6 set_serial_out()

```
int set_serial_out (
    int device )
```

Definition at line 75 of file serial.c.

```
76 {
77     serial_port_out = device;
78     return NO_ERROR;
79 }
```

5.5 include/core/tables.h File Reference

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

Classes

- 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

5.5.1 Function Documentation

5.5.1.1 `__attribute__()`

```
struct gdt_entry_struct __attribute__ (
    (packed) )
```

5.5.1.2 `gdt_init_entry()`

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

Definition at line 57 of file tables.c.

```
59 {
60     gdt_entry *new_entry = &gdt_entries[idx];
61     new_entry->base_low = (base & 0xFFFF);
62     new_entry->base_mid = (base >> 16) & 0xFF;
63     new_entry->base_high = (base >> 24) & 0xFF;
64     new_entry->limit_low = (limit & 0xFFFF);
65     new_entry->flags = (limit >> 16) & 0xFF;
66     new_entry->flags |= flags & 0xF0;
67     new_entry->access = access;
68 }
```

5.5.1.3 `idt_set_gate()`

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

Definition at line 27 of file tables.c.

```
29 {
30     idt_entry *new_entry = &idt_entries[idx];
31     new_entry->base_low = (base & 0xFFFF);
32     new_entry->base_high = (base >> 16) & 0xFFFF;
33     new_entry->sselect = sel;
34     new_entry->zero = 0;
35     new_entry->flags = flags;
36 }
```

5.5.1.4 `init_gdt()`

```
void init_gdt ( )
```

Definition at line 75 of file tables.c.

```
76 {
77     gdt_ptr.limit = 5 * sizeof(gdt_entry) - 1;
78     gdt_ptr.base = (u32int) gdt_entries;
79
80     u32int limit = 0xFFFFFFFF;
81     gdt_init_entry(0, 0, 0, 0, 0); //required null segment
82     gdt_init_entry(1, 0, limit, 0x9A, 0xCF); //code segment
83     gdt_init_entry(2, 0, limit, 0x92, 0xCF); //data segment
84     gdt_init_entry(3, 0, limit, 0xFA, 0xCF); //user mode code segment
85     gdt_init_entry(4, 0, limit, 0xF2, 0xCF); //user mode data segment
86
87     write_gdt_ptr((u32int) &gdt_ptr, sizeof(gdt_ptr));
88 }
```

5.5.1.5 init_idt()

```
void init_idt ( )
```

Definition at line 43 of file tables.c.

```
44 {  
45     idt_ptr.limit = 256*sizeof(idt_descriptor) - 1;  
46     idt_ptr.base  = (u32int)idt_entries;  
47     memset(idt_entries, 0, 256*sizeof(idt_descriptor));  
48  
49     write_idt_ptr((u32int)&idt_ptr);  
50 }
```

5.5.2 Variable Documentation

5.5.2.1 access

```
u8int access
```

Definition at line 3 of file tables.h.

5.5.2.2 base

```
u32int base
```

Definition at line 1 of file tables.h.

5.5.2.3 base_high

```
u8int base_high
```

Definition at line 4 of file tables.h.

5.5.2.4 base_low

```
u16int base_low
```

Definition at line 0 of file tables.h.

5.5.2.5 base_mid

```
u8int base_mid
```

Definition at line 2 of file tables.h.

5.5.2.6 flags

```
u8int flags
```

Definition at line 3 of file tables.h.

5.5.2.7 limit

```
u16int limit
```

Definition at line 0 of file tables.h.

5.5.2.8 limit_low

```
u16int limit_low
```

Definition at line 0 of file tables.h.

5.5.2.9 sselect

```
u16int sselect
```

Definition at line 1 of file tables.h.

5.5.2.10 zero

```
u8int zero
```

Definition at line 2 of file tables.h.

5.6 include/mem/heap.h File Reference

Classes

- struct [header](#)
- struct [footer](#)
- struct [index_entry](#)
- struct [index_table](#)
- struct [heap](#)

Macros

- #define [TABLE_SIZE](#) 0x1000
- #define [KHEAP_BASE](#) 0xD000000
- #define [KHEAP_MIN](#) 0x10000
- #define [KHEAP_SIZE](#) 0x1000000

Functions

- [u32int_kmalloc](#) ([u32int](#) size, int align, [u32int](#) *phys_addr)
- [u32int_kmalloc](#) ([u32int](#) size)
- [u32int_kfree](#) ()
- void [init_kheap](#) ()
- [u32int_alloc](#) ([u32int](#) size, [heap](#) *hp, int align)
- [heap](#) * [make_heap](#) ([u32int](#) base, [u32int](#) max, [u32int](#) min)

5.6.1 Macro Definition Documentation

5.6.1.1 KHEAP_BASE

```
#define KHEAP_BASE 0xD000000
```

Definition at line 6 of file heap.h.

5.6.1.2 KHEAP_MIN

```
#define KHEAP_MIN 0x10000
```

Definition at line 7 of file heap.h.

5.6.1.3 KHEAP_SIZE

```
#define KHEAP_SIZE 0x1000000
```

Definition at line 8 of file heap.h.

5.6.1.4 TABLE_SIZE

```
#define TABLE_SIZE 0x1000
```

Definition at line 5 of file heap.h.

5.6.2 Function Documentation

5.6.2.1 _kmalloc()

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

Definition at line 24 of file heap.c.

```
25 {
26     u32int *addr;
27
28     // Allocate on the kernel heap if one has been created
29     if (kheap != 0){
30         addr = (u32int*)alloc(size, kheap, page_align);
31         if (phys_addr){
32             page_entry *page = get_page((u32int)addr, kdir, 0);
33             *phys_addr = (page->frameaddr*0x1000) + ((u32int)addr & 0xFFF);
34         }
35         return (u32int)addr;
36     }
37     // Else, allocate directly from physical memory
38     else {
39         if (page_align && (phys_alloc_addr & 0xFFFFF000)){
40             phys_alloc_addr &= 0xFFFFF000;
41             phys_alloc_addr += 0x1000;
42         }
43         addr = (u32int*)phys_alloc_addr;
44         if (phys_addr){
45             *phys_addr = phys_alloc_addr;
46         }
47         phys_alloc_addr += size;
48         return (u32int)addr;
49     }
50 }
```

5.6.2.2 alloc()

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

Definition at line 57 of file heap.c.

```
58 {
59     no_warn(size||align||h);
60     static u32int heap_addr = KHEAP_BASE;
61
62     u32int base = heap_addr;
63     heap_addr += size;
64
65     if (heap_addr > KHEAP_BASE + KHEAP_MIN)
66         serial_println("Heap is full!");
67
68     return base;
69 }
```

5.6.2.3 init_kheap()

```
void init_kheap ( )
```

5.6.2.4 kfree()

```
u32int kfree ( )
```

5.6.2.5 kmalloc()

```
u32int kmalloc (
    u32int size )
```

Definition at line 52 of file heap.c.

```
53 {
54     return _kmalloc(size,0,0);
55 }
```

5.6.2.6 make_heap()

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

Definition at line 71 of file heap.c.

```
72 {
73     no_warn(base||max||min);
74     return (heap*) kmalloc(sizeof(heap));
75 }
```


5.7 include/mem/paging.h File Reference

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

Classes

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

Macros

- `#define` [PAGE_SIZE](#) 0x1000

Functions

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

5.7.1 Macro Definition Documentation

5.7.1.1 PAGE_SIZE

```
#define PAGE_SIZE 0x1000
```

Definition at line 6 of file paging.h.

5.7.2 Function Documentation

5.7.2.1 clear_bit()

```
void clear_bit (
    u32int addr )
```

Definition at line 44 of file paging.c.

```
45 {
46     u32int frame = addr/page_size;
47     u32int index = frame/32;
48     u32int offset = frame%32;
49     frames[index] &= ~(1 « offset);
50 }
```

5.7.2.2 first_free()

```
u32int first_free ( )
```

5.7.2.3 get_bit()

```
u32int get_bit (
    u32int addr )
```

Definition at line 56 of file paging.c.

```
57 {
58     u32int frame = addr/page_size;
59     u32int index = frame/32;
60     u32int offset = frame%32;
61     return (frames[index] & (1 « offset));
62 }
```

5.7.2.4 get_page()

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

Definition at line 85 of file paging.c.

```
86 {
87     u32int phys_addr;
88     u32int index = addr / page_size / 1024;
89     u32int offset = addr / page_size % 1024;
90
91     //return it if it exists
92     if (dir->tables[index])
93         return &dir->tables[index]->pages[offset];
94
95     //create it
96     else if (make_table){
97         dir->tables[index] = (page_table*)_kmalloc(sizeof(page_table), 1, &phys_addr);
98         dir->tables_phys[index] = phys_addr | 0x7; //enable present, writable
99         return &dir->tables[index]->pages[offset];
100     }
101     else return 0;
102 }
```

5.7.2.5 init_paging()

```
void init_paging ( )
```

Definition at line 111 of file paging.c.

```
112 {
113     //create frame bitmap
114     nframes = (u32int) (mem_size/page_size);
115     frames = (u32int*) kmalloc(nframes/32);
116     memset(frames, 0, nframes/32);
117
118     //create kernel directory
119     kdir = (page_dir*) _kmalloc(sizeof(page_dir), 1, 0); //page aligned
120     memset(kdir, 0, sizeof(page_dir));
121
122     //get pages for kernel heap
123     u32int i = 0x0;
124     for(i=KHEAP_BASE; i<(KHEAP_BASE+KHEAP_MIN); i+=1){
125         get_page(i, kdir, 1);
126     }
127
128     //perform identity mapping of used memory
129     //note: placement_addr gets incremented in get_page,
130     //so we're mapping the first frames as well
131     i = 0x0;
132     while (i < (phys_alloc_addr+0x10000)){
133         new_frame(get_page(i, kdir, 1));
134         i += page_size;
135     }
136
137     //allocate heap frames now that the placement addr has increased.
138     //placement addr increases here for heap
139     for(i=KHEAP_BASE; i<(KHEAP_BASE+KHEAP_MIN); i+=PAGE_SIZE){
140         new_frame(get_page(i, kdir, 1));
141     }
142
143     //load the kernel page directory; enable paging
144     load_page_dir(kdir);
145
146     //setup the kernel heap
147     kheap = make_heap(KHEAP_BASE, KHEAP_SIZE, KHEAP_BASE+KHEAP_MIN);
148 }
```

5.7.2.6 load_page_dir()

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

Definition at line 158 of file paging.c.

```
159 {
160     cdir = new_dir;
161     asm volatile ("mov %0,%%cr3:: "b"(&cdir->tables_phys[0]));
162     u32int cr0;
163     asm volatile ("mov %%cr0,%0: "=b"(cr0));
164     cr0 |= 0x80000000;
165     asm volatile ("mov %0,%%cr0:: "b"(cr0));
166 }
```

5.7.2.7 new_frame()

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

Definition at line 173 of file paging.c.

```
174 {
175     u32int index;
```

```

176  if (page->frameaddr != 0) return;
177  if ( (u32int)(-1) == (index=find_free()) ) kpanic("Out of memory");
178
179  //mark a frame as in-use
180  set_bit(index*page_size);
181  page->present = 1;
182  page->frameaddr = index;
183  page->writeable = 1;
184  page->usermode = 0;
185 }

```

5.7.2.8 set_bit()

```

void set_bit (
    u32int addr )

```

Definition at line 32 of file paging.c.

```

33 {
34     u32int frame = addr/page_size;
35     u32int index = frame/32;
36     u32int offset = frame%32;
37     frames[index] |= (1 « offset);
38 }

```

5.8 include/string.h File Reference

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

Functions

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

5.8.1 Function Documentation

5.8.1.1 atoi()

```
int atoi (
    const char * s )
```

Definition at line 48 of file string.c.

```
49 {
50     int res=0;
51     int charVal=0;
52     char sign = ' ';
53     char c = *s;
54
55
56     while(isspace(&c)){ ++s; c = *s;} // advance past whitespace
57
58
59     if (*s == '-' || *s == '+') sign = *(s++); // save the sign
60
61
62     while(*s != '\0'){
63         charVal = *s - 48;
64         res = res * 10 + charVal;
65         s++;
66     }
67
68
69     if ( sign == '-') res=res * -1;
70
71
72     return res; // return integer
73 }
```

5.8.1.2 isspace()

```
int isspace (
    const char * c )
```

Definition at line 119 of file string.c.

```
120 {
121     if (*c == ' ' ||
122         *c == '\n' ||
123         *c == '\r' ||
124         *c == '\f' ||
125         *c == '\t' ||
126         *c == '\v'){
127         return 1;
128     }
129     return 0;
130 }
```

5.8.1.3 memset()

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

Definition at line 137 of file string.c.

```
138 {
139     unsigned char *p = (unsigned char *) s;
140     while(n--){
141         *p++ = (unsigned char) c;
142     }
143     return s;
144 }
```

5.8.1.4 strcat()

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

Definition at line 106 of file string.c.

```
107 {
108     char *rc = s1;
109     if (*s1) while(++s1);
110     while( (*s1++ = *s2++) );
111     return rc;
112 }
```

5.8.1.5 strcmp()

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

Definition at line 79 of file string.c.

```
80 {
81
82     // Remarks:
83     // 1) If we made it to the end of both strings (i. e. our pointer points to a
84     // '\0' character), the function will return 0
85     // 2) If we didn't make it to the end of both strings, the function will
86     // return the difference of the characters at the first index of
87     // indifference.
88     while ( (*s1) && (*s1==*s2) ){
89         ++s1;
90         ++s2;
91     }
92     return ( *(unsigned char *)s1 - *(unsigned char *)s2 );
93 }
```

5.8.1.6 strcpy()

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

Definition at line 36 of file string.c.

```
37 {
38     char *rc = s1;
39     while( (*s1++ = *s2++) );
40     return rc; // return pointer to destination string
41 }
```

5.8.1.7 strlen()

```
int strlen (
    const char * s )
```

Definition at line 24 of file string.c.

```
25 {
26     int r1 = 0;
27     if (*s) while(*s++) r1++;
28     return r1; //return length of string
29 }
```

5.8.1.8 strtok()

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

Definition at line 151 of file string.c.

```
152 {
153     static char *tok_tmp = NULL;
154     const char *p = s2;
155
156     //new string
157     if (s1!=NULL){
158         tok_tmp = s1;
159     }
160     //old string cont'd
161     else {
162         if (tok_tmp==NULL){
163             return NULL;
164         }
165         s1 = tok_tmp;
166     }
167
168     //skip leading s2 characters
169     while ( *p && *s1 ){
170         if (*s1==*p){
171             ++s1;
172             p = s2;
173             continue;
174         }
175         ++p;
176     }
177
178     //no more to parse
179     if (!*s1){
180         return (tok_tmp = NULL);
181     }
182
183     //skip non-s2 characters
184     tok_tmp = s1;
185     while (*tok_tmp){
186         p = s2;
187         while (*p){
188             if (*tok_tmp==*p++){
189                 *tok_tmp++ = '\0';
190                 return s1;
191             }
192         }
193         ++tok_tmp;
194     }
195
196     //end of string
197     tok_tmp = NULL;
198     return s1;
199 }
```

5.9 include/system.h File Reference

Classes

- struct [date_time](#)

Macros

- #define [NULL](#) 0
- #define [no_warn](#)(p) if (p) while (1) break
- #define [asm](#) __asm__
- #define [volatile](#) __volatile__
- #define [sti](#)() [asm volatile](#) ("sti::")

- `#define cli() asm volatile ("cli::")`
- `#define nop() asm volatile ("nop::")`
- `#define hlt() asm volatile ("hlt::")`
- `#define iret() asm volatile ("iret::")`
- `#define GDT_CS_ID 0x01`
- `#define GDT_DS_ID 0x02`

Typedefs

- `typedef unsigned int size_t`
- `typedef unsigned char u8int`
- `typedef unsigned short u16int`
- `typedef unsigned long u32int`

Functions

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

5.9.1 Macro Definition Documentation

5.9.1.1 asm

```
#define asm __asm__
```

Definition at line 11 of file system.h.

5.9.1.2 cli

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

Definition at line 15 of file system.h.

5.9.1.3 GDT_CS_ID

```
#define GDT_CS_ID 0x01
```

Definition at line 20 of file system.h.

5.9.1.4 GDT_DS_ID

```
#define GDT_DS_ID 0x02
```

Definition at line 21 of file system.h.

5.9.1.5 hlt

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

Definition at line 17 of file system.h.

5.9.1.6 iret

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

Definition at line 18 of file system.h.

5.9.1.7 no_warn

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

Definition at line 7 of file system.h.

5.9.1.8 nop

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

Definition at line 16 of file system.h.

5.9.1.9 NULL

```
#define NULL 0
```

Definition at line 4 of file system.h.

5.9.1.10 sti

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

Definition at line 14 of file system.h.

5.9.1.11 volatile

```
#define volatile __volatile__
```

Definition at line 12 of file system.h.

5.9.2 Typedef Documentation

5.9.2.1 size_t

```
typedef unsigned int size_t
```

Definition at line 24 of file system.h.

5.9.2.2 u16int

```
typedef unsigned short u16int
```

Definition at line 26 of file system.h.

5.9.2.3 u32int

```
typedef unsigned long u32int
```

Definition at line 27 of file system.h.

5.9.2.4 u8int

```
typedef unsigned char u8int
```

Definition at line 25 of file system.h.

5.9.3 Function Documentation

5.9.3.1 klogv()

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

Definition at line 11 of file system.c.

```
12 {
13     char logmsg[64] = {'\0'}, prefix[] = "klogv: ";
14     strcat(logmsg, prefix);
15     strcat(logmsg, msg);
16     serial_println(logmsg);
17 }
```

5.9.3.2 kpanic()

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

Definition at line 24 of file system.c.

```
25 {
26     cli(); //disable interrupts
27     char logmsg[64] = {'\0'}, prefix[] = "Panic: ";
28     strcat(logmsg, prefix);
29     strcat(logmsg, msg);
30     klogv(logmsg);
31     hlt(); //halt
32 }
```

5.10 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>
```

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 [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]

5.10.1 Macro Definition Documentation

5.10.1.1 ICW1

```
#define ICW1 0x11
```

Definition at line 20 of file interrupts.c.

5.10.1.2 ICW4

```
#define ICW4 0x01
```

Definition at line 21 of file interrupts.c.

5.10.1.3 io_wait

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

Definition at line 28 of file interrupts.c.

5.10.1.4 PIC1

```
#define PIC1 0x20
```

Definition at line 16 of file interrupts.c.

5.10.1.5 PIC2

```
#define PIC2 0xA0
```

Definition at line 17 of file interrupts.c.

5.10.2 Function Documentation

5.10.2.1 bounds()

```
void bounds ( )
```

5.10.2.2 breakpoint()

```
void breakpoint ( )
```

5.10.2.3 coprocessor()

```
void coprocessor ( )
```

5.10.2.4 coprocessor_segment()

```
void coprocessor_segment ( )
```

5.10.2.5 debug()

```
void debug ( )
```

5.10.2.6 device_not_available()

```
void device_not_available ( )
```

5.10.2.7 divide_error()

```
void divide_error ( )
```

5.10.2.8 do_bounds()

```
void do_bounds ( )
```

Definition at line 149 of file interrupts.c.

```
150 {  
151     kpanic("Bounds error");  
152 }
```

5.10.2.9 do_breakpoint()

```
void do_breakpoint ( )
```

Definition at line 141 of file interrupts.c.

```
142 {  
143     kpanic("Breakpoint");  
144 }
```

5.10.2.10 do_coprocessor()

```
void do_coprocessor ( )
```

Definition at line 193 of file interrupts.c.

```
194 {  
195     kpanic("Coprocessor error");  
196 }
```

5.10.2.11 do_coprocessor_segment()

```
void do_coprocessor_segment ( )
```

Definition at line 165 of file interrupts.c.

```
166 {  
167     kpanic("Coprocessor segment error");  
168 }
```

5.10.2.12 do_debug()

```
void do_debug ( )
```

Definition at line 133 of file interrupts.c.

```
134 {  
135     kpanic("Debug");  
136 }
```

5.10.2.13 do_device_not_available()

```
void do_device_not_available ( )
```

Definition at line 157 of file interrupts.c.

```
158 {  
159     kpanic("Device not available");  
160 }
```

5.10.2.14 do_divide_error()

```
void do_divide_error ( )
```

Definition at line 129 of file interrupts.c.

```
130 {  
131     kpanic("Division-by-zero");  
132 }
```

5.10.2.15 do_double_fault()

```
void do_double_fault ( )
```

Definition at line 161 of file interrupts.c.

```
162 {  
163     kpanic("Double fault");  
164 }
```

5.10.2.16 do_general_protection()

```
void do_general_protection ( )
```

Definition at line 181 of file interrupts.c.

```
182 {  
183     kpanic("General protection fault");  
184 }
```

5.10.2.17 do_invalid_op()

```
void do_invalid_op ( )
```

Definition at line 153 of file interrupts.c.

```
154 {  
155     kpanic("Invalid operation");  
156 }
```

5.10.2.18 do_invalid_tss()

```
void do_invalid_tss ( )
```

Definition at line 169 of file interrupts.c.

```
170 {  
171     kpanic("Invalid TSS");  
172 }
```


5.10.2.19 do_isr()

```
void do_isr ( )
```

Definition at line 53 of file interrupts.c.

```
54 {  
55     char in = inb(COM2);  
56     serial_print(&in);  
57     serial_println("here");  
58     outb(0x20,0x20); //EOI  
59 }
```

5.10.2.20 do_nmi()

```
void do_nmi ( )
```

Definition at line 137 of file interrupts.c.

```
138 {  
139     kpanic("NMI");  
140 }
```

5.10.2.21 do_overflow()

```
void do_overflow ( )
```

Definition at line 145 of file interrupts.c.

```
146 {  
147     kpanic("Overflow error");  
148 }
```

5.10.2.22 do_page_fault()

```
void do_page_fault ( )
```

Definition at line 185 of file interrupts.c.

```
186 {  
187     kpanic("Page Fault");  
188 }
```

5.10.2.23 do_reserved()

```
void do_reserved ( )
```

Definition at line 189 of file interrupts.c.

```
190 {  
191     serial_println("die: reserved");  
192 }
```

5.10.2.24 do_segment_not_present()

```
void do_segment_not_present ( )
```

Definition at line 173 of file interrupts.c.

```
174 {
175     kpanic("Segment not present");
176 }
```

5.10.2.25 do_stack_segment()

```
void do_stack_segment ( )
```

Definition at line 177 of file interrupts.c.

```
178 {
179     kpanic("Stack segment error");
180 }
```

5.10.2.26 double_fault()

```
void double_fault ( )
```

5.10.2.27 general_protection()

```
void general_protection ( )
```

5.10.2.28 init_irq()

```
void init_irq (
    void )
```

Definition at line 66 of file interrupts.c.

```
67 {
68     int i;
69
70     // Necessary interrupt handlers for protected mode
71     u32int isrs[17] = {
72         (u32int)divide_error,
73         (u32int)debug,
74         (u32int)nmi,
75         (u32int)breakpoint,
76         (u32int)overflow,
77         (u32int)bounds,
78         (u32int)invalid_op,
79         (u32int)device_not_available,
80         (u32int)double_fault,
81         (u32int)coprocessor_segment,
82         (u32int)invalid_tss,
83         (u32int)segment_not_present,
84         (u32int)stack_segment,
85         (u32int)general_protection,
86         (u32int)page_fault,
87         (u32int)reserved,
88         (u32int)coprocessor
89     };
90
91     // Install handlers; 0x08=sel, 0x8e=flags
92     for(i=0; i<32; i++){
93         if (i<17) idt_set_gate(i, isrs[i], 0x08, 0x8e);
94         else idt_set_gate(i, (u32int)reserved, 0x08, 0x8e);
95     }
96     // Ignore interrupts from the real time clock
97     idt_set_gate(0x08, (u32int)rtc_isr, 0x08, 0x8e);
98 }
```

5.10.2.29 init_pic()

```
void init_pic (
    void )
```

Definition at line 106 of file interrupts.c.

```
107 {
108     outb(PIC1, ICW1);    //send initialization code words 1 to PIC1
109     io_wait();
110     outb(PIC2, ICW1);    //send icw1 to PIC2
111     io_wait();
112     outb(PIC1+1, 0x20);  //icw2: remap irq0 to 32
113     io_wait();
114     outb(PIC2+1, 0x28);  //icw2: remap irq8 to 40
115     io_wait();
116     outb(PIC1+1, 4);     //icw3
117     io_wait();
118     outb(PIC2+1, 2);     //icw3
119     io_wait();
120     outb(PIC1+1, ICW4);  //icw4: 80x86, automatic handling
121     io_wait();
122     outb(PIC2+1, ICW4);  //icw4: 80x86, automatic handling
123     io_wait();
124     outb(PIC1+1, 0xFF);  //disable irqs for PIC1
125     io_wait();
126     outb(PIC2+1, 0xFF);  //disable irqs for PIC2
127 }
```

5.10.2.30 invalid_op()

```
void invalid_op ( )
```

5.10.2.31 invalid_tss()

```
void invalid_tss ( )
```

5.10.2.32 isr0()

```
void isr0 ( )
```

5.10.2.33 nmi()

```
void nmi ( )
```

5.10.2.34 overflow()

```
void overflow ( )
```

5.10.2.35 page_fault()

```
void page_fault ( )
```

5.10.2.36 reserved()

```
void reserved ( )
```

5.10.2.37 rtc_isr()

```
void rtc_isr ( )
```

5.10.2.38 segment_not_present()

```
void segment_not_present ( )
```

5.10.2.39 stack_segment()

```
void stack_segment ( )
```

5.10.3 Variable Documentation**5.10.3.1 idt_entries**

```
idt_entry idt_entries[256] [extern]
```

Definition at line 17 of file tables.c.

5.11 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 <mem/heap.h>
#include <mem/paging.h>
#include "modules/mpx_supt.h"
#include "modules/R1/commhand.h"
```

Functions

- void [kmain](#) (void)

5.11.1 Function Documentation

5.11.1.1 kmain()

```
void kmain (
    void )
```

Definition at line 28 of file kmain.c.

```
29 {
30     // extern uint32_t magic;
31     // Uncomment if you want to access the multiboot header
32     // extern void *mbd;
33     // char *boot_loader_name = (char*)((long*)mbd)[16];
34
35
36     // 0) Initialize Serial I/O
37     // functions to initialize serial I/O can be found in serial.c
38     // there are 3 functions to call
39
40     init_serial(COM1);
41     set_serial_in(COM1);
42     set_serial_out(COM1);
43
44     klogv("Starting MPX boot sequence...");
45     klogv("Initialized serial I/O on COM1 device...");
46
47     // 1) Initialize the support software by identifying the current
48     //     MPX Module. This will change with each module.
49     // you will need to call mpx_init from the mpx_supt.c
50
51     mpx_init(MODULE_R1);
52
53     // 2) Check that the boot was successful and correct when using grub
54     // Comment this when booting the kernel directly using QEMU, etc.
55     //if ( magic != 0x2BADB002 ){
56     //    kpanic("Boot was not error free. Halting.");
57     //}
58
59     // 3) Descriptor Tables -- tables.c
60     // you will need to initialize the global
61     // this keeps track of allocated segments and pages
62     klogv("Initializing descriptor tables...");
63 }
```

```

64  init_gdt();
65  init_idt();
66
67  init_pic();
68  sti();
69
70  // 4) Interrupt vector table -- tables.c
71  // this creates and initializes a default interrupt vector table
72  // this function is in tables.c
73
74  init_irq();
75
76  klogv("Interrupt vector table initialized!");
77
78  // 5) Virtual Memory -- paging.c -- init_paging
79  // this function creates the kernel's heap
80  // from which memory will be allocated when the program calls
81  // sys_alloc_mem UNTIL the memory management module is completed
82  // this allocates memory using discrete "pages" of physical memory
83  // NOTE: You will only have about 70000 bytes of dynamic memory
84  //
85  klogv("Initializing virtual memory...");
86
87  init_paging();
88
89  // 6) Call YOUR command handler - interface method
90  klogv("Transferring control to commhand...");
91  commhand();
92
93  // 7) System Shutdown on return from your command handler
94
95  klogv("Starting system shutdown procedure...");
96
97  /* Shutdown Procedure */
98  klogv("Shutdown complete. You may now turn off the machine. (QEMU: C-a x)");
99  hlt();
100 }

```

5.12 kernel/core/serial.c File Reference

```

#include <stdint.h>
#include <string.h>
#include <core/io.h>
#include <core/serial.h>

```

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)
- int * `polling` (char *buffer, int *count)

Variables

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

5.12.1 Macro Definition Documentation

5.12.1.1 NO_ERROR

```
#define NO_ERROR 0
```

Definition at line 13 of file serial.c.

5.12.2 Function Documentation

5.12.2.1 init_serial()

```
int init_serial (
    int device )
```

Definition at line 23 of file serial.c.

```
24 {
25     outb(device + 1, 0x00);           //disable interrupts
26     outb(device + 3, 0x80);           //set line control register
27     outb(device + 0, 115200 / 9600); //set bsd least sig bit
28     outb(device + 1, 0x00);           //brd most significant bit
29     outb(device + 3, 0x03);           //lock divisor; 8bits, no parity, one stop
30     outb(device + 2, 0xC7);           //enable fifo, clear, 14byte threshold
31     outb(device + 4, 0x0B);           //enable interrupts, rts/dsr set
32     (void)inb(device);               //read bit to reset port
33     return NO_ERROR;
34 }
```

5.12.2.2 polling()

```
int* polling (
    char * buffer,
    int * count )
```

Definition at line 93 of file serial.c.

```
94 {
95     // insert your code to gather keyboard input via the technique of polling.
96
97     char keyboard_character;
98
99     int cursor = 0;
100
101     char log[] = {'\0', '\0', '\0', '\0'};
102
103     int characters_in_buffer = 0;
104
105     while (1)
106     {
107
108         if (inb(COM1 + 5) & 1)
109         {
110             // is there input char?
111             keyboard_character = inb(COM1); //read the char from COM1
112
113             if (keyboard_character == '\n' || keyboard_character == '\r')
114             { // HANDLEING THE CARRIAGE RETURN AND NEW LINE CHARACTERS
```

```

114
115     buffer[characters_in_buffer] = '\0';
116     break;
117 }
118 else if ((keyboard_character == 127 || keyboard_character == 8) && cursor > 0)
119 { // HANDELING THE BACKSPACE CHARACTER
120
121     //serial_println("Handleing backspace character.");
122     serial_print("\033[K");
123
124     buffer[cursor - 1] = '\0';
125     serial_print("\b \b");
126     serial_print(buffer + cursor);
127     cursor--;
128
129     int temp_cursor = cursor;
130
131     while (buffer[temp_cursor + 1] != '\0')
132     {
133         buffer[temp_cursor] = buffer[temp_cursor + 1];
134         buffer[temp_cursor + 1] = '\0';
135         temp_cursor++;
136     }
137
138     characters_in_buffer--;
139     cursor = characters_in_buffer;
140 }
141 else if (keyboard_character == '~' && cursor < 99)
142 { //HANDLING THE DELETE KEY
143     // \033[3~
144
145     serial_print("\033[K");
146
147     buffer[cursor + 1] = '\0';
148     serial_print("\b \b");
149     serial_print(buffer + cursor);
150
151     int temp_cursor = cursor + 1;
152
153     while (buffer[temp_cursor + 1] != '\0')
154     {
155         buffer[temp_cursor] = buffer[temp_cursor + 1];
156         buffer[temp_cursor + 1] = '\0';
157         temp_cursor++;
158     }
159
160     characters_in_buffer--;
161     cursor = characters_in_buffer;
162 }
163 else if (keyboard_character == '\033')
164 { // HANDELING FIRST CHARACTER FOR ARROW KEYS
165
166     log[0] = keyboard_character;
167 }
168 else if (keyboard_character == '[' && log[0] == '\033')
169 { // HANDELING SECOND CHARACTER FOR ARROW KEYS
170
171     log[1] = keyboard_character;
172 }
173 else if (log[0] == '\033' && log[1] == '[')
174 { // HANDELING LAST CHARACTER FOR ARROW KEYS
175     log[2] = keyboard_character;
176
177     if (keyboard_character == 'A')
178     { //Up arrow
179         //Call a history function from the commhand or do nothing
180     }
181     else if (keyboard_character == 'B')
182     { //Down arrow
183         //Call a history command from the commhand or do nothing
184     }
185     else if (keyboard_character == 'C' && cursor != 99)
186     { //Right arrow
187
188         serial_print("\033[C");
189         cursor++;
190     }
191     else if (keyboard_character == 'D' && cursor != 0)
192     { //Left arrow
193
194         serial_print("\033[D");
195         cursor--;
196     }
197
198     memset(log, '\0', 4);
199 }
200 else

```



```

201     {
202
203         if (cursor == 0 && buffer[cursor] == '\0') //Adding character at beginning of buffer
204         {
205             buffer[cursor] = keyboard_character;
206             serial_print(&keyboard_character);
207             cursor++;
208         }
209         else if (buffer[cursor] == '\0') //Adding character at the end of the buffer
210         {
211             buffer[cursor] = keyboard_character;
212             serial_print(&keyboard_character);
213             cursor++;
214         }
215         else //Inserting character to the middle of the buffer
216         {
217             char temp_buffer[strlen(buffer)];
218             memset(temp_buffer, '\0', strlen(buffer));
219
220             int temp_cursor = 0;
221             while (temp_cursor <= characters_in_buffer) //Filling the temp_buffer with all of the
characters from buffer, and inserting the new character.
222             {
223                 if (temp_cursor < cursor)
224                 {
225                     temp_buffer[temp_cursor] = buffer[temp_cursor];
226                 }
227                 else if (temp_cursor > cursor)
228                 {
229                     temp_buffer[temp_cursor] = buffer[temp_cursor - 1];
230                 }
231                 else
232                 { //temp_cursor == cursor
233                     temp_buffer[temp_cursor] = keyboard_character;
234                 }
235                 temp_cursor++;
236             }
237
238             temp_cursor = 0;
239             int temp_buffer_size = strlen(temp_buffer);
240             while (temp_cursor <= temp_buffer_size) //Setting the contents of the buffer equal to the
temp_buffer.
241             {
242                 buffer[temp_cursor] = temp_buffer[temp_cursor];
243                 temp_cursor++;
244             }
245
246             serial_print("\033[K");
247             serial_print(&keyboard_character);
248             serial_print(buffer + cursor + 1);
249             cursor++;
250         }
251         characters_in_buffer++;
252     }
253 }
254 }
255
256 *count = characters_in_buffer; // buffer count
257
258 return count;
259 }

```

5.12.2.3 serial_print()

```

int serial_print (
    const char * msg )

```

Definition at line 57 of file serial.c.

```

58 {
59     int i;
60     for (i = 0; *(i + msg) != '\0'; i++)
61     {
62         outb(serial_port_out, *(i + msg));
63     }
64     if (*msg == '\r')
65         outb(serial_port_out, '\n');
66     return NO_ERROR;
67 }

```

5.12.2.4 serial_println()

```
int serial_println (
    const char * msg )
```

Definition at line 41 of file serial.c.

```
42 {
43     int i;
44     for (i = 0; *(i + msg) != '\0'; i++)
45     {
46         outb(serial_port_out, *(i + msg));
47     }
48     outb(serial_port_out, '\r');
49     outb(serial_port_out, '\n');
50     return NO_ERROR;
51 }
```

5.12.2.5 set_serial_in()

```
int set_serial_in (
    int device )
```

Definition at line 87 of file serial.c.

```
88 {
89     serial_port_in = device;
90     return NO_ERROR;
91 }
```

5.12.2.6 set_serial_out()

```
int set_serial_out (
    int device )
```

Definition at line 75 of file serial.c.

```
76 {
77     serial_port_out = device;
78     return NO_ERROR;
79 }
```

5.12.3 Variable Documentation

5.12.3.1 serial_port_in

```
int serial_port_in = 0
```

Definition at line 17 of file serial.c.

5.12.3.2 serial_port_out

```
int serial_port_out = 0
```

Definition at line 16 of file serial.c.

5.13 kernel/core/system.c File Reference

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

Functions

- void [klogv](#) (const char *msg)
- void [kpanic](#) (const char *msg)

5.13.1 Function Documentation

5.13.1.1 klogv()

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

Definition at line 11 of file system.c.

```
12 {
13     char logmsg[64] = {'\0'}, prefix[] = "klogv: ";
14     strcat(logmsg, prefix);
15     strcat(logmsg, msg);
16     serial_println(logmsg);
17 }
```

5.13.1.2 kpanic()

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

Definition at line 24 of file system.c.

```
25 {
26     cli(); //disable interrupts
27     char logmsg[64] = {'\0'}, prefix[] = "Panic: ";
28     strcat(logmsg, prefix);
29     strcat(logmsg, msg);
30     klogv(logmsg);
31     hlt(); //halt
32 }
```

5.14 kernel/core/tables.c File Reference

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

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]

5.14.1 Function Documentation

5.14.1.1 gdt_init_entry()

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

Definition at line 57 of file tables.c.

```
59 {
60     gdt_entry *new_entry = &gdt_entries[idx];
61     new_entry->base_low = (base & 0xFFFF);
62     new_entry->base_mid = (base >> 16) & 0xFF;
63     new_entry->base_high = (base >> 24) & 0xFF;
64     new_entry->limit_low = (limit & 0xFFFF);
65     new_entry->flags = (limit >> 16) & 0xFF;
66     new_entry->flags |= flags & 0xF0;
67     new_entry->access = access;
68 }
```

5.14.1.2 idt_set_gate()

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

Definition at line 27 of file tables.c.

```
29 {
30     idt_entry *new_entry = &idt_entries[idx];
31     new_entry->base_low = (base & 0xFFFF);
32     new_entry->base_high = (base >> 16) & 0xFFFF;
33     new_entry->sselect = sel;
34     new_entry->zero = 0;
35     new_entry->flags = flags;
36 }
```

5.14.1.3 init_gdt()

```
void init_gdt ( )
```

Definition at line 75 of file tables.c.

```
76 {
77     gdt_ptr.limit = 5 * sizeof(gdt_entry) - 1;
78     gdt_ptr.base = (u32int) gdt_entries;
79
80     u32int limit = 0xFFFFFFFF;
81     gdt_init_entry(0, 0, 0, 0, 0); //required null segment
82     gdt_init_entry(1, 0, limit, 0x9A, 0xCF); //code segment
83     gdt_init_entry(2, 0, limit, 0x92, 0xCF); //data segment
84     gdt_init_entry(3, 0, limit, 0xFA, 0xCF); //user mode code segment
85     gdt_init_entry(4, 0, limit, 0xF2, 0xCF); //user mode data segment
86
87     write_gdt_ptr((u32int) &gdt_ptr, sizeof(gdt_ptr));
88 }
```

5.14.1.4 init_idt()

```
void init_idt ( )
```

Definition at line 43 of file tables.c.

```
44 {
45     idt_ptr.limit = 256*sizeof(idt_descriptor) - 1;
46     idt_ptr.base = (u32int)idt_entries;
47     memset(idt_entries, 0, 256*sizeof(idt_descriptor));
48
49     write_idt_ptr((u32int)&idt_ptr);
50 }
```

5.14.1.5 write_gdt_ptr()

```
void write_gdt_ptr (
    u32int ,
    size_t )
```

5.14.1.6 write_idt_ptr()

```
void write_idt_ptr (
    u32int )
```

5.14.2 Variable Documentation

5.14.2.1 gdt_entries

```
gdt_entry gdt_entries[5]
```

Definition at line 13 of file tables.c.

5.14.2.2 gdt_ptr

```
gdt_descriptor gdt_ptr
```

Definition at line 12 of file tables.c.

5.14.2.3 idt_entries

```
idt_entry idt_entries[256]
```

Definition at line 17 of file tables.c.

5.14.2.4 idt_ptr

```
idt_descriptor idt_ptr
```

Definition at line 16 of file tables.c.

5.15 kernel/mem/heap.c File Reference

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

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`

5.15.1 Function Documentation

5.15.1.1 _kmalloc()

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

Definition at line 24 of file heap.c.

```
25 {
26     u32int *addr;
27
28     // Allocate on the kernel heap if one has been created
29     if (kheap != 0){
30         addr = (u32int*)alloc(size, kheap, page_align);
31         if (phys_addr){
32             page_entry *page = get_page((u32int)addr, kdir, 0);
33             *phys_addr = (page->frameaddr*0x1000) + ((u32int)addr & 0xFFF);
34         }
35         return (u32int)addr;
36     }
37     // Else, allocate directly from physical memory
38     else {
39         if (page_align && (phys_alloc_addr & 0xFFFFF000)){
40             phys_alloc_addr &= 0xFFFFF000;
41             phys_alloc_addr += 0x1000;
42         }
43         addr = (u32int*)phys_alloc_addr;
44         if (phys_addr){
45             *phys_addr = phys_alloc_addr;
46         }
47         phys_alloc_addr += size;
48         return (u32int)addr;
49     }
50 }
```

5.15.1.2 alloc()

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

Definition at line 57 of file heap.c.

```
58 {
59     no_warn(size||align||h);
60     static u32int heap_addr = KHEAP_BASE;
61
62     u32int base = heap_addr;
63     heap_addr += size;
64
65     if (heap_addr > KHEAP_BASE + KHEAP_MIN)
66         serial_println("Heap is full!");
67
68     return base;
69 }
```

5.15.1.3 kmalloc()

```
u32int kmalloc (
    u32int size )
```

Definition at line 52 of file heap.c.

```
53 {
54     return _kmalloc(size,0,0);
55 }
```

5.15.1.4 make_heap()

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

Definition at line 71 of file heap.c.

```
72 {
73     no_warn(base||max||min);
74     return (heap*) kmalloc(sizeof(heap));
75 }
```

5.15.2 Variable Documentation

5.15.2.1 __end

```
void __end
```

Definition at line 18 of file heap.c.

5.15.2.2 `_end`

```
void _end
```

Definition at line 18 of file heap.c.

5.15.2.3 `curr_heap`

```
heap* curr_heap = 0
```

Definition at line 15 of file heap.c.

5.15.2.4 `end`

```
void* end [extern]
```

5.15.2.5 `kdir`

```
page_dir* kdir [extern]
```

Definition at line 21 of file paging.c.

5.15.2.6 `kheap`

```
heap* kheap = 0
```

Definition at line 14 of file heap.c.

5.15.2.7 `phys_alloc_addr`

```
u32int phys_alloc_addr = (u32int)&end
```

Definition at line 22 of file heap.c.

5.16 kernel/mem/paging.c File Reference

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

Functions

- void [set_bit](#) (u32int addr)
- void [clear_bit](#) (u32int addr)
- u32int [get_bit](#) (u32int addr)
- u32int [find_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](#)

5.16.1 Function Documentation

5.16.1.1 [clear_bit\(\)](#)

```
void clear_bit (
    u32int addr )
```

Definition at line 44 of file [paging.c](#).

```
45 {
46     u32int frame = addr/page\_size;
47     u32int index = frame/32;
48     u32int offset = frame%32;
49     frames[index] &= ~(1 « offset);
50 }
```

5.16.1.2 find_free()

```
u32int find_free ( )
```

Definition at line 68 of file paging.c.

```
69 {
70     u32int i, j;
71     for (i=0; i<nframes/32; i++)
72         if (frames[i] != 0xFFFFFFFF) //if frame not full
73             for (j=0; j<32; j++) //find first free bit
74                 if (!(frames[i] & (1 << j)))
75                     return i*32+j;
76
77     return -1; //no free frames
78 }
```

5.16.1.3 get_bit()

```
u32int get_bit (
    u32int addr )
```

Definition at line 56 of file paging.c.

```
57 {
58     u32int frame = addr/page_size;
59     u32int index = frame/32;
60     u32int offset = frame%32;
61     return (frames[index] & (1 << offset));
62 }
```

5.16.1.4 get_page()

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

Definition at line 85 of file paging.c.

```
86 {
87     u32int phys_addr;
88     u32int index = addr / page_size / 1024;
89     u32int offset = addr / page_size % 1024;
90
91     //return it if it exists
92     if (dir->tables[index])
93         return &dir->tables[index]->pages[offset];
94
95     //create it
96     else if (make_table){
97         dir->tables[index] = (page_table*)_kmalloc(sizeof(page_table), 1, &phys_addr);
98         dir->tables_phys[index] = phys_addr | 0x7; //enable present, writable
99         return &dir->tables[index]->pages[offset];
100     }
101     else return 0;
102 }
```

5.16.1.5 init_paging()

```
void init_paging ( )
```

Definition at line 111 of file paging.c.

```
112 {
113     //create frame bitmap
114     nframes = (u32int) (mem_size/page_size);
115     frames = (u32int*) kmalloc(nframes/32);
116     memset(frames, 0, nframes/32);
117
118     //create kernel directory
119     kdir = (page_dir*) _kmalloc(sizeof(page_dir), 1, 0); //page aligned
120     memset(kdir, 0, sizeof(page_dir));
121
122     //get pages for kernel heap
123     u32int i = 0x0;
124     for(i=KHEAP_BASE; i<(KHEAP_BASE+KHEAP_MIN); i+=1){
125         get_page(i,kdir,1);
126     }
127
128     //perform identity mapping of used memory
129     //note: placement_addr gets incremented in get_page,
130     //so we're mapping the first frames as well
131     i = 0x0;
132     while (i < (phys_alloc_addr+0x10000)){
133         new_frame(get_page(i,kdir,1));
134         i += page_size;
135     }
136
137     //allocate heap frames now that the placement addr has increased.
138     //placement addr increases here for heap
139     for(i=KHEAP_BASE; i<(KHEAP_BASE+KHEAP_MIN); i+=PAGE_SIZE){
140         new_frame(get_page(i,kdir,1));
141     }
142
143     //load the kernel page directory; enable paging
144     load_page_dir(kdir);
145
146     //setup the kernel heap
147     kheap = make_heap(KHEAP_BASE, KHEAP_SIZE, KHEAP_BASE+KHEAP_MIN);
148 }
```

5.16.1.6 load_page_dir()

```
void load_page_dir (
    page_dir * new_dir )
```

Definition at line 158 of file paging.c.

```
159 {
160     cdir = new_dir;
161     asm volatile ("mov %0,%{cr3}:: "b"(&cdir->tables_phys[0]);");
162     u32int cr0;
163     asm volatile ("mov %%cr0,%0": "=b"(cr0));
164     cr0 |= 0x80000000;
165     asm volatile ("mov %0,%{cr0}:: "b"(cr0));
166 }
```

5.16.1.7 new_frame()

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

Definition at line 173 of file paging.c.

```
174 {
175     u32int index;
```

```
176  if (page->frameaddr != 0) return;
177  if ( (u32int)(-1) == (index=find_free()) ) kpanic("Out of memory");
178
179  //mark a frame as in-use
180  set_bit(index*page_size);
181  page->present = 1;
182  page->frameaddr = index;
183  page->writeable = 1;
184  page->usermode = 0;
185 }
```

5.16.1.8 set_bit()

```
void set_bit (
    u32int addr )
```

Definition at line 32 of file paging.c.

```
33 {
34  u32int frame = addr/page_size;
35  u32int index = frame/32;
36  u32int offset = frame%32;
37  frames[index] |= (1 « offset);
38 }
```

5.16.2 Variable Documentation

5.16.2.1 cdir

```
page_dir* cdir = 0
```

Definition at line 22 of file paging.c.

5.16.2.2 frames

```
u32int* frames
```

Definition at line 19 of file paging.c.

5.16.2.3 kdir

```
page_dir* kdir = 0
```

Definition at line 21 of file paging.c.

5.16.2.4 kheap

```
heap* kheap [extern]
```

Definition at line 14 of file heap.c.

5.16.2.5 mem_size

```
u32int mem_size = 0x4000000
```

Definition at line 15 of file paging.c.

5.16.2.6 nframes

```
u32int nframes
```

Definition at line 18 of file paging.c.

5.16.2.7 page_size

```
u32int page_size = 0x1000
```

Definition at line 16 of file paging.c.

5.16.2.8 phys_alloc_addr

```
u32int phys_alloc_addr [extern]
```

Definition at line 22 of file heap.c.

5.17 lib/string.c File Reference

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

Functions

- int [strlen](#) (const char *s)
- char * [strcpy](#) (char *s1, const char *s2)
- int [atoi](#) (const char *s)
- int [strcmp](#) (const char *s1, const char *s2)
- char * [strcat](#) (char *s1, const char *s2)
- int [isspace](#) (const char *c)
- void * [memset](#) (void *s, int c, [size_t](#) n)
- char * [strtok](#) (char *s1, const char *s2)

5.17.1 Function Documentation

5.17.1.1 atoi()

```
int atoi (
    const char * s )
```

Definition at line 48 of file string.c.

```
49 {
50     int res=0;
51     int charVal=0;
52     char sign = ' ';
53     char c = *s;
54
55
56     while(isspace(&c)){ ++s; c = *s;} // advance past whitespace
57
58
59     if (*s == '-' || *s == '+') sign = *(s++); // save the sign
60
61
62     while(*s != '\0'){
63         charVal = *s - 48;
64         res = res * 10 + charVal;
65         s++;
66
67     }
68
69
70     if ( sign == '-') res=res * -1;
71
72     return res; // return integer
73 }
```

5.17.1.2 isspace()

```
int isspace (
    const char * c )
```

Definition at line 119 of file string.c.

```
120 {
121     if (*c == ' ' ||
122         *c == '\n' ||
123         *c == '\r' ||
124         *c == '\f' ||
125         *c == '\t' ||
126         *c == '\v'){
127         return 1;
128     }
129     return 0;
130 }
```

5.17.1.3 memset()

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

Definition at line 137 of file string.c.

```
138 {
139     unsigned char *p = (unsigned char *) s;
140     while(n--){
141         *p++ = (unsigned char) c;
142     }
143     return s;
144 }
```

5.17.1.4 strcat()

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

Definition at line 106 of file string.c.

```
107 {
108     char *rc = s1;
109     if (*s1) while(*++s1);
110     while( (*s1++ = *s2++) );
111     return rc;
112 }
```

5.17.1.5 strcmp()

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

Definition at line 79 of file string.c.

```
80 {
81
82     // Remarks:
83     // 1) If we made it to the end of both strings (i. e. our pointer points to a
84     // '\0' character), the function will return 0
85     // 2) If we didn't make it to the end of both strings, the function will
86     // return the difference of the characters at the first index of
87     // indifference.
88     while ( (*s1) && (*s1==*s2) ){
89         ++s1;
90         ++s2;
91     }
92     return ( *(unsigned char *)s1 - *(unsigned char *)s2 );
93 }
```


5.17.1.6 strcpy()

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

Definition at line 36 of file string.c.

```
37 {
38     char *rc = s1;
39     while( (*s1++ = *s2++) );
40     return rc; // return pointer to destination string
41 }
```

5.17.1.7 strlen()

```
int strlen (
    const char * s )
```

Definition at line 24 of file string.c.

```
25 {
26     int r1 = 0;
27     if (*s) while(*s++) r1++;
28     return r1; //return length of string
29 }
```

5.17.1.8 strtok()

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

Definition at line 151 of file string.c.

```
152 {
153     static char *tok_tmp = NULL;
154     const char *p = s2;
155
156     //new string
157     if (s1!=NULL){
158         tok_tmp = s1;
159     }
160     //old string cont'd
161     else {
162         if (tok_tmp==NULL){
163             return NULL;
164         }
165         s1 = tok_tmp;
166     }
167
168     //skip leading s2 characters
169     while ( *p && *s1 ){
170         if (*s1==*p){
171             ++s1;
172             p = s2;
173             continue;
174         }
175         ++p;
176     }
177
178     //no more to parse
179     if (!*s1){
180         return (tok_tmp = NULL);
181     }
182
183     //skip non-s2 characters
184     tok_tmp = s1;
```

```

185 while (*tok_tmp){
186     p = s2;
187     while (*p){
188         if (*tok_tmp==*p++){
189             *tok_tmp++ = '\0';
190             return s1;
191         }
192     }
193     ++tok_tmp;
194 }
195
196 //end of string
197 tok_tmp = NULL;
198 return s1;
199 }

```

5.18 modules/mpx_supt.c File Reference

```

#include "mpx_supt.h"
#include <mem/heap.h>
#include <string.h>
#include <core/serial.h>

```

Functions

- int [sys_req](#) (int op_code, int device_id, char *buffer_ptr, int *count_ptr)
- void [mpx_init](#) (int cur_mod)
- void [sys_set_malloc](#) (u32int(*func)(u32int))
- void [sys_set_free](#) (int(*func)(void *))
- void * [sys_alloc_mem](#) (u32int size)
- int [sys_free_mem](#) (void *ptr)
- void [idle](#) ()

Variables

- [param](#) params
- int [current_module](#) = -1
- u32int(* [student_malloc](#))(u32int)
- int(* [student_free](#))(void *)

5.18.1 Function Documentation

5.18.1.1 idle()

```
void idle ( )
```

Definition at line 173 of file mpx_supt.c.

```

174 {
175     char msg[30];
176     int count=0;
177
178     memset( msg, '\0', sizeof(msg));
179     strcpy(msg, "IDLE PROCESS EXECUTING.\n");
180     count = strlen(msg);
181
182     while(1){
183         sys_req( WRITE, DEFAULT_DEVICE, msg, &count);
184         sys_req(IDLE, DEFAULT_DEVICE, NULL, NULL);
185     }
186 }

```

5.18.1.2 mpx_init()

```
void mpx_init (
    int cur_mod )
```

Definition at line 106 of file mpx_supt.c.

```
107 {
108
109     current_module = cur_mod;
110     if (cur_mod == MEM_MODULE)
111         mem_module_active = TRUE;
112
113     if (cur_mod == IO_MODULE)
114         io_module_active = TRUE;
115 }
```

5.18.1.3 sys_alloc_mem()

```
void* sys_alloc_mem (
    u32int size )
```

Definition at line 144 of file mpx_supt.c.

```
145 {
146     if (!mem_module_active)
147         return (void *) kcalloc(size);
148     else
149         return (void *) (*student_malloc)(size);
150 }
```

5.18.1.4 sys_free_mem()

```
int sys_free_mem (
    void * ptr )
```

Definition at line 158 of file mpx_supt.c.

```
159 {
160     if (mem_module_active)
161         return (*student_free)(ptr);
162     // otherwise we don't free anything
163     return -1;
164 }
```

5.18.1.5 sys_req()

```
int sys_req (
    int op_code,
    int device_id,
    char * buffer_ptr,
    int * count_ptr )
```

Definition at line 49 of file mpx_supt.c.

```
54 {
55     int return_code =0;
56
57     if (op_code == IDLE || op_code == EXIT){
```

```

58     // store the process's operation request
59     // trigger interrupt 60h to invoke
60     params.op_code = op_code;
61     asm volatile ("int $60");
62 } // idle or exit
63
64 else if (op_code == READ || op_code == WRITE) {
65     // validate buffer pointer and count pointer
66     if (buffer_ptr == NULL)
67         return_code = INVALID_BUFFER;
68     else if (count_ptr == NULL || *count_ptr <= 0)
69         return_code = INVALID_COUNT;
70
71     // if parameters are valid store in the params structure
72     if ( return_code == 0){
73         params.op_code = op_code;
74         params.device_id = device_id;
75         params.buffer_ptr = buffer_ptr;
76         params.count_ptr = count_ptr;
77
78         if (!io_module_active){
79             // if default device
80             if (op_code == READ)
81                 return_code = *(polling(buffer_ptr, count_ptr));
82
83             else //must be WRITE
84                 return_code = serial_print(buffer_ptr);
85
86         } else { // I/O module is implemented
87             asm volatile ("int $60");
88         } // NOT IO_MODULE
89     }
90     else return_code = INVALID_OPERATION;
91
92     return return_code;
93 } // end of sys_req

```

5.18.1.6 sys_set_free()

```

void sys_set_free (
    int(*) (void *) func )

```

Definition at line 134 of file mpx_supt.c.

```

135 {
136     student_free = func;
137 }

```

5.18.1.7 sys_set_malloc()

```

void sys_set_malloc (
    u32int(*) (u32int) func )

```

Definition at line 124 of file mpx_supt.c.

```

125 {
126     student_malloc = func;
127 }

```

5.18.2 Variable Documentation

5.18.2.1 `current_module`

```
int current_module = -1
```

Definition at line 18 of file mpx_supt.c.

5.18.2.2 `params`

```
param params
```

Definition at line 15 of file mpx_supt.c.

5.18.2.3 `student_free`

```
int (* student_free) (void *)
```

Definition at line 28 of file mpx_supt.c.

5.18.2.4 `student_malloc`

```
u32int (* student_malloc) (u32int)
```

Definition at line 24 of file mpx_supt.c.

5.19 modules/mpx_supt.h File Reference

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

Classes

- struct `param`

Macros

- `#define EXIT 0`
- `#define IDLE 1`
- `#define READ 2`
- `#define WRITE 3`
- `#define INVALID_OPERATION 4`
- `#define TRUE 1`
- `#define FALSE 0`
- `#define MODULE_R1 0`
- `#define MODULE_R2 1`
- `#define MODULE_R3 2`
- `#define MODULE_R4 4`
- `#define MODULE_R5 8`
- `#define MODULE_F 9`
- `#define IO_MODULE 10`
- `#define MEM_MODULE 11`
- `#define INVALID_BUFFER 1000`
- `#define INVALID_COUNT 2000`
- `#define DEFAULT_DEVICE 111`
- `#define COM_PORT 222`

Functions

- `int sys_req (int op_code, int device_id, char *buffer_ptr, int *count_ptr)`
- `void mpx_init (int cur_mod)`
- `void sys_set_malloc (u32int)(*func)(u32int))`
- `void sys_set_free (int)(*func)(void *)`
- `void * sys_alloc_mem (u32int size)`
- `int sys_free_mem (void *ptr)`
- `void idle ()`

5.19.1 Macro Definition Documentation

5.19.1.1 COM_PORT

```
#define COM_PORT 222
```

Definition at line 29 of file mpx_supt.h.

5.19.1.2 DEFAULT_DEVICE

```
#define DEFAULT_DEVICE 111
```

Definition at line 28 of file mpx_supt.h.

5.19.1.3 EXIT

```
#define EXIT 0
```

Definition at line 6 of file mpx_supt.h.

5.19.1.4 FALSE

```
#define FALSE 0
```

Definition at line 13 of file mpx_supt.h.

5.19.1.5 IDLE

```
#define IDLE 1
```

Definition at line 7 of file mpx_supt.h.

5.19.1.6 INVALID_BUFFER

```
#define INVALID_BUFFER 1000
```

Definition at line 25 of file mpx_supt.h.

5.19.1.7 INVALID_COUNT

```
#define INVALID_COUNT 2000
```

Definition at line 26 of file mpx_supt.h.

5.19.1.8 INVALID_OPERATION

```
#define INVALID_OPERATION 4
```

Definition at line 10 of file mpx_supt.h.

5.19.1.9 IO_MODULE

```
#define IO_MODULE 10
```

Definition at line 21 of file mpx_supt.h.

5.19.1.10 MEM_MODULE

```
#define MEM_MODULE 11
```

Definition at line 22 of file mpx_supt.h.

5.19.1.11 MODULE_F

```
#define MODULE_F 9
```

Definition at line 20 of file mpx_supt.h.

5.19.1.12 MODULE_R1

```
#define MODULE_R1 0
```

Definition at line 15 of file mpx_supt.h.

5.19.1.13 MODULE_R2

```
#define MODULE_R2 1
```

Definition at line 16 of file mpx_supt.h.

5.19.1.14 MODULE_R3

```
#define MODULE_R3 2
```

Definition at line 17 of file mpx_supt.h.

5.19.1.15 MODULE_R4

```
#define MODULE_R4 4
```

Definition at line 18 of file mpx_supt.h.

5.19.1.16 MODULE_R5

```
#define MODULE_R5 8
```

Definition at line 19 of file mpx_supt.h.

5.19.1.17 READ

```
#define READ 2
```

Definition at line 8 of file mpx_supt.h.

5.19.1.18 TRUE

```
#define TRUE 1
```

Definition at line 12 of file mpx_supt.h.

5.19.1.19 WRITE

```
#define WRITE 3
```

Definition at line 9 of file mpx_supt.h.

5.19.2 Function Documentation

5.19.2.1 idle()

```
void idle ( )
```

Definition at line 173 of file mpx_supt.c.

```
174 {
175     char msg[30];
176     int count=0;
177
178     memset( msg, '\0', sizeof(msg));
179     strcpy(msg, "IDLE PROCESS EXECUTING.\n");
180     count = strlen(msg);
181
182     while(1){
183         sys_req( WRITE, DEFAULT_DEVICE, msg, &count);
184         sys_req(IDLE, DEFAULT_DEVICE, NULL, NULL);
185     }
186 }
```

5.19.2.2 mpx_init()

```
void mpx_init (
    int cur_mod )
```

Definition at line 106 of file mpx_supt.c.

```
107 {
108
109     current_module = cur_mod;
110     if (cur_mod == MEM_MODULE)
111         mem_module_active = TRUE;
112
113     if (cur_mod == IO_MODULE)
114         io_module_active = TRUE;
115 }
```

5.19.2.3 sys_alloc_mem()

```
void* sys_alloc_mem (
    u32int size )
```

Definition at line 144 of file mpx_supt.c.

```
145 {
146     if (!mem_module_active)
147         return (void *) kmalloc(size);
148     else
149         return (void *) (*student_malloc)(size);
150 }
```

5.19.2.4 sys_free_mem()

```
int sys_free_mem (
    void * ptr )
```

Definition at line 158 of file mpx_supt.c.

```
159 {
160     if (mem_module_active)
161         return (*student_free)(ptr);
162     // otherwise we don't free anything
163     return -1;
164 }
```

5.19.2.5 sys_req()

```
int sys_req (
    int op_code,
    int device_id,
    char * buffer_ptr,
    int * count_ptr )
```

Definition at line 49 of file mpx_supt.c.

```
54 {
55     int return_code =0;
56
57     if (op_code == IDLE || op_code == EXIT){
58         // store the process's operation request
59         // trigger interrupt 60h to invoke
60         params.op_code = op_code;
61         asm volatile ("int $60");
62     } // idle or exit
63
64     else if (op_code == READ || op_code == WRITE) {
65         // validate buffer pointer and count pointer
66         if (buffer_ptr == NULL)
67             return_code = INVALID_BUFFER;
68         else if (count_ptr == NULL || *count_ptr <= 0)
69             return_code = INVALID_COUNT;
70
71         // if parameters are valid store in the params structure
72         if ( return_code == 0){
73             params.op_code = op_code;
74             params.device_id = device_id;
75             params.buffer_ptr = buffer_ptr;
76             params.count_ptr = count_ptr;
77
78             if (!io_module_active){
79                 // if default device
80                 if (op_code == READ)
81                     return_code = *(polling(buffer_ptr, count_ptr));
82
83                 else //must be WRITE
84                     return_code = serial_print(buffer_ptr);
85
86             } else { // I/O module is implemented
87                 asm volatile ("int $60");
88             } // NOT IO_MODULE
89         }
90     } else return_code = INVALID_OPERATION;
91
92     return return_code;
93 } // end of sys_req
```

5.19.2.6 sys_set_free()

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

Definition at line 134 of file mpx_supt.c.

```
135 {
136     student_free = func;
137 }
```

5.19.2.7 sys_set_malloc()

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

Definition at line 124 of file mpx_supt.c.

```
125 {
126     student_malloc = func;
127 }
```

5.20 modules/R1/commhand.c File Reference

```
#include <core/serial.h>
#include <string.h>
#include "../mpx_supt.h"
#include "R1commands.h"
```

Functions

- int [commhand](#) ()

5.20.1 Function Documentation

5.20.1.1 commhand()

int commhand ()

Definition at line 8 of file commhand.c.

```
9 {
10
11     char cmdBuffer[100];
12     int bufferSize;
13
14     int quit = 0;
15
16     while (!quit)
17     {
18         //get a command: cal polling fx
19
20         memset(cmdBuffer, '\0', 100);
21
22         bufferSize = 99; // reset size before each call to read
23
24         sys_req(READ, DEFAULT_DEVICE, cmdBuffer, &bufferSize);
25
26         serial_print("\n");
27
28         if (strcmp(cmdBuffer, "help") == 0)
29         {
30             help();
31         }
32         else if (strcmp(cmdBuffer, "version") == 0)
33         {
34             version();
35         }
36         else if (strcmp(cmdBuffer, "getDate") == 0)
37         {
38             getDate();
39         }
40         else if (strcmp(cmdBuffer, "setDate") == 0)
41         {
42             setDate();
43         }
44         else if (strcmp(cmdBuffer, "getTime") == 0)
45         {
46             getTime();
47         }
48         else if (strcmp(cmdBuffer, "setTime") == 0)
49         {
50             setTime();
51         }
52         else if (strcmp(cmdBuffer, "quit") == 0)
53         {
54
55             // Need a check here
```

```

56
57     quit = 1;
58 }
59 else
60 {
61     char message[] = "Unrecognized Command\n";
62
63     int tempBuffer = strlen(message);
64
65     sys_req(WRITE, DEFAULT_DEVICE, (char *)message, &tempBuffer);
66 }
67
68 // process the command: take array buffer chars and make a string. Decide what the cmd wants to
69 do
70 // see if quit was entered: if string == quit = 1
71 }
72 return 0;
73 }

```

5.21 modules/R1/commhand.h File Reference

Functions

- int `commhand()`

5.21.1 Function Documentation

5.21.1.1 `commhand()`

```
int commhand ( )
```

Definition at line 8 of file `commhand.c`.

```

9 {
10
11     char cmdBuffer[100];
12     int bufferSize;
13
14     int quit = 0;
15
16     while (!quit)
17     {
18         //get a command: cal polling fx
19
20         memset(cmdBuffer, '\0', 100);
21
22         bufferSize = 99; // reset size before each call to read
23
24         sys_req(READ, DEFAULT_DEVICE, cmdBuffer, &bufferSize);
25
26         serial_print("\n");
27
28         if (strcmp(cmdBuffer, "help") == 0)
29         {
30             help();
31         }
32         else if (strcmp(cmdBuffer, "version") == 0)
33         {
34             version();
35         }
36         else if (strcmp(cmdBuffer, "getDate") == 0)
37         {
38             getDate();
39         }
40         else if (strcmp(cmdBuffer, "setDate") == 0)
41         {
42             setDate();

```

```

43     }
44     else if (strcmp(cmdBuffer, "getTime") == 0)
45     {
46         getTime();
47     }
48     else if (strcmp(cmdBuffer, "setTime") == 0)
49     {
50         setTime();
51     }
52     else if (strcmp(cmdBuffer, "quit") == 0)
53     {
54
55         // Need a check here
56
57         quit = 1;
58     }
59     else
60     {
61         char message[] = "Unrecognized Command\n";
62         int tempBuffer = strlen(message);
63
64         sys_req(WRITE, DEFAULT_DEVICE, (char *)message, &tempBuffer);
65     }
66
67     // process the command: take array buffer chars and make a string. Decide what the cmd wants to
68     do
69     // see if quit was entered: if string == quit = 1
70     }
71
72     return 0;
73 }

```

5.22 modules/R1/R1commands.c File Reference

```

#include <core/serial.h>
#include <string.h>
#include "../mpx_supt.h"
#include <core/io.h>

```

Functions

- int [BCDtoChar](#) (unsigned char test, char *buffer)
- unsigned char [intToBCD](#) (int test)
- int [help](#) ()
- int [version](#) ()
- void [getTime](#) ()
- int [setTime](#) ()
- void [getDate](#) ()
- int [setDate](#) ()
- unsigned char [change_int_to_binary](#) (int test)

5.22.1 Function Documentation

5.22.1.1 BCDtoChar()

```
int BCDtoChar (
    unsigned char test,
    char * buffer )
```

Definition at line 464 of file R1commands.c.

```
464                                     {
465
466     int val1 = (test/16);
467     int val2 =(test%16);
468
469     buffer[0] = val1+'0';
470     buffer[1] = val2+'0';
471
472     return 0;
473 }
```

5.22.1.2 change_int_to_binary()

```
unsigned char change_int_to_binary (
    int test )
```

Definition at line 457 of file R1commands.c.

```
457                                     {
458
459     return (((test/10)<<4)|(test%10));
460
461 }
```

5.22.1.3 getDate()

```
void getDate ( )
```

Definition at line 229 of file R1commands.c.

```
229     {
230
231     char buffer[4]="\0\0\0\0";
232     int count = 4;
233     char divider = '/';
234     char newLine[1] = "\n";
235     int newLineCount = 1;
236
237     outb(0x70, 0x07); // getting Day of month value
238     BCDtoChar(inb(0x71), buffer);
239     buffer[2] = divider;
240     sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
241
242     outb(0x70, 0x08); // getting Month value
243     BCDtoChar(inb(0x71), buffer);
244     buffer[2] = divider;
245     sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
246
247     outb(0x70, 0x32); // getting Year value second byte
248     BCDtoChar(inb(0x71), buffer);
249     buffer[2] = '\0';
250     sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
251
252     outb(0x70, 0x09); // getting Year value first byte
253     BCDtoChar(inb(0x71), buffer);
254     sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
255
256     sys_req(WRITE, DEFAULT_DEVICE, newLine, &newLineCount);
257
258     return 0;
259 }
```

5.22.1.4 getTime()

```
void getTime ( )
```

Definition at line 87 of file R1commands.c.

```

87     {
88
89         char buffer[4]="\0\0\0";
90         int count = 4;
91         char divider = ':';
92         char newLine[1] = "\n";
93         int newLineCount = 1;
94
95         outb(0x70, 0x04); // getting Hour value
96         BCDtoChar(inb(0x71), buffer);
97         buffer[2] = divider;
98         sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
99
100        outb(0x70, 0x02); // getting Minute value
101        BCDtoChar(inb(0x71), buffer);
102        buffer[2] = divider;
103        sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
104
105        outb(0x70, 0x00); // getting Second value
106        BCDtoChar(inb(0x71), buffer);
107        buffer[2] = '\0';
108        sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
109
110        return 0;
111    }
112 }
```

5.22.1.5 help()

```
int help ( )
```

Definition at line 13 of file R1commands.c.

```

13     {
14
15         // Help Description section
16         char helpDesc[] = "Help: Returns basic command information.\n";
17
18         int tempBuffer = strlen(helpDesc);
19
20         sys_req(WRITE, DEFAULT_DEVICE, (char*)helpDesc, &tempBuffer);
21
22
23         // Version Description section
24         char versionDesc[] = "Version: Returns the current version of the software.\n";
25
26         tempBuffer = strlen(versionDesc);
27
28         sys_req(WRITE, DEFAULT_DEVICE, (char*)versionDesc, &tempBuffer);
29
30
31         // getTime Description section
32         char getTimeDesc[] = "getTime: Returns the current set time.\n";
33
34         tempBuffer = strlen(getTimeDesc);
35
36         sys_req(WRITE, DEFAULT_DEVICE, (char*)getTimeDesc, &tempBuffer);
37
38
39         // setTime Description section
40         char setTimeDesc[] = "setTime: Allows the user to change the set time.\n";
41
42         tempBuffer = strlen(setTimeDesc);
43
44         sys_req(WRITE, DEFAULT_DEVICE, (char*)setTimeDesc, &tempBuffer);
45
46
47         // getDate Description section
48         char getDateDesc[] = "getDate: Returns the current set date.\n";
49
50         tempBuffer = strlen(getDateDesc);
51     }
```



```

52     sys_req(WRITE, DEFAULT_DEVICE, (char*)getDateDesc, &tempBuffer);
53
54
55     // setDate Description section
56     char setDateDesc[] = "setDate: Allows the user to change the set date.\n";
57
58     tempBuffer = strlen(setDateDesc);
59
60     sys_req(WRITE, DEFAULT_DEVICE, (char*)setDateDesc, &tempBuffer);
61
62
63     // quit Description section
64     char quitDesc[] = "Quit: Allows the user to shut the system down.\n";
65
66     tempBuffer = strlen(quitDesc);
67
68     sys_req(WRITE, DEFAULT_DEVICE, (char*)quitDesc, &tempBuffer);
69
70     return 0;
71 }

```

5.22.1.6 intToBCD()

```

unsigned char intToBCD (
    int test )

```

5.22.1.7 setDate()

```

int setDate ( )

```

Definition at line 262 of file R1commands.c.

```

262     {
263
264         int count = 4; // used to print year
265
266         char spacer[1] = "\n"; // used to space out terminal outputs
267         int spaceCount = 1;
268
269
270         char instruction1[] = "Please type the desired year. I.E.: yyyy.\n";
271         int length = strlen(instruction1);
272
273         sys_req(WRITE, DEFAULT_DEVICE, instruction1, &length);
274
275         char year[5] = "\0\0\0\0\0"; // year buffer
276
277         int flag = 0; // thrown if input is invalid
278
279         do{
280             sys_req(READ, DEFAULT_DEVICE, year, &count);
281             if(atoi(year) > 0){
282
283                 sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
284                 flag = 0;
285
286                 char yearUpper[3] = "\0\0\0";
287                 char yearLower[3] = "\0\0\0";
288
289                 yearUpper[0] = year[0];
290                 yearUpper[1] = year[1];
291                 yearLower[0] = year[2];
292                 yearLower[1] = year[3];
293
294
295                 cli();
296
297                 outb(0x70, 0x32); // Setting first byte year value
298                 outb(0x71, intToBCD(atoi(yearUpper)));
299
300                 outb(0x70, 0x09); // Setting second byte year value
301                 outb(0x71, intToBCD(atoi(yearLower)));
302

```

```

303
304         sti();
305     }
306 }
307 else{
308     char invalid[] = "Invalid year.\n";
309     int lengthInval = strlen(invalid);
310     sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
311     sys_req(WRITE, DEFAULT_DEVICE, invalid, &lengthInval);
312     flag = 1;
313 }
314 }while(flag == 1);
315
316
317
318
319 char instruction2[] = "Please type the desired month. I.E.: mm.\n";
320 length = strlen(instruction2);
321
322 sys_req(WRITE, DEFAULT_DEVICE, instruction2, &length);
323
324
325 char month[4] = "\0\0\n\0";
326 count = 4; // used to print month
327
328 do{
329     sys_req(READ, DEFAULT_DEVICE, month, &count);
330     if(atoi(month)<13 && atoi(month) > 0){
331
332         sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
333         flag = 0;
334
335         cli();
336
337         outb(0x70, 0x08); // Setting month value
338         outb(0x71, intToBCD(atoi(month)));
339
340         sti();
341     }
342 }
343 else{
344     char invalid[] = "Invalid month.\n";
345     int lengthInval = strlen(invalid);
346     sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
347     sys_req(WRITE, DEFAULT_DEVICE, invalid, &lengthInval);
348     flag = 1;
349 }
350 }while(flag == 1);
351
352
353
354 char instruction3[] = "Please type the desired day of month. I.E.: dd.\n";
355 length = strlen(instruction3);
356 sys_req(WRITE, DEFAULT_DEVICE, instruction3, &length);
357
358
359 char day[4] = "\0\0\n\0";
360 count = 4; // used to print day
361
362
363 do{
364     sys_req(READ, DEFAULT_DEVICE, day, &count);
365     sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
366     if((atoi(year) % 4 == 0 && atoi(year) % 100 != 0) || atoi(year) % 400 == 0){ // checking for
367 leap year
368
369         char leapYear[] = "This is a leap year. February has 29 days.\n";
370         length = strlen(leapYear);
371
372         sys_req(WRITE, DEFAULT_DEVICE, leapYear, &length);
373
374
375         if((atoi(month) == 1 || atoi(month) == 3 || atoi(month) == 5 || atoi(month) == 7
376 ||atoi(month) == 8 || atoi(month) == 10 || atoi(month) == 12) && atoi(day) > 31){
377             flag = 1;
378             char invalid[] = "Invalid day.\n";
379             length = strlen(invalid);
380             sys_req(WRITE, DEFAULT_DEVICE, invalid, &length);
381         }
382         else if((atoi(month) == 4 || atoi(month) == 6 || atoi(month) == 9 || atoi(month) == 11) &&
383 atoi(day) >30){
384             flag = 1;
385             char invalid[] = "Invalid day.\n";
386             length = strlen(invalid);
387             sys_req(WRITE, DEFAULT_DEVICE, invalid, &length);
388         }
389         else if((atoi(month) == 2) && atoi(day) >29){
390             flag = 1;

```

```

389         char invalid[] = "Invalid day.\n";
390         length = strlen(invalid);
391         sys_req(WRITE, DEFAULT_DEVICE, invalid, &length);
392     }
393     else{
394
395         flag = 0;
396         cli();
397
398         outb(0x70, 0x07); // Setting day of month value
399         outb(0x71, intToBCD(atoi(day)));
400
401         sti();
402
403     }
404 }
405
406 }
407 else if(atoi(year) %4 != 0 || atoi(year) %400 !=0){ // checking for leap year
408
409     char noLeap[] = "This is not a leap year.\n";
410     length = strlen(noLeap);
411     sys_req(WRITE, DEFAULT_DEVICE, noLeap, &length);
412
413
414     if((atoi(month) == 1 || atoi(month) == 3 || atoi(month) == 5 || atoi(month) == 7
||atoi(month) == 8 || atoi(month) == 10 || atoi(month) == 12) && atoi(day) > 31){
415         flag = 1;
416         char invalid[] = "Invalid day.\n";
417         length = strlen(invalid);
418         sys_req(WRITE, DEFAULT_DEVICE, invalid, &length);
419     }
420     else if((atoi(month) == 4 || atoi(month) == 6 || atoi(month) == 9 || atoi(month) == 11) &&
atoi(day) >30){
421         flag = 1;
422         char invalid[] = "Invalid day.\n";
423         length = strlen(invalid);
424         sys_req(WRITE, DEFAULT_DEVICE, invalid, &length);
425     }
426     else if((atoi(month) == 2) && atoi(day) >28){
427         flag = 1;
428         char invalid[] = "Invalid day.\n";
429         length = strlen(invalid);
430         sys_req(WRITE, DEFAULT_DEVICE, invalid, &length);
431     }
432     else{
433
434         cli();
435
436         outb(0x70, 0x07); // Setting day of month value
437         outb(0x71, intToBCD(atoi(day)));
438
439         sti();
440
441     }
442 }
443 }
444
445 }while(flag == 1);
446
447
448 char exitMessage[] = "The date has been set.\n";
449 int exitLength = strlen(exitMessage);
450 sys_req(WRITE, DEFAULT_DEVICE, exitMessage, &exitLength);
451
452
453 return 0;
454 }

```

5.22.1.8 setTime()

```
int setTime ( )
```

Definition at line 115 of file R1commands.c.

```

115     {
116
117         int count = 4; // counter for printing
118
119         char spacer[1] = "\n"; // used to space out terminal outputs

```

```

120     int spaceCount = 1;
121
122     char instruction1[] = "Please type the desired hours. I.E.: hh.\n";
123
124     int length = strlen(instruction1);
125
126     sys_req(WRITE, DEFAULT_DEVICE, instruction1, &length);
127
128     char hour[4] = "\0\0\n\0";
129
130     int flag = 0;
131
132     do{
133         sys_req(READ, DEFAULT_DEVICE, hour, &count);
134         if(atoi(hour)<24 && atoi(hour) >= 0){
135             sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
136             flag = 0;
137         }
138         else{
139             char invalid[] = "Invalid hours.\n";
140             int lengthInval = strlen(invalid);
141             sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
142             sys_req(WRITE, DEFAULT_DEVICE, invalid, &lengthInval);
143             flag = 1;
144         }
145     }while(flag == 1);
146
147     char instruction2[] = "Please type the desired minutes. I.E.: mm.\n";
148
149     length = strlen(instruction2);
150
151     sys_req(WRITE, DEFAULT_DEVICE, instruction2, &length);
152
153     char minute[4] = "\0\0\n\0";
154
155     do{
156         sys_req(READ, DEFAULT_DEVICE, minute, &count);
157         if(atoi(minute)<60 && atoi(minute) >= 0){
158             sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
159             flag = 0;
160         }
161         else{
162             char invalid[] = "Invalid minutes.\n";
163             int lengthInval = strlen(invalid);
164             sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
165             sys_req(WRITE, DEFAULT_DEVICE, invalid, &lengthInval);
166             flag = 1;
167         }
168     }while(flag == 1);
169
170     char instruction3[] = "Please type the desired seconds. I.E.: ss.\n";
171
172     length = strlen(instruction3);
173
174     sys_req(WRITE, DEFAULT_DEVICE, instruction3, &length);
175
176     char second[4] = "\0\0\n\0";
177
178     do{
179         sys_req(READ, DEFAULT_DEVICE, second, &count);
180         if(atoi(second)<60 && atoi(second) >= 0){
181             sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
182             flag = 0;
183         }
184         else{
185             char invalid[] = "Invalid seconds.\n";
186             int lengthInval = strlen(invalid);
187             sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
188             sys_req(WRITE, DEFAULT_DEVICE, invalid, &lengthInval);
189             flag = 1;
190         }
191     }while(flag == 1);
192
193     cli();
194
195     outb(0x70, 0x04); // Hour
196     outb(0x71, intToBCD(atoi(hour)));

```

```

210
211
212     outb(0x70, 0x02); // Minute
213     outb(0x71, intToBCD(atoi(minute)));
214
215
216     outb(0x70, 0x00); // Second
217     outb(0x71, intToBCD(atoi(second)));
218
219     sti();
220
221     char exitMessage[] = "The time has been set.\n";
222     int exitLength = strlen(exitMessage);
223     sys_req(WRITE, DEFAULT_DEVICE, exitMessage, &exitLength);
224
225
226     return 0;
227 }

```

5.22.1.9 version()

```
int version ( )
```

Definition at line 74 of file R1commands.c.

```

74     {
75
76         char version[] = "Version 1.0\n";
77
78         int tempBuffer = strlen(version);
79
80         sys_req(WRITE, DEFAULT_DEVICE, (char*)version, &tempBuffer);
81
82         return 0;
83
84     }

```

5.23 modules/R1/R1commands.h File Reference

Functions

- void [help](#) ()
- void [version](#) ()
- void [getTime](#) ()
- void [setTime](#) ()
- void [getDate](#) ()
- void [setDate](#) ()
- unsigned int [change_int_to_binary](#) (int test)
- int [BCDtoChar](#) (unsigned char test, char *buffer)

5.23.1 Function Documentation

5.23.1.1 BCDtoChar()

```
int BCDtoChar (
    unsigned char test,
    char * buffer )
```

Definition at line 464 of file R1commands.c.

```
464                                     {
465
466     int val1 = (test/16);
467     int val2 =(test%16);
468
469     buffer[0] = val1+'0';
470     buffer[1] = val2+'0';
471
472     return 0;
473 }
```

5.23.1.2 change_int_to_binary()

```
unsigned int change_int_to_binary (
    int test )
```

Definition at line 457 of file R1commands.c.

```
457                                     {
458
459     return (((test/10)«4)|(test%10));
460
461 }
```

5.23.1.3 getDate()

```
void getDate ( )
```

Definition at line 229 of file R1commands.c.

```
229     {
230
231     char buffer[4]="\0\0\0\0";
232     int count = 4;
233     char divider = '/';
234     char newLine[1] = "\n";
235     int newLineCount = 1;
236
237     outb(0x70, 0x07); // getting Day of month value
238     BCDtoChar(inb(0x71), buffer);
239     buffer[2] = divider;
240     sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
241
242     outb(0x70, 0x08); // getting Month value
243     BCDtoChar(inb(0x71), buffer);
244     buffer[2] = divider;
245     sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
246
247     outb(0x70, 0x32); // getting Year value second byte
248     BCDtoChar(inb(0x71), buffer);
249     buffer[2] = '\0';
250     sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
251
252     outb(0x70, 0x09); // getting Year value first byte
253     BCDtoChar(inb(0x71), buffer);
254     sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
255
256     sys_req(WRITE, DEFAULT_DEVICE, newLine, &newLineCount);
257
258     return 0;
259 }
```

5.23.1.4 getTime()

```
void getTime ( )
```

Definition at line 87 of file R1commands.c.

```

87     {
88
89         char buffer[4]="\0\0\0";
90         int count = 4;
91         char divider = ':';
92         char newLine[1] = "\n";
93         int newLineCount = 1;
94
95         outb(0x70, 0x04); // getting Hour value
96         BCDtoChar(inb(0x71), buffer);
97         buffer[2] = divider;
98         sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
99
100        outb(0x70, 0x02); // getting Minute value
101        BCDtoChar(inb(0x71), buffer);
102        buffer[2] = divider;
103        sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
104
105        outb(0x70, 0x00); // getting Second value
106        BCDtoChar(inb(0x71), buffer);
107        buffer[2] = '\0';
108        sys_req(WRITE, DEFAULT_DEVICE, buffer, &count);
109
110        return 0;
111    }
112 }
```

5.23.1.5 help()

```
void help ( )
```

Definition at line 13 of file R1commands.c.

```

13     {
14
15         // Help Description section
16         char helpDesc[] = "Help: Returns basic command information.\n";
17
18         int tempBuffer = strlen(helpDesc);
19
20         sys_req(WRITE, DEFAULT_DEVICE, (char*)helpDesc, &tempBuffer);
21
22
23         // Version Description section
24         char versionDesc[] = "Version: Returns the current version of the software.\n";
25
26         tempBuffer = strlen(versionDesc);
27
28         sys_req(WRITE, DEFAULT_DEVICE, (char*)versionDesc, &tempBuffer);
29
30
31         // getTime Description section
32         char getTimeDesc[] = "getTime: Returns the current set time.\n";
33
34         tempBuffer = strlen(getTimeDesc);
35
36         sys_req(WRITE, DEFAULT_DEVICE, (char*)getTimeDesc, &tempBuffer);
37
38
39         // setTime Description section
40         char setTimeDesc[] = "setTime: Allows the user to change the set time.\n";
41
42         tempBuffer = strlen(setTimeDesc);
43
44         sys_req(WRITE, DEFAULT_DEVICE, (char*)setTimeDesc, &tempBuffer);
45
46
47         // getDate Description section
48         char getDateDesc[] = "getDate: Returns the current set date.\n";
49
50         tempBuffer = strlen(getDateDesc);
51     }
```

```

52     sys_req(WRITE, DEFAULT_DEVICE, (char*)getDateDesc, &tempBuffer);
53
54
55     // setDate Description section
56     char setDateDesc[] = "setDate: Allows the user to change the set date.\n";
57
58     tempBuffer = strlen(setDateDesc);
59
60     sys_req(WRITE, DEFAULT_DEVICE, (char*)setDateDesc, &tempBuffer);
61
62
63     // quit Description section
64     char quitDesc[] = "Quit: Allows the user to shut the system down.\n";
65
66     tempBuffer = strlen(quitDesc);
67
68     sys_req(WRITE, DEFAULT_DEVICE, (char*)quitDesc, &tempBuffer);
69
70     return 0;
71 }

```

5.23.1.6 setDate()

```
void setDate ( )
```

Definition at line 262 of file R1commands.c.

```

262     {
263
264         int count = 4; // used to print year
265
266         char spacer[1] = "\n"; // used to space out terminal outputs
267         int spaceCount = 1;
268
269
270         char instruction1[] = "Please type the desired year. I.E.: yyyy.\n";
271         int length = strlen(instruction1);
272
273         sys_req(WRITE, DEFAULT_DEVICE, instruction1, &length);
274
275
276         char year[5] = "\0\0\0\0\0"; // year buffer
277
278         int flag = 0; // thrown if input is invalid
279
280         do{
281             sys_req(READ, DEFAULT_DEVICE, year, &count);
282             if(atoi(year) > 0){
283
284                 sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
285                 flag = 0;
286
287                 char yearUpper[3] = "\0\0\0";
288                 char yearLower[3] = "\0\0\0";
289
290                 yearUpper[0] = year[0];
291                 yearUpper[1] = year[1];
292                 yearLower[0] = year[2];
293                 yearLower[1] = year[3];
294
295
296                 cli();
297
298                 outb(0x70, 0x32); // Setting first byte year value
299                 outb(0x71, intToBCD(atoi(yearUpper)));
300
301                 outb(0x70, 0x09); // Setting second byte year value
302                 outb(0x71, intToBCD(atoi(yearLower)));
303
304                 sti();
305
306             }
307             else{
308                 char invalid[] = "Invalid year.\n";
309                 int lengthInval = strlen(invalid);
310                 sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
311                 sys_req(WRITE, DEFAULT_DEVICE, invalid, &lengthInval);
312                 flag = 1;
313             }
314         }while(flag == 1);
315

```



```

316
317
318
320     char instruction2[] = "Please type the desired month. I.E.: mm.\n";
321     length = strlen(instruction2);
322
323     sys_req(WRITE, DEFAULT_DEVICE, instruction2, &length);
324
325     char month[4] = "\0\0\n\0";
326     count = 4; // used to print month
327
328     do{
329         sys_req(READ, DEFAULT_DEVICE, month, &count);
330         if(atoi(month)<13 && atoi(month) > 0){
331
332             sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
333             flag = 0;
334
335             cli();
336
337             outb(0x70, 0x08); // Setting month value
338             outb(0x71, intToBCD(atoi(month)));
339
340             sti();
341
342         }
343         else{
344             char invalid[] = "Invalid month.\n";
345             int lengthInvalid = strlen(invalid);
346             sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
347             sys_req(WRITE, DEFAULT_DEVICE, invalid, &lengthInvalid);
348             flag = 1;
349         }
350     }while(flag == 1);
351
352
353
355     char instruction3[] = "Please type the desired day of month. I.E.: dd.\n";
356
357     length = strlen(instruction3);
358     sys_req(WRITE, DEFAULT_DEVICE, instruction3, &length);
359
360     char day[4] = "\0\0\n\0";
361     count = 4; // used to print day
362
363
364     do{
365         sys_req(READ, DEFAULT_DEVICE, day, &count);
366         sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
367         if((atoi(year) % 4 == 0 && atoi(year) % 100 != 0) || atoi(year) % 400 == 0){ // checking for
368         leap year
369
370             char leapYear[] = "This is a leap year. February has 29 days.\n";
371             length = strlen(leapYear);
372
373             sys_req(WRITE, DEFAULT_DEVICE, leapYear, &length);
374
375             if((atoi(month) == 1 || atoi(month) == 3 || atoi(month) == 5 || atoi(month) == 7
376             ||atoi(month) == 8 || atoi(month) == 10 || atoi(month) == 12) && atoi(day) > 31){
377                 flag = 1;
378                 char invalid[] = "Invalid day.\n";
379                 length = strlen(invalid);
380                 sys_req(WRITE, DEFAULT_DEVICE, invalid, &length);
381             }
382             else if((atoi(month) == 4 || atoi(month) == 6 || atoi(month) == 9 || atoi(month) == 11) &&
383             atoi(day) >30){
384                 flag = 1;
385                 char invalid[] = "Invalid day.\n";
386                 length = strlen(invalid);
387                 sys_req(WRITE, DEFAULT_DEVICE, invalid, &length);
388             }
389             else if((atoi(month) == 2) && atoi(day) >29){
390                 flag = 1;
391                 char invalid[] = "Invalid day.\n";
392                 length = strlen(invalid);
393                 sys_req(WRITE, DEFAULT_DEVICE, invalid, &length);
394             }
395             else{
396
397                 flag = 0;
398                 cli();
399
400                 outb(0x70, 0x07); // Setting day of month value
401                 outb(0x71, intToBCD(atoi(day)));
402
403                 sti();

```

```

402
403
404     }
405
406     }
407     else if(atoi(year) %4 != 0 || atoi(year) %400 !=0){ // checking for leap year
408
409         char noLeap[] = "This is not a leap year.\n";
410         length = strlen(noLeap);
411         sys_req(WRITE, DEFAULT_DEVICE, noLeap, &length);
412
413
414         if((atoi(month) == 1 || atoi(month) == 3 || atoi(month) == 5 || atoi(month) == 7
||atoi(month) == 8 || atoi(month) == 10 || atoi(month) == 12) && atoi(day) > 31){
415             flag = 1;
416             char invalid[] = "Invalid day.\n";
417             length = strlen(invalid);
418             sys_req(WRITE, DEFAULT_DEVICE, invalid, &length);
419         }
420         else if((atoi(month) == 4 || atoi(month) == 6 || atoi(month) == 9 || atoi(month) == 11) &&
atoi(day) >30){
421             flag = 1;
422             char invalid[] = "Invalid day.\n";
423             length = strlen(invalid);
424             sys_req(WRITE, DEFAULT_DEVICE, invalid, &length);
425         }
426         else if((atoi(month) == 2) && atoi(day) >28){
427             flag = 1;
428             char invalid[] = "Invalid day.\n";
429             length = strlen(invalid);
430             sys_req(WRITE, DEFAULT_DEVICE, invalid, &length);
431         }
432         else{
433
434             cli();
435
436             outb(0x70, 0x07); // Setting day of month value
437             outb(0x71, intToBCD(atoi(day)));
438
439             sti();
440
441         }
442
443     }
444
445 }while(flag == 1);
446
447
448 char exitMessage[] = "The date has been set.\n";
449 int exitLength = strlen(exitMessage);
450 sys_req(WRITE, DEFAULT_DEVICE, exitMessage, &exitLength);
451
452
453 return 0;
454 }

```

5.23.1.7 setTime()

```
void setTime ( )
```

Definition at line 115 of file R1commands.c.

```

115     {
116
117     int count = 4; // counter for printing
118
119     char spacer[1] = "\n"; // used to space out terminal outputs
120     int spaceCount = 1;
121
122     char instruction1[] = "Please type the desired hours. I.E.: hh.\n";
123
124     int length = strlen(instruction1);
125
126     sys_req(WRITE, DEFAULT_DEVICE, instruction1, &length);
127
128     char hour[4] = "\0\0\n\0";
129
130     int flag = 0;
131
132     do{
133

```

```

134     sys_req(READ, DEFAULT_DEVICE, hour, &count);
135     if(atoi(hour)<24 && atoi(hour) >= 0){
136
137         sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
138         flag = 0;
139     }
140     else{
141         char invalid[] = "Invalid hours.\n";
142         int lengthInval = strlen(invalid);
143         sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
144         sys_req(WRITE, DEFAULT_DEVICE, invalid, &lengthInval);
145         flag = 1;
146     }
147 }while(flag == 1);
148
149
150
151 char instruction2[] = "Please type the desired minutes. I.E.: mm.\n";
152
153 length = strlen(instruction2);
154
155 sys_req(WRITE, DEFAULT_DEVICE, instruction2, &length);
156
157 char minute[4] = "\0\0\n\0";
158
159 do{
160     sys_req(READ, DEFAULT_DEVICE, minute, &count);
161     if(atoi(minute)<60 && atoi(minute) >= 0){
162
163         sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
164         flag = 0;
165     }
166     else{
167         char invalid[] = "Invalid minutes.\n";
168         int lengthInval = strlen(invalid);
169         sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
170         sys_req(WRITE, DEFAULT_DEVICE, invalid, &lengthInval);
171         flag = 1;
172     }
173 }while(flag == 1);
174
175
176
177 char instruction3[] = "Please type the desired seconds. I.E.: ss.\n";
178
179 length = strlen(instruction3);
180
181 sys_req(WRITE, DEFAULT_DEVICE, instruction3, &length);
182
183 char second[4] = "\0\0\n\0";
184
185 do{
186     sys_req(READ, DEFAULT_DEVICE, second, &count);
187     if(atoi(second)<60 && atoi(second) >= 0){
188
189         sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
190         flag = 0;
191     }
192     else{
193         char invalid[] = "Invalid seconds.\n";
194         int lengthInval = strlen(invalid);
195         sys_req(WRITE, DEFAULT_DEVICE, spacer, &spaceCount);
196         sys_req(WRITE, DEFAULT_DEVICE, invalid, &lengthInval);
197         flag = 1;
198     }
199 }while(flag == 1);
200
201 cli();
202
203
204 outb(0x70, 0x04); // Hour
205 outb(0x71, intToBCD(atoi(hour)));
206
207
208 outb(0x70, 0x02); // Minute
209 outb(0x71, intToBCD(atoi(minute)));
210
211
212 outb(0x70, 0x00); // Second
213 outb(0x71, intToBCD(atoi(second)));
214
215
216 sti();
217
218 char exitMessage[] = "The time has been set.\n";
219 int exitLength = strlen(exitMessage);

```

```
223     sys_req(WRITE, DEFAULT_DEVICE, exitMessage, &exitLength);
224
225
226     return 0;
227 }
```

5.23.1.8 version()

```
void version ( )
```

Definition at line 74 of file R1commands.c.

```
74     {
75
76         char version[] = "Version 1.0\n";
77
78         int tempBuffer = strlen(version);
79
80         sys_req(WRITE, DEFAULT_DEVICE, (char*)version, &tempBuffer);
81
82         return 0;
83
84     }
```

5.24 README.md File Reference

Index

- `__attribute__`
 - `tables.h`, [30](#)
 - `__end`
 - `heap.c`, [68](#)
 - `_end`
 - `heap.c`, [68](#)
 - `_kmalloc`
 - `heap.c`, [67](#)
 - `heap.h`, [35](#)
- `access`
 - `gdt_entry_struct`, [11](#)
 - `tables.h`, [32](#)
- `accessed`
 - `page_entry`, [19](#)
- `alloc`
 - `heap.c`, [67](#)
 - `heap.h`, [35](#)
- `asm`
 - `system.h`, [44](#)
- `atoi`
 - `string.c`, [75](#)
 - `string.h`, [40](#)
- `base`
 - `gdt_descriptor_struct`, [10](#)
 - `heap`, [13](#)
 - `idt_struct`, [15](#)
 - `tables.h`, [32](#)
- `base_high`
 - `gdt_entry_struct`, [11](#)
 - `idt_entry_struct`, [14](#)
 - `tables.h`, [32](#)
- `base_low`
 - `gdt_entry_struct`, [11](#)
 - `idt_entry_struct`, [14](#)
 - `tables.h`, [32](#)
- `base_mid`
 - `gdt_entry_struct`, [11](#)
 - `tables.h`, [32](#)
- `BCDtoChar`
 - `R1commands.c`, [90](#)
 - `R1commands.h`, [97](#)
- `block`
 - `index_entry`, [16](#)
- `bounds`
 - `interrupts.c`, [49](#)
- `breakpoint`
 - `interrupts.c`, [49](#)
- `buffer_ptr`
 - `param`, [21](#)
- `cdir`
 - `paging.c`, [73](#)
- `change_int_to_binary`
 - `R1commands.c`, [91](#)
 - `R1commands.h`, [98](#)
- `clear_bit`
 - `paging.c`, [70](#)
 - `paging.h`, [37](#)
- `cli`
 - `system.h`, [44](#)
- `COM1`
 - `serial.h`, [25](#)
- `COM2`
 - `serial.h`, [26](#)
- `COM3`
 - `serial.h`, [26](#)
- `COM4`
 - `serial.h`, [26](#)
- `COM_PORT`
 - `mpx_supt.h`, [82](#)
- `commhand`
 - `commhand.c`, [88](#)
 - `commhand.h`, [89](#)
- `commhand.c`
 - `commhand`, [88](#)
- `commhand.h`
 - `commhand`, [89](#)
- `coprocessor`
 - `interrupts.c`, [50](#)
- `coprocessor_segment`
 - `interrupts.c`, [50](#)
- `count_ptr`
 - `param`, [21](#)
- `curr_heap`
 - `heap.c`, [69](#)
- `current_module`
 - `mpx_supt.c`, [80](#)
- `date_time`, [7](#)
 - `day_m`, [7](#)
 - `day_w`, [7](#)
 - `day_y`, [8](#)
 - `hour`, [8](#)
 - `min`, [8](#)
 - `mon`, [8](#)
 - `sec`, [8](#)
 - `year`, [8](#)
- `day_m`

- date_time, [7](#)
- day_w
 - date_time, [7](#)
- day_y
 - date_time, [8](#)
- debug
 - interrupts.c, [50](#)
- DEFAULT_DEVICE
 - mpx_supt.h, [82](#)
- device_id
 - param, [22](#)
- device_not_available
 - interrupts.c, [50](#)
- dirty
 - page_entry, [19](#)
- divide_error
 - interrupts.c, [50](#)
- do_bounds
 - interrupts.c, [50](#)
- do_breakpoint
 - interrupts.c, [50](#)
- do_coprocessor
 - interrupts.c, [51](#)
- do_coprocessor_segment
 - interrupts.c, [51](#)
- do_debug
 - interrupts.c, [51](#)
- do_device_not_available
 - interrupts.c, [51](#)
- do_divide_error
 - interrupts.c, [51](#)
- do_double_fault
 - interrupts.c, [52](#)
- do_general_protection
 - interrupts.c, [52](#)
- do_invalid_op
 - interrupts.c, [52](#)
- do_invalid_tss
 - interrupts.c, [52](#)
- do_isr
 - interrupts.c, [52](#)
- do_nmi
 - interrupts.c, [53](#)
- do_overflow
 - interrupts.c, [53](#)
- do_page_fault
 - interrupts.c, [53](#)
- do_reserved
 - interrupts.c, [53](#)
- do_segment_not_present
 - interrupts.c, [53](#)
- do_stack_segment
 - interrupts.c, [54](#)
- double_fault
 - interrupts.c, [54](#)
- empty
 - index_entry, [16](#)
- end
 - heap.c, [69](#)
- EXIT
 - mpx_supt.h, [82](#)
- FALSE
 - mpx_supt.h, [83](#)
- find_free
 - paging.c, [70](#)
- first_free
 - paging.h, [38](#)
- flags
 - gdt_entry_struct, [11](#)
 - idt_entry_struct, [14](#)
 - tables.h, [33](#)
- footer, [9](#)
 - head, [9](#)
- frameaddr
 - page_entry, [19](#)
- frames
 - paging.c, [73](#)
- GDT_CS_ID
 - system.h, [44](#)
- gdt_descriptor_struct, [9](#)
 - base, [10](#)
 - limit, [10](#)
- GDT_DS_ID
 - system.h, [44](#)
- gdt_entries
 - tables.c, [66](#)
- gdt_entry_struct, [10](#)
 - access, [11](#)
 - base_high, [11](#)
 - base_low, [11](#)
 - base_mid, [11](#)
 - flags, [11](#)
 - limit_low, [11](#)
- gdt_init_entry
 - tables.c, [64](#)
 - tables.h, [31](#)
- gdt_ptr
 - tables.c, [66](#)
- general_protection
 - interrupts.c, [54](#)
- get_bit
 - paging.c, [71](#)
 - paging.h, [38](#)
- get_page
 - paging.c, [71](#)
 - paging.h, [38](#)
- getDate
 - R1commands.c, [91](#)
 - R1commands.h, [98](#)
- getTime
 - R1commands.c, [91](#)
 - R1commands.h, [98](#)
- head
 - footer, [9](#)

- header, 12
 - index_id, 12
 - size, 12
- heap, 13
 - base, 13
 - index, 13
 - max_size, 13
 - min_size, 13
- heap.c
 - __end, 68
 - _end, 68
 - _kmalloc, 67
 - alloc, 67
 - curr_heap, 69
 - end, 69
 - kdir, 69
 - kheap, 69
 - kmalloc, 68
 - make_heap, 68
 - phys_alloc_addr, 69
- heap.h
 - _kmalloc, 35
 - alloc, 35
 - init_kheap, 36
 - kfree, 36
 - KHEAP_BASE, 34
 - KHEAP_MIN, 34
 - KHEAP_SIZE, 34
 - kmalloc, 36
 - make_heap, 36
 - TABLE_SIZE, 35
- help
 - R1commands.c, 92
 - R1commands.h, 99
- hlt
 - system.h, 45
- hour
 - date_time, 8
- ICW1
 - interrupts.c, 48
- ICW4
 - interrupts.c, 49
- id
 - index_table, 17
- IDLE
 - mpx_supt.h, 83
- idle
 - mpx_supt.c, 78
 - mpx_supt.h, 85
- idt_entries
 - interrupts.c, 56
 - tables.c, 66
- idt_entry_struct, 14
 - base_high, 14
 - base_low, 14
 - flags, 14
 - sselect, 15
 - zero, 15
- idt_ptr
 - tables.c, 66
- idt_set_gate
 - tables.c, 64
 - tables.h, 31
- idt_struct, 15
 - base, 15
 - limit, 16
- inb
 - io.h, 24
- include/core/asm.h, 23
- include/core/interrupts.h, 23
- include/core/io.h, 24
- include/core/serial.h, 25
- include/core/tables.h, 30
- include/mem/heap.h, 34
- include/mem/paging.h, 37
- include/string.h, 40
- include/system.h, 43
- index
 - heap, 13
- index_entry, 16
 - block, 16
 - empty, 16
 - size, 17
- index_id
 - header, 12
- index_table, 17
 - id, 17
 - table, 17
- init_gdt
 - tables.c, 65
 - tables.h, 31
- init_idt
 - tables.c, 65
 - tables.h, 31
- init_irq
 - interrupts.c, 54
 - interrupts.h, 23
- init_kheap
 - heap.h, 36
- init_paging
 - paging.c, 71
 - paging.h, 38
- init_pic
 - interrupts.c, 54
 - interrupts.h, 24
- init_serial
 - serial.c, 59
 - serial.h, 26
- interrupts.c
 - bounds, 49
 - breakpoint, 49
 - coprocessor, 50
 - coprocessor_segment, 50
 - debug, 50
 - device_not_available, 50
 - divide_error, 50

- do_bounds, 50
- do_breakpoint, 50
- do_coprocessor, 51
- do_coprocessor_segment, 51
- do_debug, 51
- do_device_not_available, 51
- do_divide_error, 51
- do_double_fault, 52
- do_general_protection, 52
- do_invalid_op, 52
- do_invalid_tss, 52
- do_isr, 52
- do_nmi, 53
- do_overflow, 53
- do_page_fault, 53
- do_reserved, 53
- do_segment_not_present, 53
- do_stack_segment, 54
- double_fault, 54
- general_protection, 54
- ICW1, 48
- ICW4, 49
- idt_entries, 56
- init_irq, 54
- init_pic, 54
- invalid_op, 55
- invalid_tss, 55
- io_wait, 49
- isr0, 55
- nmi, 55
- overflow, 55
- page_fault, 56
- PIC1, 49
- PIC2, 49
- reserved, 56
- rtc_isr, 56
- segment_not_present, 56
- stack_segment, 56
- interrupts.h
 - init_irq, 23
 - init_pic, 24
- intToBCD
 - R1commands.c, 93
- INVALID_BUFFER
 - mpx_supt.h, 83
- INVALID_COUNT
 - mpx_supt.h, 83
- invalid_op
 - interrupts.c, 55
- INVALID_OPERATION
 - mpx_supt.h, 83
- invalid_tss
 - interrupts.c, 55
- io.h
 - inb, 24
 - outb, 25
- IO_MODULE
 - mpx_supt.h, 83
- io_wait
 - interrupts.c, 49
- iret
 - system.h, 45
- isr0
 - interrupts.c, 55
- isspace
 - string.c, 75
 - string.h, 41
- kdir
 - heap.c, 69
 - paging.c, 73
- kernel/core/interrupts.c, 47
- kernel/core/kmain.c, 57
- kernel/core/serial.c, 58
- kernel/core/system.c, 63
- kernel/core/tables.c, 64
- kernel/mem/heap.c, 66
- kernel/mem/paging.c, 70
- kfree
 - heap.h, 36
- kheap
 - heap.c, 69
 - paging.c, 73
- KHEAP_BASE
 - heap.h, 34
- KHEAP_MIN
 - heap.h, 34
- KHEAP_SIZE
 - heap.h, 34
- klogv
 - system.c, 63
 - system.h, 47
- kmain
 - kmain.c, 57
- kmain.c
 - kmain, 57
- kmalloc
 - heap.c, 68
 - heap.h, 36
- kpanic
 - system.c, 63
 - system.h, 47
- lib/string.c, 74
- limit
 - gdt_descriptor_struct, 10
 - idt_struct, 16
 - tables.h, 33
- limit_low
 - gdt_entry_struct, 11
 - tables.h, 33
- load_page_dir
 - paging.c, 72
 - paging.h, 39
- make_heap
 - heap.c, 68

- heap.h, 36
- max_size
 - heap, 13
- MEM_MODULE
 - mpx_supt.h, 84
- mem_size
 - paging.c, 74
- memset
 - string.c, 75
 - string.h, 41
- min
 - date_time, 8
- min_size
 - heap, 13
- MODULE_F
 - mpx_supt.h, 84
- MODULE_R1
 - mpx_supt.h, 84
- MODULE_R2
 - mpx_supt.h, 84
- MODULE_R3
 - mpx_supt.h, 84
- MODULE_R4
 - mpx_supt.h, 84
- MODULE_R5
 - mpx_supt.h, 85
- modules/mpx_supt.c, 78
- modules/mpx_supt.h, 81
- modules/R1/commhand.c, 88
- modules/R1/commhand.h, 89
- modules/R1/R1commands.c, 90
- modules/R1/R1commands.h, 97
- mon
 - date_time, 8
- mpx_init
 - mpx_supt.c, 78
 - mpx_supt.h, 86
- mpx_supt.c
 - current_module, 80
 - idle, 78
 - mpx_init, 78
 - params, 81
 - student_free, 81
 - student_malloc, 81
 - sys_alloc_mem, 79
 - sys_free_mem, 79
 - sys_req, 79
 - sys_set_free, 80
 - sys_set_malloc, 80
- mpx_supt.h
 - COM_PORT, 82
 - DEFAULT_DEVICE, 82
 - EXIT, 82
 - FALSE, 83
 - IDLE, 83
 - idle, 85
 - INVALID_BUFFER, 83
 - INVALID_COUNT, 83
 - INVALID_OPERATION, 83
 - IO_MODULE, 83
 - MEM_MODULE, 84
 - MODULE_F, 84
 - MODULE_R1, 84
 - MODULE_R2, 84
 - MODULE_R3, 84
 - MODULE_R4, 84
 - MODULE_R5, 85
 - mpx_init, 86
 - READ, 85
 - sys_alloc_mem, 86
 - sys_free_mem, 86
 - sys_req, 86
 - sys_set_free, 87
 - sys_set_malloc, 87
 - TRUE, 85
 - WRITE, 85
- new_frame
 - paging.c, 72
 - paging.h, 39
- nframes
 - paging.c, 74
- nmi
 - interrupts.c, 55
- NO_ERROR
 - serial.c, 59
- no_warn
 - system.h, 45
- nop
 - system.h, 45
- NULL
 - system.h, 45
- op_code
 - param, 22
- outb
 - io.h, 25
- overflow
 - interrupts.c, 55
- page_dir, 18
 - tables, 18
 - tables_phys, 18
- page_entry, 19
 - accessed, 19
 - dirty, 19
 - frameaddr, 19
 - present, 19
 - reserved, 20
 - usermode, 20
 - writable, 20
- page_fault
 - interrupts.c, 56
- PAGE_SIZE
 - paging.h, 37
- page_size
 - paging.c, 74

- page_table, [20](#)
 - pages, [21](#)
- pages
 - page_table, [21](#)
- paging.c
 - cdir, [73](#)
 - clear_bit, [70](#)
 - find_free, [70](#)
 - frames, [73](#)
 - get_bit, [71](#)
 - get_page, [71](#)
 - init_paging, [71](#)
 - kdir, [73](#)
 - kheap, [73](#)
 - load_page_dir, [72](#)
 - mem_size, [74](#)
 - new_frame, [72](#)
 - nframes, [74](#)
 - page_size, [74](#)
 - phys_alloc_addr, [74](#)
 - set_bit, [73](#)
- paging.h
 - clear_bit, [37](#)
 - first_free, [38](#)
 - get_bit, [38](#)
 - get_page, [38](#)
 - init_paging, [38](#)
 - load_page_dir, [39](#)
 - new_frame, [39](#)
 - PAGE_SIZE, [37](#)
 - set_bit, [40](#)
- param, [21](#)
 - buffer_ptr, [21](#)
 - count_ptr, [21](#)
 - device_id, [22](#)
 - op_code, [22](#)
- params
 - mpx_supt.c, [81](#)
- phys_alloc_addr
 - heap.c, [69](#)
 - paging.c, [74](#)
- PIC1
 - interrupts.c, [49](#)
- PIC2
 - interrupts.c, [49](#)
- polling
 - serial.c, [59](#)
 - serial.h, [26](#)
- present
 - page_entry, [19](#)
- R1commands.c
 - BCDtoChar, [90](#)
 - change_int_to_binary, [91](#)
 - getDate, [91](#)
 - getTime, [91](#)
 - help, [92](#)
 - intToBCD, [93](#)
 - setDate, [93](#)
 - setTime, [95](#)
 - version, [97](#)
- R1commands.h
 - BCDtoChar, [97](#)
 - change_int_to_binary, [98](#)
 - getDate, [98](#)
 - getTime, [98](#)
 - help, [99](#)
 - setDate, [100](#)
 - setTime, [102](#)
 - version, [104](#)
- READ
 - mpx_supt.h, [85](#)
- README.md, [104](#)
- reserved
 - interrupts.c, [56](#)
 - page_entry, [20](#)
- rtc_isr
 - interrupts.c, [56](#)
- sec
 - date_time, [8](#)
- segment_not_present
 - interrupts.c, [56](#)
- serial.c
 - init_serial, [59](#)
 - NO_ERROR, [59](#)
 - polling, [59](#)
 - serial_port_in, [62](#)
 - serial_port_out, [62](#)
 - serial_print, [61](#)
 - serial_println, [61](#)
 - set_serial_in, [62](#)
 - set_serial_out, [62](#)
- serial.h
 - COM1, [25](#)
 - COM2, [26](#)
 - COM3, [26](#)
 - COM4, [26](#)
 - init_serial, [26](#)
 - polling, [26](#)
 - serial_print, [29](#)
 - serial_println, [29](#)
 - set_serial_in, [29](#)
 - set_serial_out, [29](#)
- serial_port_in
 - serial.c, [62](#)
- serial_port_out
 - serial.c, [62](#)
- serial_print
 - serial.c, [61](#)
 - serial.h, [29](#)
- serial_println
 - serial.c, [61](#)
 - serial.h, [29](#)
- set_bit
 - paging.c, [73](#)
 - paging.h, [40](#)
- set_serial_in

- serial.c, 62
- serial.h, 29
- set_serial_out
 - serial.c, 62
 - serial.h, 29
- setDate
 - R1commands.c, 93
 - R1commands.h, 100
- setTime
 - R1commands.c, 95
 - R1commands.h, 102
- size
 - header, 12
 - index_entry, 17
- size_t
 - system.h, 46
- sselect
 - idt_entry_struct, 15
 - tables.h, 33
- stack_segment
 - interrupts.c, 56
- sti
 - system.h, 45
- strcat
 - string.c, 76
 - string.h, 41
- strcmp
 - string.c, 76
 - string.h, 42
- strcpy
 - string.c, 76
 - string.h, 42
- string.c
 - atoi, 75
 - isspace, 75
 - memset, 75
 - strcat, 76
 - strcmp, 76
 - strcpy, 76
 - strlen, 77
 - strtok, 77
- string.h
 - atoi, 40
 - isspace, 41
 - memset, 41
 - strcat, 41
 - strcmp, 42
 - strcpy, 42
 - strlen, 42
 - strtok, 42
- strlen
 - string.c, 77
 - string.h, 42
- strtok
 - string.c, 77
 - string.h, 42
- student_free
 - mpx_supt.c, 81
- student_malloc
 - mpx_supt.c, 81
- sys_alloc_mem
 - mpx_supt.c, 79
 - mpx_supt.h, 86
- sys_free_mem
 - mpx_supt.c, 79
 - mpx_supt.h, 86
- sys_req
 - mpx_supt.c, 79
 - mpx_supt.h, 86
- sys_set_free
 - mpx_supt.c, 80
 - mpx_supt.h, 87
- sys_set_malloc
 - mpx_supt.c, 80
 - mpx_supt.h, 87
- system.c
 - klogv, 63
 - kpanic, 63
- system.h
 - asm, 44
 - cli, 44
 - GDT_CS_ID, 44
 - GDT_DS_ID, 44
 - hlt, 45
 - iret, 45
 - klogv, 47
 - kpanic, 47
 - no_warn, 45
 - nop, 45
 - NULL, 45
 - size_t, 46
 - sti, 45
 - u16int, 46
 - u32int, 46
 - u8int, 46
 - volatile, 46
- table
 - index_table, 17
- TABLE_SIZE
 - heap.h, 35
- tables
 - page_dir, 18
- tables.c
 - gdt_entries, 66
 - gdt_init_entry, 64
 - gdt_ptr, 66
 - idt_entries, 66
 - idt_ptr, 66
 - idt_set_gate, 64
 - init_gdt, 65
 - init_idt, 65
 - write_gdt_ptr, 65
 - write_idt_ptr, 65
- tables.h
 - __attribute__, 30
 - access, 32

- base, [32](#)
- base_high, [32](#)
- base_low, [32](#)
- base_mid, [32](#)
- flags, [33](#)
- gdt_init_entry, [31](#)
- idt_set_gate, [31](#)
- init_gdt, [31](#)
- init_idt, [31](#)
- limit, [33](#)
- limit_low, [33](#)
- sselect, [33](#)
- zero, [33](#)
- tables_phys
 - page_dir, [18](#)
- TRUE
 - mpx_supt.h, [85](#)
- u16int
 - system.h, [46](#)
- u32int
 - system.h, [46](#)
- u8int
 - system.h, [46](#)
- usermode
 - page_entry, [20](#)
- version
 - R1commands.c, [97](#)
 - R1commands.h, [104](#)
- volatile
 - system.h, [46](#)
- WRITE
 - mpx_supt.h, [85](#)
- write_gdt_ptr
 - tables.c, [65](#)
- write_idt_ptr
 - tables.c, [65](#)
- writable
 - page_entry, [20](#)
- year
 - date_time, [8](#)
- zero
 - idt_entry_struct, [15](#)
 - tables.h, [33](#)