MAMA

Generated by Doxygen 1.9.3

Chapter 1

MAMA

Check out the `who did what markdown page` for a list of contributions from each user during each milestone.

Use the `user manual` to find information on available commands.

Use the `programmer manual` to find information on individual functions, structs, constants, and other code documentation.

Chapter 2

Who did what table

Update with your contributions every module

	R1	R2	R3	R4	R5	R6

Austin Williams	term/visuals/colorize.c
	
	term/visuals/cursor.c
	
	term/visuals/syntax_highlight.c
	
	term/history.c
	term/syntax.c
	term/args.c
	polling()
	commhand()
					cmd_alias()
					showFree()	

Maximillian Campbell	polling()
	commhand()
	gettime()
	settime()
	getdate()
	setdate()
	cmd_help()
	cmd_shutdown()
	
	itoa()
	Setting up doxygen
	Help pages
		allocatePCB()
		FreePCB()
		SetupPCB()
		CreatePCB()
		UnblockPCB()
		setPriority↩
		PCB()
		MAMA.pdf
		HTML Docs
		Help Pages
			irq.s
			yield()
			loadr3()
			sys_call()
			dispatcher()
			Documentation
			
			Help Pages
				
				createAlarm()
				freeAlarm()
				showAlarms()
				dispatchAlarm()
				
				Documentaion
				
				Help Pages
					initHeap()
					allocateMemory()
					
					freeMemory()
					Documentation	

	R1	R2	R3	R4	R5	R6
Mohammad Alenezi	print_color_code() display_fg_color() display_bg_color() display_reset() display_italicize() print_color_code() cursor_left() cursor_right() cursor_down() cursor_up() cursor_return()	FindPCB() Show block processes DeletePCB() BlockPCB()			Show↩ Allocated() display_clear()	
Abdullah Alqallaf	cmd_version() VersionOs() Some of Help.c comments for Manual	Set Priority() Show All Processes() Show ready processes() Show Block Processes() Show PCB()			isEmpty() freeMemory()	

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Class Documentation

5.1 cmcb_s Struct Reference

Complete Memory Control Block (CMBC)

```
#include <mm.h>
```

Public Attributes

- mcb_state_e type

  The type of the CMCB

- u32int addr

  Beginning address of the CMCB.
- u32int size

  Size of the CMCB.
- char name [32]

  Name of CMCB.
- struct cmcb_s ∗ next

  Next CMCB.
- struct cmcb_s ∗ prev

  Previous CMCB.

5.1.1 Detailed Description

Complete Memory Control Block (CMBC)

5.1.2 Member Data Documentation

5.1.2.1 addr

```
u32int cmcb_s::addr
```

Beginning address of the CMCB.

5.1.2.2 name

```
char cmcb_s::name[32]
```

Name of CMCB.

5.1.2.3 next

```
struct cmcb_s* cmcb_s::next
```

Next CMCB.

5.1.2.4 prev

```
struct cmcb_s* cmcb_s::prev
```

Previous CMCB.

5.1.2.5 size

```
u32int cmcb_s::size
```

Size of the CMCB.

5.1.2.6 type

```
mcb_state_e cmcb_s::type
```

The type of the CMCB

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

- /home/maximillian/Desktop/MAMA/term/memory_management/mm.h

5.2 cmd_mapping Struct Reference

Public Attributes

- char ∗ cmd_name
- cmd_func_t cmd_handler
- char ∗ default_args

5.2.1 Member Data Documentation

5.2.1.1 cmd_handler

```
cmd_func_t cmd_mapping::cmd_handler
```

5.2.1.2 cmd_name

```
char* cmd_mapping::cmd_name
```

5.2.1.3 default_args

```
char* cmd_mapping::default_args
```

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

- /home/maximillian/Desktop/MAMA/term/commhand.c

5.3 context Struct Reference

Context of the currently operating process.

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

Public Attributes

- u32int gs

  Segment registers.
- u32int fs
- u32int es
- u32int ds
- u32int edi

  General purpose registers.
- u32int esi
- u32int ebp
- u32int esp
- u32int ebx
- u32int edx
- u32int ecx
- u32int eax
- u32int eip
- u32int cs
- u32int eflags

5.3.1 Detailed Description

Context of the currently operating process.

5.3.2 Member Data Documentation

5.3.2.1 cs

`u32int context::cs`

5.3.2.2 ds

`u32int context::ds`

5.3.2.3 eax

`u32int context::eax`

5.3.2.4 ebp

`u32int context::ebp`

5.3.2.5 ebx

`u32int context::ebx`

5.3.2.6 ecx

`u32int context::ecx`

5.3.2.7 edi

`u32int context::edi`

General purpose registers.

5.3.2.8 edx

`u32int context::edx`

5.3.2.9 eflags

`u32int context::eflags`

5.3.2.10 eip

`u32int context::eip`

5.3.2.11 es

```
u32int context::es
```

5.3.2.12 esi

```
u32int context::esi
```

5.3.2.13 esp

```
u32int context::esp
```

5.3.2.14 fs

```
u32int context::fs
```

5.3.2.15 gs

```
u32int context::gs
```

Segment registers.

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

- /home/maximillian/Desktop/MAMA/term/dispatch/context.h

5.4 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

5.4.1 Member Data Documentation

5.4.1.1 day_m

`int date_time::day_m`

5.4.1.2 day_w

`int date_time::day_w`

5.4.1.3 day_y

`int date_time::day_y`

5.4.1.4 hour

`int date_time::hour`

5.4.1.5 min

`int date_time::min`

5.4.1.6 mon

`int date_time::mon`

5.4.1.7 sec

`int date_time::sec`

5.4.1.8 year

```
int date_time::year
```

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

- /home/maximillian/Desktop/MAMA/include/system.h

5.5 dcb_t Struct Reference

Public Attributes

- int ∗ eflag_p
- device_ready_state_t ready_state
- device_status_t oper_status
- char ∗ user_read_buf
- int ∗ user_read_count
- char ∗ user_write_buf
- int ∗ user_write_count
- char ring_buffer [RING_BUFFER_SIZE]
- int ring_buffer_head = 0
- int ring_buffer_tail = 0

5.5.1 Member Data Documentation

5.5.1.1 eflag_p

```
int∗ dcb_t::eflag_p
```

5.5.1.2 oper_status

```
device_status_t dcb_t::oper_status
```

5.5.1.3 ready_state

```
device_ready_state_t dcb_t::ready_state
```

5.5.1.4 ring_buffer

```
char dcb_t::ring_buffer[RING_BUFFER_SIZE]
```

5.5.1.5 ring_buffer_head

```
int dcb_t::ring_buffer_head = 0
```

5.5.1.6 ring_buffer_tail

```
int dcb_t::ring_buffer_tail = 0
```

5.5.1.7 user_read_buf

```
char* dcb_t::user_read_buf
```

5.5.1.8 user_read_count

```
int* dcb_t::user_read_count
```

5.5.1.9 user_write_buf

```
char* dcb_t::user_write_buf
```

5.5.1.10 user_write_count

```
int* dcb_t::user_write_count
```

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

- /home/maximillian/Desktop/MAMA/serial_driver/driver.c

5.6 footer Struct Reference

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

Public Attributes

- header head

5.6.1 Member Data Documentation

5.6.1.1 head

```
header footer::head
```

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

- /home/maximillian/Desktop/MAMA/include/mem/heap.h

5.7 gdt_descriptor_struct Struct Reference

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

Public Attributes

- u16int limit
- u32int base

5.7.1 Member Data Documentation

5.7.1.1 base

```
u32int gdt_descriptor_struct::base
```

5.7.1.2 limit

```
u16int gdt_descriptor_struct::limit
```

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

- /home/maximillian/Desktop/MAMA/include/core/tables.h

5.8 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

5.8.1 Member Data Documentation

5.8.1.1 access

```
u8int gdt_entry_struct::access
```

5.8.1.2 base_high

```
u8int gdt_entry_struct::base_high
```

5.8.1.3 base_low

```
u16int gdt_entry_struct::base_low
```

5.8.1.4 base_mid

```
u8int gdt_entry_struct::base_mid
```

5.8.1.5 flags

```
u8int gdt_entry_struct::flags
```

5.8.1.6 limit_low

```
u16int gdt_entry_struct::limit_low
```

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

- /home/maximillian/Desktop/MAMA/include/core/tables.h

5.9 header Struct Reference

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

Public Attributes

- int size
- int index_id

5.9.1 Member Data Documentation

5.9.1.1 index_id

```
int header::index_id
```

5.9.1.2 size

```
int header::size
```

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

- /home/maximillian/Desktop/MAMA/include/mem/heap.h

5.10 heap Struct Reference

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

Public Attributes

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

5.10.1 Member Data Documentation

5.10.1.1 base

```
u32int heap::base
```

5.10.1.2 index

```
index_table heap::index
```

5.10.1.3 max_size

```
u32int heap::max_size
```

5.10.1.4 min_size

```
u32int heap::min_size
```

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

- /home/maximillian/Desktop/MAMA/include/mem/heap.h

5.11 idt_entry_struct Struct Reference

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

Public Attributes

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

5.11.1 Member Data Documentation

5.11.1.1 base_high

```
u16int idt_entry_struct::base_high
```

5.11.1.2 base_low

```
u16int idt_entry_struct::base_low
```

5.11.1.3 flags

```
u8int idt_entry_struct::flags
```

5.11.1.4 sselect

```
u16int idt_entry_struct::sselect
```

5.11.1.5 zero

```
u8int idt_entry_struct::zero
```

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

- /home/maximillian/Desktop/MAMA/include/core/tables.h

5.12 idt_struct Struct Reference

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

Public Attributes

- u16int limit
- u32int base

5.12.1 Member Data Documentation

5.12.1.1 base

```
u32int idt_struct::base
```

5.12.1.2 limit

```
u16int idt_struct::limit
```

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

- /home/maximillian/Desktop/MAMA/include/core/tables.h

5.13 index_entry Struct Reference

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

Public Attributes

- int size
- int empty
- u32int block

5.13.1 Member Data Documentation

5.13.1.1 block

```
u32int index_entry::block
```

5.13.1.2 empty

```
int index_entry::empty
```

5.13.1.3 size

```
int index_entry::size
```

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

- /home/maximillian/Desktop/MAMA/include/mem/heap.h

5.14 index_table Struct Reference

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

Public Attributes

- index_entry table [TABLE_SIZE]
- int id

5.14.1 Member Data Documentation

5.14.1.1 id

```
int index_table::id
```

5.14.1.2 table

```
index_entry index_table::table[TABLE_SIZE]
```

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

- /home/maximillian/Desktop/MAMA/include/mem/heap.h

5.15 mcb_queue_s Struct Reference

"Master" controller of the MCB queue

```
#include <mm.h>
```

Public Attributes

- cmcb_s ∗ mcbq_head

  Head of the MCB queue.
- mcb_state_e mcb_queue_type

  Queue order of the Master controller.

5.15.1 Detailed Description

"Master" controller of the MCB queue

5.15.2 Member Data Documentation

5.15.2.1 mcb_queue_type

```
mcb_state_e mcb_queue_s::mcb_queue_type
```

Queue order of the Master controller.

5.15.2.2 mcbq_head

```
cmcb_s* mcb_queue_s::mcbq_head
```

Head of the MCB queue.

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

- /home/maximillian/Desktop/MAMA/term/memory_management/mm.h

5.16 page_dir Struct Reference

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

Public Attributes

- page_table ∗ tables [1024]
- u32int tables_phys [1024]

5.16.1 Member Data Documentation

5.16.1.1 tables

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

5.16.1.2 tables_phys

```
u32int page_dir::tables_phys[1024]
```

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

- /home/maximillian/Desktop/MAMA/include/mem/paging.h

5.17 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

5.17.1 Member Data Documentation

5.17.1.1 accessed

```
u32int page_entry::accessed
```

5.17.1.2 dirty

```
u32int page_entry::dirty
```

5.17.1.3 frameaddr

```
u32int page_entry::frameaddr
```

5.17.1.4 present

```
u32int page_entry::present
```

5.17.1.5 reserved

```
u32int page_entry::reserved
```

5.17.1.6 usermode

```
u32int page_entry::usermode
```

5.17.1.7 writeable

```
u32int page_entry::writeable
```

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

- /home/maximillian/Desktop/MAMA/include/mem/paging.h

5.18 page_table Struct Reference

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

Public Attributes

- page_entry pages [1024]

5.18.1 Member Data Documentation

5.18.1.1 pages

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

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

- /home/maximillian/Desktop/MAMA/include/mem/paging.h

5.19 param Struct Reference

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

Public Attributes

- int op_code
- int device_id
- char ∗ buffer_ptr
- int ∗ count_ptr

5.19.1 Member Data Documentation

5.19.1.1 buffer_ptr

```
char∗ param::buffer_ptr
```

5.19.1.2 count_ptr

```
int∗ param::count_ptr
```

5.19.1.3 device_id

```
int param::device_id
```

5.19.1.4 op_code

```
int param::op_code
```

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

- /home/maximillian/Desktop/MAMA/modules/mpx_supt.h

5.20 parsed_args Struct Reference

```
#include <args.h>
```

Public Attributes

- int flag_count
- int named_arg_count
- int unnamed_arg_count
- int unnamed_args_used_so_far
- char flags [MAX_CMD_FLAG_COUNT][MAX_CMD_ARG_NAME_LEN+1]
- char named_arg_names [MAX_CMD_NAMED_ARG_COUNT][MAX_CMD_ARG_NAME_LEN+1]
- char named_arg_values [MAX_CMD_NAMED_ARG_COUNT][MAX_CMD_ARG_VALUE_LEN+1]
- char unnamed_args [MAX_CMD_UNNAMED_ARG_COUNT][MAX_CMD_ARG_VALUE_LEN+1]

5.20.1 Member Data Documentation

5.20.1.1 flag_count

```
int parsed_args::flag_count
```

5.20.1.2 flags

```
char parsed_args::flags[MAX_CMD_FLAG_COUNT][MAX_CMD_ARG_NAME_LEN+1]
```

5.20.1.3 named_arg_count

```
int parsed_args::named_arg_count
```

5.20.1.4 named_arg_names

```
char parsed_args::named_arg_names[MAX_CMD_NAMED_ARG_COUNT][MAX_CMD_ARG_NAME_LEN+1]
```

5.20.1.5 named_arg_values

```
char parsed_args::named_arg_values[MAX_CMD_NAMED_ARG_COUNT][MAX_CMD_ARG_VALUE_LEN+1]
```

5.20.1.6 unnamed_arg_count

```
int parsed_args::unnamed_arg_count
```

5.20.1.7 unnamed_args

```
char parsed_args::unnamed_args[MAX_CMD_UNNAMED_ARG_COUNT][MAX_CMD_ARG_VALUE_LEN+1]
```

5.20.1.8 unnamed_args_used_so_far

```
int parsed_args::unnamed_args_used_so_far
```

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

- /home/maximillian/Desktop/MAMA/term/args.h

5.21 pcb_node_t Struct Reference

Individual PCB nodes. Each PCB is associated with one node.

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

Public Attributes

- struct pcb_node_t ∗ pcbn_next_pcb

  Pointer to the Next PCB.
- struct pcb_node_t ∗ pcbn_prev_pcb

  Pointer to the Previous PCB.
- pcb_t ∗ pcb

  Pointer to PCB.

5.21.1 Detailed Description

Individual PCB nodes. Each PCB is associated with one node.

5.21.2 Member Data Documentation

5.21.2.1 pcb

```
pcb_t* pcb_node_t::pcb
```

Pointer to PCB.

5.21.2.2 pcbn_next_pcb

```
struct pcb_node_t* pcb_node_t::pcbn_next_pcb
```

Pointer to the Next PCB.

5.21.2.3 pcbn_prev_pcb

```
struct pcb_node_t* pcb_node_t::pcbn_prev_pcb
```

Pointer to the Previous PCB.

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

- /home/maximillian/Desktop/MAMA/term/pcb/pcb.h

5.22 pcb_queue Struct Reference

"Master" controller of the PCB queue

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

Public Attributes

- int pcbq_count

  Number of PCB's currently in the queue.
- pcb_node_t ∗ pcbq_head

  Head of the PCB queue.
- pcb_node_t ∗ pcbq_tail

  Tail of the PCB queue.
- pcb_queue_order_t queue_order

  Queue order of the Master controller.

5.22.1 Detailed Description

"Master" controller of the PCB queue

5.22.2 Member Data Documentation

5.22.2.1 pcbq_count

```
int pcb_queue::pcbq_count
```

Number of PCB's currently in the queue.

5.22.2.2 pcbq_head

```
pcb_node_t* pcb_queue::pcbq_head
```

Head of the PCB queue.

5.22.2.3 pcbq_tail

```
pcb_node_t* pcb_queue::pcbq_tail
```

Tail of the PCB queue.

5.22.2.4 queue_order

```
pcb_queue_order_t pcb_queue::queue_order
```

Queue order of the Master controller.

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

- /home/maximillian/Desktop/MAMA/term/pcb/pcb.h

5.23 pcb_t Struct Reference

Process Control Block Structure.

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

Public Attributes

- char pcb_name [32]

  PCB Name.
- int pcb_process_class

  Process Class.
- int pcb_priority

  Priority of PCB.
- p_state_t pcb_process_state

  State of the PCB.
- p_protection_mode_t pcb_protection_mode
- unsigned char ∗ pcb_stack_top

  Top of the Stack. Set equal to the stack base + size of the stack.
- unsigned char ∗ pcb_stack_bottom

  Beginning of the Stack.

5.23.1 Detailed Description

Process Control Block Structure.

5.23.2 Member Data Documentation

5.23.2.1 pcb_name

```
char pcb_t::pcb_name[32]
```

PCB Name.

5.23.2.2 pcb_priority

```
int pcb_t::pcb_priority
```

Priority of PCB.

5.23.2.3 pcb_process_class

```
int pcb_t::pcb_process_class
```

Process Class.

5.23.2.4 pcb_process_state

```
p_state_t pcb_t::pcb_process_state
```

State of the PCB.

5.23.2.5 pcb_protection_mode

```
p_protection_mode_t pcb_t::pcb_protection_mode
```

5.23.2.6 pcb_stack_bottom

```
unsigned char* pcb_t::pcb_stack_bottom
```

Beginning of the Stack.

5.23.2.7 pcb_stack_top

```
unsigned char* pcb_t::pcb_stack_top
```

Top of the Stack. Set equal to the stack base + size of the stack.

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

- /home/maximillian/Desktop/MAMA/term/pcb/pcb.h

Chapter 6

File Documentation

6.1 /home/maximillian/Desktop/MAMA/include/core/asm.h File Reference

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

6.2 asm.h

Go to the documentation of this file.

```
#ifndef _ASM_H
#define _ASM_H

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

#endif
```

6.3 /home/maximillian/Desktop/MAMA/include/core/comhand.h File Reference

Functions

- int comhand ()

6.3.1 Function Documentation

6.3.1.1 comhand()

```
int comhand ( )
```

6.4 comhand.h

Go to the documentation of this file.

```
#ifndef _COMHAND_H
#define _COMHAND_H

int comhand();

#endif
```

6.5 /home/maximillian/Desktop/MAMA/include/core/interrupts.h File Reference

Functions

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

6.5.1 Function Documentation

6.5.1.1 init_irq()

```
void init_irq (
            void   )
```

6.5.1.2 init_pic()

```
void init_pic (
            void   )
```

6.6 interrupts.h

Go to the documentation of this file.

```
#ifndef _INTERRUPTS_H
#define _INTERRUPTS_H

/*
  Procedure..: init_irq
  Description..: Installs the initial interrupt handlers for
      the first 32 irq lines. Most do a panic for now.
*/
void init_irq(void);

/*
  Procedure..: init_pic
  Description..: Initializes the programmable interrupt controllers
      and performs the necessary remapping of IRQs. Leaves interrupts
      turned off.
*/
void init_pic(void);

#endif
```

6.7 /home/maximillian/Desktop/MAMA/include/core/io.h File Reference

Macros

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

6.7.1 Macro Definition Documentation

6.7.1.1 inb

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

Value:

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

6.7.1.2 outb

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

6.8 io.h

Go to the documentation of this file.

```
#ifndef _IO_H
#define _IO_H

/*
  Procedure..: outb
  Description..: Write a byte of data to a port.
*/
#define outb(port, data)                          \
  asm volatile ("outb %%al,%%dx" : : "a" (data), "d" (port))

/*
  Procedure..: inb
  Description..: Read a byte of data from a port.
*/
#define inb(port) ({                              \
      unsigned char r;                            \
      asm volatile ("inb %%dx,%%al": "=a" (r): "d" (port)); \
      r;                                   \
    })

#endif
```

6.9 /home/maximillian/Desktop/MAMA/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)

  Serially poll characters from command line.

6.9.1 Macro Definition Documentation

6.9.1.1 COM1

```
#define COM1 0x3f8
```

6.9.1.2 COM2

```
#define COM2 0x2f8
```

6.9.1.3 COM3

```
#define COM3 0x3e8
```

6.9.1.4 COM4

```
#define COM4 0x2e8
```

6.9.2 Function Documentation

6.9.2.1 init_serial()

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

6.9.2.2 polling()

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

Serially poll characters from command line.

Polls input from keyboard and interprets each character individually as it is entered from the keyboard.

Parameters

buffer	Space allocated for single line on the command line
count	Size of the space allocated

Returns

Returns 0 upon success, -1 upon error

6.9.2.3 serial_print()

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

6.9.2.4 serial_println()

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

6.9.2.5 set_serial_in()

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

6.9.2.6 set_serial_out()

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

6.10 serial.h

Go to the documentation of this file.

```
#ifndef _SERIAL_H
#define _SERIAL_H

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

/*
  Procedure..: init_serial
  Description..: Initializes devices for user interaction, logging, ...
*/
int init_serial(int device);

/*
  Procedure..: serial_println
  Description..: Writes a message to the active serial output device.
    Appends a newline character.
*/
int serial_println(const char *msg);

/*
  Procedure..: serial_print
  Description..: Writes a message to the active serial output device.
*/
int serial_print(const char *msg);

/*
  Procedure..: set_serial_out
  Description..: Sets serial_port_out to the given device address.
    All serial output, such as that from serial_println, will be
    directed to this device.
*/
int set_serial_out(int device);

/*
  Procedure..: set_serial_in
  Description..: Sets serial_port_in to the given device address.
    All serial input, such as console input via a virtual machine,
    QEMU/Bochs/etc, will be directed to this device.
*/
int set_serial_in(int device);

/*
   Procedure:  Polling
   Description:   Gathers keyboard input via the serial port using
           the technique of polling
*/

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

#endif
```

6.11 /home/maximillian/Desktop/MAMA/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

6.11.1 Function Documentation

6.11.1.1 __attribute__()

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

6.11.1.2 gdt_init_entry()

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

6.11.1.3 idt_set_gate()

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

6.11.1.4 init_gdt()

```
void init_gdt ( )
```

6.11.1.5 init_idt()

```
void init_idt ( )
```

6.11.2 Variable Documentation

6.11.2.1 access

```
u8int access
```

6.11.2.2 base

```
u32int base
```

6.11.2.3 base_high

```
u8int base_high
```

6.11.2.4 base_low

```
u16int base_low
```

6.11.2.5 base_mid

```
u8int base_mid
```

6.11.2.6 flags

```
u8int flags
```

6.11.2.7 limit

```
u16int limit
```

6.11.2.8 limit_low

```
u16int limit_low
```

6.11.2.9 sselect

```
u16int sselect
```

6.11.2.10 zero

```
u8int zero
```

6.12 tables.h

Go to the documentation of this file.

```
#ifndef _TABLES_H
#define _TABLES_H

#include "system.h"

typedef struct idt_entry_struct
{
  u16int base_low;  //offset bits 0..15
  u16int sselect;   //stack selector in gdt or ldt
  u8int  zero;      //this stays zero; unused
  u8int  flags;     //attributes
  u16int base_high; //offset bits 16..31
}
  __attribute__ ((packed)) idt_entry;

typedef struct idt_struct
{
  u16int limit;
  u32int base;
}
  __attribute__ ((packed)) idt_descriptor;

typedef struct gdt_descriptor_struct
{
  u16int limit;
  u32int base;
}
  __attribute__ ((packed)) gdt_descriptor;

typedef struct gdt_entry_struct
{
  u16int limit_low; //first 16 bits of limit
  u16int base_low;  //first 16 bits of base
  u8int  base_mid;  //bits 16-23 of base
  u8int  access;    //next 8 bits; access flags
  u8int  flags;     //page granularity, size
  u8int  base_high; //last 8 bits of the base
}
  __attribute__ ((packed)) gdt_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();

#endif
```

6.13 /home/maximillian/Desktop/MAMA/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)

6.13.1 Macro Definition Documentation

6.13.1.1 KHEAP_BASE

```
#define KHEAP_BASE 0xD000000
```

6.13.1.2 KHEAP_MIN

```
#define KHEAP_MIN 0x10000
```

6.13.1.3 KHEAP_SIZE

```
#define KHEAP_SIZE 0x1000000
```

6.13.1.4 TABLE_SIZE

```
#define TABLE_SIZE 0x1000
```

6.13.2 Function Documentation

6.13.2.1 _kmalloc()

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

6.13.2.2 alloc()

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

6.13.2.3 init_kheap()

```
void init_kheap ( )
```

6.13.2.4 kfree()

```
u32int kfree ( )
```

6.13.2.5 kmalloc()

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

6.13.2.6 make_heap()

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

6.14 heap.h

Go to the documentation of this file.

```
#ifndef _HEAP_H
#define _HEAP_H

/* Kernel heap */
#define TABLE_SIZE 0x1000
#define KHEAP_BASE 0xD000000
#define KHEAP_MIN  0x10000
#define KHEAP_SIZE 0x1000000

/* Heap allocation header */
typedef struct {
  int size;
  int index_id;
} header;

typedef struct {
  header head;
} footer;

typedef struct {
  int size;
  int empty;
  u32int block;
} index_entry;

/* Kernel heap index table */
typedef struct {
  index_entry table[TABLE_SIZE];
  int id;
} index_table;

/* Heap structure */
typedef struct {
  index_table index;
  u32int base;
  u32int max_size;
  u32int min_size;
} heap;

/*
  Procedure..: _kmalloc
  Description..: Base-level kernel memory allocation routine. Used to
      provide page alignment and access physical addresses of allocations.
      Called by kmalloc with align=0, physical_address=0.
*/
u32int _kmalloc(u32int size, int align, u32int *phys_addr);

/*
  Procedure..: kmalloc
  Description..: Standard kernel memory allocation rountine. Use this unless you
      need to specify alignment or obtain a physical address. Calls _kmalloc.
*/
u32int kmalloc(u32int size);

/*
  Procedure..: kfree
  Description..: Free kernel memory.
*/
u32int kfree();

/*
  Procedure..: init_kheap
  Description..: Initialize the kernel heap, and set it as the current heap.
*/
void init_kheap();

/*
  Procedure..: alloc
  Description..: Allocate some memory using the given heap. Can specify page-alignment.
*/
u32int alloc(u32int size, heap *hp, int align);

/*
  Procedure..: make_heap
  Description..: Create a new heap.
  Parameters..: base - physical start address of the heap
                max  - maximum size the heap may grow to
        min  - minimum/initial size
*/
heap* make_heap(u32int base, u32int max, u32int min);

#endif
```

6.15 /home/maximillian/Desktop/MAMA/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)

6.15.1 Macro Definition Documentation

6.15.1.1 PAGE_SIZE

```
#define PAGE_SIZE 0x1000
```

6.15.2 Function Documentation

6.15.2.1 clear_bit()

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

6.15.2.2 first_free()

```
u32int first_free ( )
```

6.15.2.3 get_bit()

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

6.15.2.4 get_page()

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

6.15.2.5 init_paging()

```
void init_paging ( )
```

6.15.2.6 load_page_dir()

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

6.15.2.7 new_frame()

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

6.15.2.8 set_bit()

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

6.16 paging.h

Go to the documentation of this file.

```
#ifndef _PAGING_H
#define _PAGING_H

#include <system.h>

#define PAGE_SIZE 0x1000

/*
  Page entry structure
   Describes a single page in memory
*/
typedef struct {
  u32int present   : 1;
  u32int writeable : 1;
  u32int usermode  : 1;
  u32int accessed  : 1;
  u32int dirty     : 1;
  u32int reserved  : 7;
  u32int frameaddr : 20;
} page_entry;

/*
  Page table structure
  Contains 1024 pages/frames
*/
typedef struct {
  page_entry pages[1024];
} page_table;

/*
  Page directory structure
  Limited to 1024 tables for now
*/
typedef struct {
  page_table *tables[1024];
  u32int tables_phys[1024];
} page_dir;

/*
  Procedure..: set_bit
  Description..: Marks a page frame bit as in use (1).
*/
void set_bit(u32int addr);

/*
  Procedure..: clear_bit
  Description..: Marks a page frame bit as free (0).
*/
void clear_bit(u32int addr);

/*
  Procedure..: get_bit
  Description..: Checks if page frame is in use.
*/
u32int get_bit(u32int addr);

/*
  Procedure..: first_free
  Description..: Finds the first free page frame.
*/
u32int first_free();

/*
  Procedure..: init_paging
  Description..: Initializes the kernel page directory and
    initial kernel heap area. Performs identity mapping of
    the kernel frames such that the virtual addresses are
    equivalent to the physical addresses.
*/
void init_paging();

/*
  Procedure..: load_page_dir
  Description..: Sets a page directory as the current
    directory and enables paging via the cr0 register.
    The cr3 register enables address translation from
    linear to physical addresses.
    http://en.wikipedia.org/wiki/Control_register#Control_registers_in_x86_series
*/
void load_page_dir(page_dir *new_page_dir);

/*
```

```
  Procedure..: get_page
  Description..: Finds and returns a page, allocating a new
   page table if necessary.
*/
page_entry* get_page(u32int addr, page_dir *dir, int make_table);

/*
  Procedure..: new_frame
  Description..: Marks a frame as in use in the frame bitmap,
    sets up the page, and saves the frame index in the page.
*/
void new_frame(page_entry* page);

#endif
```

6.17 /home/maximillian/Desktop/MAMA/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)
- char ∗ itoa (int i)

    Converts 32-bit integer to an array of 8-bit characters.

6.17.1 Function Documentation

6.17.1.1 atoi()

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

6.17.1.2 isspace()

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

6.17.1.3 itoa()

```
char * itoa (
            int i )
```

Converts 32-bit integer to an array of 8-bit characters.

Converts an integer data type by breaking it down into its individual digits. Digits are stored individually into a character array.

Parameters

i	Integer that will be converted into ascii

Returns

Returns a pointer to the start of the array of character bytes

6.17.1.4 memset()

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

6.17.1.5 strcat()

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

6.17.1.6 strcmp()

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

6.17.1.7 strcpy()

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

6.17.1.8 strlen()

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

6.17.1.9 strtok()

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

6.18 string.h

Go to the documentation of this file.

```
#ifndef _STRING_H
#define _STRING_H

#include <system.h>

/*
 Procedure..: isspace
 Description..: Determine if a character is whitespace.
 Params..: c-character to check
*/
int isspace(const char *c);

/*
  Procedure..: memset
  Description..: Set a region of memory.
  Params..: s-destination, c-byte to write, n-count
*/
void* memset(void *s, int c, size_t n);

/*
  Procedure..: strcpy
  Description..: Copy one string to another.
  Params..: s1-destination, s2-source
*/
char* strcpy(char *s1, const char *s2);

/*
  Procedure..: strcat
  Description..: Concatenate the contents of one string onto another.
  Params..: s1-destination, s2-source
*/
char* strcat(char *s1, const char *s2);

/*
  Procedure..: strlen
  Description..: Returns the length of a string.
  Params..: s-input string
*/
int   strlen(const char *s);

/*
  Procedure..: strcmp
  Description..: String comparison
  Params..: s1-string 1, s2-string 2
*/
int   strcmp(const char *s1, const char *s2);

/*
  Procedure..: strtok
  Description..: Split string into tokens
  Params..: s1-string, s2-delimiter
*/
char* strtok(char *s1, const char *s2);

/*
  Procedure..: atoi
  Description..: Convert an ASCII string to an integer
  Params..: const char *s -- String
*/
int atoi(const char *s);

char *itoa(int i);

#endif
```

6.19 /home/maximillian/Desktop/MAMA/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)

6.19.1 Macro Definition Documentation

6.19.1.1 asm

```
#define asm __asm__
```

6.19.1.2 cli

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

6.19.1.3 GDT_CS_ID

```
#define GDT_CS_ID 0x01
```

6.19.1.4 GDT_DS_ID

```
#define GDT_DS_ID 0x02
```

6.19.1.5 hlt

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

6.19.1.6 iret

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

6.19.1.7 no_warn

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

6.19.1.8 nop

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

6.19.1.9 NULL

```
#define NULL 0
```

6.19.1.10 sti

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

6.19.1.11 volatile

```
#define volatile __volatile__
```

6.19.2 Typedef Documentation

6.19.2.1 size_t

```
typedef unsigned int size_t
```

6.19.2.2 u16int

```
typedef unsigned short u16int
```

6.19.2.3 u32int

```
typedef unsigned long u32int
```

6.19.2.4 u8int

```
typedef unsigned char u8int
```

6.19.3 Function Documentation

6.19.3.1 klogv()

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

6.19.3.2 kpanic()

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

6.20 system.h

Go to the documentation of this file.

```
#ifndef _SYSTEM_H
#define _SYSTEM_H

#define NULL 0

// Suppress 'unused parameter' warnings/errors
#define no_warn(p) if (p) while (1) break

// Allows compilation with gcc -std=c89
// May also help avoid naming conflicts
#define asm __asm__
#define volatile __volatile__

#define sti()  asm volatile ("sti"::)  //turn irqs off
#define cli()  asm volatile ("cli"::)  //turn irqs on
#define nop()  asm volatile ("nop"::)  //skip cycle
#define hlt()  asm volatile ("hlt"::)  //halt
#define iret() asm volatile ("iret"::) //interrupt return

#define GDT_CS_ID 0x01 //kernel code segment ID
#define GDT_DS_ID 0x02 //kernel data segment ID

/* System Types */
typedef unsigned int   size_t;
typedef unsigned char  u8int;
typedef unsigned short u16int;
typedef unsigned long  u32int;

/* Time */
typedef struct {
  int sec;
  int min;
  int hour;
  int day_w;
  int day_m;
  int day_y;
  int mon;
  int year;
} date_time;

/* Test if interrupts are on */
static inline int irq_on()
{
  int f;
  asm volatile ("pushf\n\t"
        "popl %0"
        : "=g"(f));
  return f & (1 « 9);
}

void klogv(const char *msg);
void kpanic(const char *msg);

#endif
```

6.21 /home/maximillian/Desktop/MAMA/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 sys_call_isr ()
- void isr0 ()
- void do_isr ()
- void init_irq (void)
- void init_pic (void)
- void do_divide_error ()
- void do_debug ()
- void do_nmi ()
- void do_breakpoint ()
- void do_overflow ()
- void do_bounds ()
- void do_invalid_op ()
- void do_device_not_available ()
- void do_double_fault ()
- void do_coprocessor_segment ()
- void do_invalid_tss ()
- void do_segment_not_present ()
- void do_stack_segment ()
- void do_general_protection ()
- void do_page_fault ()
- void do_reserved ()
- void do_coprocessor ()

Variables

- idt_entry idt_entries [256]

6.21.1 Macro Definition Documentation

6.21.1.1 ICW1

```
#define ICW1 0x11
```

6.21.1.2 ICW4

```
#define ICW4 0x01
```

6.21.1.3 io_wait

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

6.21.1.4 PIC1

```
#define PIC1 0x20
```

6.21.1.5 PIC2

```
#define PIC2 0xA0
```

6.21.2 Function Documentation

6.21.2.1 bounds()

```
void bounds ( )
```

6.21.2.2 breakpoint()

```
void breakpoint ( )
```

6.21.2.3 coprocessor()

```
void coprocessor ( )
```

6.21.2.4 coprocessor_segment()

```
void coprocessor_segment ( )
```

6.21.2.5 debug()

```
void debug ( )
```

6.21.2.6 device_not_available()

```
void device_not_available ( )
```

6.21.2.7 divide_error()

```
void divide_error ( )
```

6.21.2.8 do_bounds()

```
void do_bounds ( )
```

6.21.2.9 do_breakpoint()

```
void do_breakpoint ( )
```

6.21.2.10 do_coprocessor()

```
void do_coprocessor ( )
```

6.21.2.11 do_coprocessor_segment()

```
void do_coprocessor_segment ( )
```

6.21.2.12 do_debug()

```
void do_debug ( )
```

6.21.2.13 do_device_not_available()

```
void do_device_not_available ( )
```

6.21.2.14 do_divide_error()

```
void do_divide_error ( )
```

6.21.2.15 do_double_fault()

```
void do_double_fault ( )
```

6.21.2.16 do_general_protection()

```
void do_general_protection ( )
```

6.21.2.17 do_invalid_op()

```
void do_invalid_op ( )
```

6.21.2.18 do_invalid_tss()

```
void do_invalid_tss ( )
```

6.21.2.19 do_isr()

```
void do_isr ( )
```

6.21.2.20 do_nmi()

```
void do_nmi ( )
```

6.21.2.21 do_overflow()

```
void do_overflow ( )
```

6.21.2.22 do_page_fault()

```
void do_page_fault ( )
```

6.21.2.23 do_reserved()

```
void do_reserved ( )
```

6.21.2.24 do_segment_not_present()

```
void do_segment_not_present ( )
```

6.21.2.25 do_stack_segment()

```
void do_stack_segment ( )
```

6.21.2.26 double_fault()

```
void double_fault ( )
```

6.21.2.27 general_protection()

```
void general_protection ( )
```

6.21.2.28 init_irq()

```
void init_irq (
             void  )
```

6.21.2.29 init_pic()

```
void init_pic (
             void  )
```

6.21.2.30 invalid_op()

```
void invalid_op ( )
```

6.21.2.31 invalid_tss()

```
void invalid_tss ( )
```

6.21.2.32 isr0()

```
void isr0 ( )
```

6.21.2.33 nmi()

```
void nmi ( )
```

6.21.2.34 overflow()

```
void overflow ( )
```

6.21.2.35 page_fault()

```
void page_fault ( )
```

6.21.2.36 reserved()

```
void reserved ( )
```

6.21.2.37 rtc_isr()

```
void rtc_isr ( )
```

6.21.2.38 segment_not_present()

```
void segment_not_present ( )
```

6.21.2.39 stack_segment()

```
void stack_segment ( )
```

6.21.2.40 sys_call_isr()

```
void sys_call_isr ( )
```

6.21.3 Variable Documentation

6.21.3.1 idt_entries

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

6.22 /home/maximillian/Desktop/MAMA/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 "term/commhand.c"
#include "term/dispatch/context.h"
#include "term/pcb/pcb.h"
#include "term/dnt/dnt.h"
#include "term/memory_management/mm.h"
```

Functions

- void kmain (void)

6.22.1 Function Documentation

6.22.1.1 kmain()

```
void kmain (
          void  )
```

6.23 /home/maximillian/Desktop/MAMA/kernel/core/serial.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <core/io.h>
#include <core/serial.h>
#include <term/history.h>
#include <term/visuals/syntax_highlight.h>
#include <term/visuals/syntax_highlight.c>
```

Macros

- #define NO_ERROR 0
- #define DELETE 0b00001
- #define LEFT_ARROW 0b00010
- #define RIGHT_ARROW 0b00100
- #define UP_ARROW 0b01000
- #define DOWN_ARROW 0b10000

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)
- unsigned int consume_special ()
- int ∗ polling (char ∗buffer, int ∗count)

    Serially poll characters from command line.

Variables

- int serial_port_out = 0
- int serial_port_in = 0

6.23.1 Macro Definition Documentation

6.23.1.1 DELETE

```
#define DELETE 0b00001
```

6.23.1.2 DOWN_ARROW

```
#define DOWN_ARROW 0b10000
```

6.23.1.3 LEFT_ARROW

```
#define LEFT_ARROW 0b00010
```

6.23.1.4 NO_ERROR

```
#define NO_ERROR 0
```

6.23.1.5 RIGHT_ARROW

```
#define RIGHT_ARROW 0b00100
```

6.23.1.6 UP_ARROW

```
#define UP_ARROW 0b01000
```

6.23.2 Function Documentation

6.23.2.1 consume_special()

```
unsigned int consume_special ( )
```

6.23.2.2 init_serial()

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

6.23.2.3 polling()

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

Serially poll characters from command line.

Polls input from keyboard and interprets each character individually as it is entered from the keyboard.

Parameters

buffer	Space allocated for single line on the command line
count	Size of the space allocated

Returns

Returns 0 upon success, -1 upon error

6.23.2.4 serial_print()

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

6.23.2.5 serial_println()

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

6.23.2.6 set_serial_in()

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

6.23.2.7 set_serial_out()

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

6.23.3 Variable Documentation

6.23.3.1 serial_port_in

```
int serial_port_in = 0
```

6.23.3.2 serial_port_out

```
int serial_port_out = 0
```

6.24 /home/maximillian/Desktop/MAMA/kernel/core/system.c File Reference

```
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include <modules/mpx_supt.h>
#include "term/pcb/pcb.h"
#include "term/dispatch/context.h"
#include <lib/out.h>
```

Functions

- void klogv (const char ∗msg)
- void kpanic (const char ∗msg)
- u32int ∗ sys_call (context ∗registers)

  Called to start interrupt.

Variables

- pcb_t ∗ cop

  Currently operating process.
- context ∗ global_context

  Context.
- pcb_queue_t ∗ priority_queue
- param params

6.24.1 Function Documentation

6.24.1.1 klogv()

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

6.24.1.2 kpanic()

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

6.24.1.3 sys_call()

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

Called to start interrupt.

Is called by irq to determine the next routine to load

Parameters

registers	Context registers for the current process

Returns

Pointer to the process being loaded

6.24.2 Variable Documentation

6.24.2.1 cop

```
pcb_t* cop
```

Currently operating process.

6.24.2.2 global_context

```
context* global_context
```

Context.

6.24.2.3 params

```
param params  [extern]
```

6.24.2.4 priority_queue

```
pcb_queue_t* priority_queue  [extern]
```

6.25 /home/maximillian/Desktop/MAMA/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]

6.25.1 Function Documentation

6.25.1.1 gdt_init_entry()

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

6.25.1.2 idt_set_gate()

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

6.25.1.3 init_gdt()

```
void init_gdt ( )
```

6.25.1.4 init_idt()

```
void init_idt ( )
```

6.25.1.5 write_gdt_ptr()

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

6.25.1.6 write_idt_ptr()

```
void write_idt_ptr (
            u32int  )
```

6.25.2 Variable Documentation

6.25.2.1 gdt_entries

```
gdt_entry gdt_entries[5]
```

6.25.2.2 gdt_ptr

```
gdt_descriptor gdt_ptr
```

6.25.2.3 idt_entries

```
idt_entry idt_entries[256]
```

6.25.2.4 idt_ptr

```
idt_descriptor idt_ptr
```

6.26 /home/maximillian/Desktop/MAMA/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

6.26.1 Function Documentation

6.26.1.1 _kmalloc()

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

6.26.1.2 alloc()

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

6.26.1.3 kmalloc()

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

6.26.1.4 make_heap()

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

6.26.2 Variable Documentation

6.26.2.1 __end

```
void __end
```

6.26.2.2 _end

```
void _end
```

6.26.2.3 curr_heap

```
heap* curr_heap = 0
```

6.26.2.4 end

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

6.26.2.5 kdir

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

6.26.2.6 kheap

```
heap* kheap = 0
```

6.26.2.7 phys_alloc_addr

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

6.27 /home/maximillian/Desktop/MAMA/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

6.27.1 Function Documentation

6.27.1.1 clear_bit()

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

6.27.1.2 find_free()

```
u32int find_free ( )
```

6.27.1.3 get_bit()

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

6.27.1.4 get_page()

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

6.27.1.5 init_paging()

```
void init_paging ( )
```

6.27.1.6 load_page_dir()

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

6.27.1.7 new_frame()

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

6.27.1.8 set_bit()

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

6.27.2 Variable Documentation

6.27.2.1 cdir

```
page_dir* cdir = 0
```

6.27.2.2 frames

```
u32int* frames
```

6.27.2.3 kdir

```
page_dir* kdir = 0
```

6.27.2.4 kheap

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

6.27.2.5 mem_size

```
u32int mem_size = 0x4000000
```

6.27.2.6 nframes

```
u32int nframes
```

6.27.2.7 page_size

```
u32int page_size = 0x1000
```

6.27.2.8 phys_alloc_addr

```
u32int phys_alloc_addr  [extern]
```

6.28 /home/maximillian/Desktop/MAMA/lib/out.c File Reference

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

Functions

- int print (char ∗str, int len)
- int printc (char c)
- int println (char ∗str, int len)
- void printf (char ∗str,...)
- int read (char ∗buf, int len)

6.28.1 Function Documentation

6.28.1.1 print()

```
int print (
            char * str,
            int len )
```

6.28.1.2 printc()

```
int printc (
            char c )
```

6.28.1.3 printf()

```
void printf (
            char * str,
             ... )
```

6.28.1.4 println()

```
int println (
            char * str,
            int len )
```

6.28.1.5 read()

```
int read (
            char * buf,
            int len )
```

6.29 /home/maximillian/Desktop/MAMA/lib/out.h File Reference

Functions

- int cmd_help (char ∗command)

  Prints help message for command.
- void gettimeHelp ()

  Help page for gettime() method.
- void settimeHelp ()

  Help page for settime() method.
- void getdateHelp ()

  Help page for the getdate() method.
- void setdateHelp ()

  Help page for the setdate() method.
- void helpHelp ()

  Help page for the help command.
- void shutdownHelp ()

  Help page for the shutdown command.
- void helpList ()

Displays a list of common system commands.
- void versionHelp ()

  Help page for the version command.
- void createpcbHelp ()

  Help page for createpcb.
- void deletepcbHelp ()

  Help page for deletepcb.
- void showpcbHelp ()

  Help page for showpcb.
- void showallpcbHelp ()

  Help page for showallpcb.
- void showreadypcbHelp ()

  Help page for showreadypcb.
- void showblockedpcbHelp ()

  Help page for showblockedpcb.
- void blockHelp ()

  Help page for block.
- void unblockHelp ()

  Help page for unblock.
- void setpriorityHelp ()

  Help page for setpriority.
- void resumeHelp ()

  Help page for resume.
- void suspendHelp ()

  Help page for suspend.
- void loadr3Help ()

  Help page for loadr3.
- void setalarmHelp ()

  Help page for setalarm.
- void showalarmsHelp ()

  Help page for showalarm.
- void freealarmHelp ()

  Help page for freealarm.
- void resumeallHelp ()

  Help page for resumeallpcb.
- void showallocHelp ()

  Help page for showalloc.
- void showfreeHelp ()

  Help page for showfree.
- void isemptyHelp ()

  Help page for isempty.
- void clearHelp ()

  Help page for clear.
- void aliasHelp ()

  Help page for alias.
- int print (char ∗, int)
- int printc (char)
- int println (char ∗, int)
- void printf (char ∗,...)
- int read (char ∗, int)

6.29.1 Function Documentation

6.29.1.1 aliasHelp()

```
void aliasHelp ( )
```

Help page for alias.

Displays the alias help pages

6.29.1.2 blockHelp()

```
void blockHelp ( )
```

Help page for block.

Displays the block help page

6.29.1.3 clearHelp()

```
void clearHelp ( )
```

Help page for clear.

Displays the clear help pages

6.29.1.4 cmd_help()

```
int cmd_help (
             char * command )
```

Prints help message for command.

Prints out a help message and basic syntax for a specific command

Parameters

command	Command which the user needs basic information and syntax for

Returns

1 upon success, -1 upon error

6.29.1.5 createpcbHelp()

```
void createpcbHelp ( )
```

Help page for createpcb.

Displays the createpcb help page

6.29.1.6 deletepcbHelp()

```
void deletepcbHelp ( )
```

Help page for deletepcb.

Displays the deletepcb help page

6.29.1.7 freealarmHelp()

```
void freealarmHelp ( )
```

Help page for freealarm.

Displays the freealarm help page

6.29.1.8 getdateHelp()

```
void getdateHelp ( )
```

Help page for the getdate() method.

Prints out the name, usage, return and description for the getdate() method.

6.29.1.9 gettimeHelp()

```
void gettimeHelp ( )
```

Help page for gettime() method.

Prints out the name, usage, return and description for the gettime() method.

6.29.1.10 helpHelp()

```
void helpHelp ( )
```

Help page for the help command.

Prints out the name, usage, return and description for the help command.

6.29.1.11 helpList()

```
void helpList ( )
```

Displays a list of common system commands.

Displays a list of common system commands for the user.

6.29.1.12 isemptyHelp()

```
void isemptyHelp ( )
```

Help page for isempty.

Displays the isempty help pages

6.29.1.13 loadr3Help()

```
void loadr3Help ( )
```

Help page for loadr3.

Displays the loadr3 help page

6.29.1.14 print()

```
int print (
            char * str,
            int len )
```

6.29.1.15 printc()

```
int printc (
            char c )
```

6.29.1.16 printf()

```
void printf (
            char * str,
             ...  )
```

6.29.1.17 println()

```
int println (
            char * str,
            int len )
```

6.29.1.18 read()

```
int read (
            char * buf,
            int len )
```

6.29.1.19 resumeallHelp()

```
void resumeallHelp ( )
```

Help page for resumeallpcb.

Displays the resumeallpcb help page

6.29.1.20 resumeHelp()

```
void resumeHelp ( )
```

Help page for resume.

Displays the resume help page

6.29.1.21 setalarmHelp()

```
void setalarmHelp ( )
```

Help page for setalarm.

Displays the setalarm help page

6.29.1.22 setdateHelp()

```
void setdateHelp ( )
```

Help page for the setdate() method.

Prints out the name, usage, and description for the setdate() method.

6.29.1.23 setpriorityHelp()

```
void setpriorityHelp ( )
```

Help page for setpriority.

Displays the setpriority help page

6.29.1.24 settimeHelp()

```
void settimeHelp ( )
```

Help page for settime() method.

Prints out the name, usage, and description for the settime() method.

6.29.1.25 showalarmsHelp()

```
void showalarmsHelp ( )
```

Help page for showalarm.

Displays the showalarm help page

6.29.1.26 showallocHelp()

```
void showallocHelp ( )
```

Help page for showalloc.

Displays the showalloc help page

6.29.1.27 showallpcbHelp()

```
void showallpcbHelp ( )
```

Help page for showallpcb.

Displays the showallpcb help page

6.29.1.28 showblockedpcbHelp()

```
void showblockedpcbHelp ( )
```

Help page for showblockedpcb.

Displays the showblockedpcb help page

6.29.1.29 showfreeHelp()

```
void showfreeHelp ( )
```

Help page for showfree.

Displays the showfree help pages

6.29.1.30 showpcbHelp()

```
void showpcbHelp ( )
```

Help page for showpcb.

Displays the showpcb help page

6.29.1.31 showreadypcbHelp()

```
void showreadypcbHelp ( )
```

Help page for showreadypcb.

Displays the showreadypcb help page

6.29.1.32 shutdownHelp()

```
void shutdownHelp ( )
```

Help page for the shutdown command.

Prints out the name, usage, and description for the shutdown system command.

6.29.1.33 suspendHelp()

```
void suspendHelp ( )
```

Help page for suspend.

Displays the suspend help page

6.29.1.34 unblockHelp()

```
void unblockHelp ( )
```

Help page for unblock.

Displays te unblock help page

6.29.1.35 versionHelp()

```
void versionHelp ( )
```

Help page for the version command.

Displays the current verson of the system.

6.30 out.h

Go to the documentation of this file.

```
#ifndef OUT_H
#define OUT_H

int cmd_help(char * command);

void gettimeHelp();

void settimeHelp();

void getdateHelp();

void setdateHelp();

void helpHelp();

void shutdownHelp();

void helpList();

void versionHelp();

void createpcbHelp();

void deletepcbHelp();

void showpcbHelp();

void showallpcbHelp();

void showreadypcbHelp();

void showblockedpcbHelp();

void blockHelp();

void unblockHelp();

void setpriorityHelp();

void resumeHelp();

void suspendHelp();

void loadr3Help();

void setalarmHelp();

void showalarmsHelp();

void freealarmHelp();

void resumeallHelp();

void showallocHelp();

void showfreeHelp();

void isemptyHelp();

void clearHelp();

void aliasHelp();


int print(char *, int);
int printc(char);
int println(char *, int);
void printf(char *, ...);
int read(char *, int);

#endif
```

6.31 /home/maximillian/Desktop/MAMA/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)
- char * itoa (int value)

    Converts 32-bit integer to an array of 8-bit characters.
- 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)

6.31.1 Function Documentation

6.31.1.1 atoi()

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

6.31.1.2 isspace()

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

6.31.1.3 itoa()

```
char * itoa (
            int i )
```

Converts 32-bit integer to an array of 8-bit characters.

Converts an integer data type by breaking it down into its individual digits. Digits are stored individually into a character array.

Parameters

i	Integer that will be converted into ascii

Returns

Returns a pointer to the start of the array of character bytes

6.31.1.4 memset()

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

6.31.1.5 strcat()

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

6.31.1.6 strcmp()

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

6.31.1.7 strcpy()

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

6.31.1.8 strlen()

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

6.31.1.9 strtok()

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

6.32 /home/maximillian/Desktop/MAMA/modules/mpx_supt.c File Reference

```
#include "mpx_supt.h"
#include <mem/heap.h>
#include <string.h>
#include <core/serial.h>
#include "../lib/out.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 *)

6.32.1 Function Documentation

6.32.1.1 idle()

```
void idle ( )
```

6.32.1.2 mpx_init()

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

6.32.1.3 sys_alloc_mem()

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

6.32.1.4 sys_free_mem()

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

6.32.1.5 sys_req()

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

6.32.1.6 sys_set_free()

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

6.32.1.7 sys_set_malloc()

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

6.32.2 Variable Documentation

6.32.2.1 current_module

```
int current_module = -1
```

6.32.2.2 params

```
param params
```

6.32.2.3 student_free

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

6.32.2.4 student_malloc

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

6.33 /home/maximillian/Desktop/MAMA/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 ()

6.33.1 Macro Definition Documentation

6.33.1.1 COM_PORT

```
#define COM_PORT 222
```

6.33.1.2 DEFAULT_DEVICE

```
#define DEFAULT_DEVICE 111
```

6.33.1.3 EXIT

```
#define EXIT 0
```

6.33.1.4 FALSE

```
#define FALSE 0
```

6.33.1.5 IDLE

```
#define IDLE 1
```

6.33.1.6 INVALID_BUFFER

```
#define INVALID_BUFFER 1000
```

6.33.1.7 INVALID_COUNT

```
#define INVALID_COUNT 2000
```

6.33.1.8 INVALID_OPERATION

```
#define INVALID_OPERATION 4
```

6.33.1.9 IO_MODULE

```
#define IO_MODULE 10
```

6.33.1.10 MEM_MODULE

```
#define MEM_MODULE 11
```

6.33.1.11 MODULE_F

```
#define MODULE_F 9
```

6.33.1.12 MODULE_R1

```
#define MODULE_R1 0
```

6.33.1.13 MODULE_R2

```
#define MODULE_R2 1
```

6.33.1.14 MODULE_R3

```
#define MODULE_R3 2
```

6.33.1.15 MODULE_R4

```
#define MODULE_R4 4
```

6.33.1.16 MODULE_R5

```
#define MODULE_R5 8
```

6.33.1.17 READ

```
#define READ 2
```

6.33.1.18 TRUE

```
#define TRUE 1
```

6.33.1.19 WRITE

```
#define WRITE 3
```

6.33.2 Function Documentation

6.33.2.1 idle()

```
void idle ( )
```

6.33.2.2 mpx_init()

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

6.33.2.3 sys_alloc_mem()

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

6.33.2.4 sys_free_mem()

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

6.33.2.5 sys_req()

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

6.33.2.6 sys_set_free()

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

6.33.2.7 sys_set_malloc()

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

6.34 mpx_supt.h

Go to the documentation of this file.

```
#ifndef _MPX_SUPT_H
#define _MPX_SUPT_H

#include <system.h>

#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

// error codes
#define INVALID_BUFFER 1000
#define INVALID_COUNT 2000

#define DEFAULT_DEVICE 111
#define COM_PORT 222

typedef struct {
  int op_code;
  int device_id;
  char *buffer_ptr;
  int *count_ptr;
} param;

/*
  Procedure..: sys_req
  Description..: Generate interrupt 60H
  Params..: int op_code one of (IDLE, EXIT, READ, WRITE)
*/
int sys_req( int op_code, int device_id, char *buffer_ptr,
            int *count_ptr );

/*
  Procedure..: mpx_init
  Description..: Initialize MPX support software
  Params..: int cur_mod (symbolic constants MODULE_R1, MODULE_R2, etc
*/
void mpx_init(int cur_mod);

/*
  Procedure..: sys_set_malloc
  Description..: Sets the memory allocation function for sys_alloc_mem
  Params..: Function pointer
*/
void sys_set_malloc(u32int (*func)(u32int));

/*
  Procedure..: sys_set_free
  Description..: Sets the memory free function for sys_free_mem
  Params..: s1-destination, s2-source
*/
void sys_set_free(int (*func)(void *));

/*
  Procedure..: sys_alloc_mem
  Description..: Allocates a block of memory (similar to malloc)
  Params..: Number of bytes to allocate
*/
void *sys_alloc_mem(u32int size);

/*
  Procedure..: sys_free_mem
  Description..: Frees memory
  Params..: Pointer to block of memory to free
*/
int sys_free_mem(void *ptr);

/*
  Procedure..: idle
  Description..: The idle process
  Params..: None
*/
void idle();

#endif
```

6.35 /home/maximillian/Desktop/MAMA/README.md File Reference

6.36 /home/maximillian/Desktop/MAMA/serial_driver/driver.c File Reference

Classes

- struct dcb_t

Macros

- #define BASE COM1
- #define DIVISOR_LATCH_LOW_BYTE_REGISTER 0
- #define DIVISOR_LATCH_HIGH_BYTE_REGISTER 1
- #define INTERRUPT_ENABLE_REGISTER 1
- #define INTERRUPT_IDENTIFICATION_REGISTER 2
- #define LINE_CONTROL_REGISTER 3
- #define MODEM_CONTROL_REGISTER 4
- #define LINE_STATUS_REGISTER 5
- #define MODEM_STATUS_REGISTER 6
- #define SCRATCH_REGISTER 7
- #define PIC_MASK 0x21;
- #define RING_BUFFER_SIZE 100

Typedefs

- typedef struct dcb_t dcb_t

Enumerations

- enum device_ready_state_t { OPEN , CLOSED }
- enum device_status_t { IDLE , READING , WRITING }

Functions

- int com_open (int ∗eflag_p, int baud_rate)
- int com_close ()
- int com_read (char ∗buf, int ∗count)
- int com_write (char ∗buf, int ∗count)

Variables

- const dcb_t ∗ COM1_control_block = NULL

6.36.1 Macro Definition Documentation

6.36.1.1 BASE

```
#define BASE COM1
```

6.36.1.2 DIVISOR_LATCH_HIGH_BYTE_REGISTER

```
#define DIVISOR_LATCH_HIGH_BYTE_REGISTER 1
```

6.36.1.3 DIVISOR_LATCH_LOW_BYTE_REGISTER

```
#define DIVISOR_LATCH_LOW_BYTE_REGISTER 0
```

6.36.1.4 INTERRUPT_ENABLE_REGISTER

```
#define INTERRUPT_ENABLE_REGISTER 1
```

6.36.1.5 INTERRUPT_IDENTIFICATION_REGISTER

```
#define INTERRUPT_IDENTIFICATION_REGISTER 2
```

6.36.1.6 LINE_CONTROL_REGISTER

```
#define LINE_CONTROL_REGISTER 3
```

6.36.1.7 LINE_STATUS_REGISTER

```
#define LINE_STATUS_REGISTER 5
```

6.36.1.8 MODEM_CONTROL_REGISTER

```
#define MODEM_CONTROL_REGISTER 4
```

6.36.1.9 MODEM_STATUS_REGISTER

```
#define MODEM_STATUS_REGISTER 6
```

6.36.1.10 PIC_MASK

```
#define PIC_MASK 0x21;
```

6.36.1.11 RING_BUFFER_SIZE

```
#define RING_BUFFER_SIZE 100
```

6.36.1.12 SCRATCH_REGISTER

```
#define SCRATCH_REGISTER 7
```

6.36.2 Typedef Documentation

6.36.2.1 dcb_t

```
typedef struct dcb_t dcb_t
```

6.36.3 Enumeration Type Documentation

6.36.3.1 device_ready_state_t

```
enum device_ready_state_t
```

Enumerator

OPEN	
CLOSED	

6.36.3.2 device_status_t

```
enum device_status_t
```

Enumerator

IDLE	
READING	
WRITING	

6.36.4 Function Documentation

6.36.4.1 com_close()

```
int com_close ( )
```

6.36.4.2 com_open()

```
int com_open (
             int * eflag_p,
             int baud_rate )
```

6.36.4.3 com_read()

```
int com_read (
             char * buf,
             int * count )
```

6.36.4.4 com_write()

```
int com_write (
             char * buf,
             int * count )
```

6.36.5 Variable Documentation

6.36.5.1 COM1_control_block

```
const dcb_t* COM1_control_block = NULL
```

6.37 /home/maximillian/Desktop/MAMA/term/args.c File Reference

```
#include "commhand.h"
#include "utils.h"
#include "args.h"
#include "syntax.h"
#include <lib/out.h>
#include <include/string.h>
```

Macros

- #define MAX_PARSE_STACK_SIZE 2

Functions

- int get_token (char ∗∗, char ∗, int)
- int stack_empty ()
- enum SyntaxState stack_peek ()
- void stack_push (enum SyntaxState)
- void stack_pop ()
- parsed_args ∗ parse_args (char ∗arg_str)
- int named_arg (parsed_args ∗args, char ∗arg_name, char ∗∗arg_val)
- int flag (parsed_args ∗args, char ∗flag_name)
- int next_unnamed_arg (parsed_args ∗args, char ∗∗arg_val)

Variables

- enum SyntaxState parse_stack [MAX_PARSE_STACK_SIZE]
- int stack_size = 0
- enum SyntaxState last_state
- enum SyntaxState cur_state

6.37.1 Macro Definition Documentation

6.37.1.1 MAX_PARSE_STACK_SIZE

```
#define MAX_PARSE_STACK_SIZE 2
```

6.37.2 Function Documentation

6.37.2.1 flag()

```
int flag (
             parsed_args * args,
             char * flag_name )
```

6.37.2.2 get_token()

```
int get_token (
             char ** arg_str,
             char * token,
             int max_token_len )
```

6.37.2.3 named_arg()

```
int named_arg (
             parsed_args * args,
             char * arg_name,
             char ** arg_val )
```

6.37.2.4 next_unnamed_arg()

```
int next_unnamed_arg (
             parsed_args * args,
             char ** arg_val )
```

6.37.2.5 parse_args()

```
parsed_args * parse_args (
             char * arg_str )
```

6.37.2.6 stack_empty()

```
int stack_empty ( )
```

6.37.2.7 stack_peek()

```
enum SyntaxState stack_peek ( )
```

6.37.2.8 stack_pop()

```
void stack_pop ( )
```

6.37.2.9 stack_push()

```
void stack_push (
             enum SyntaxState state )
```

6.37.3 Variable Documentation

6.37.3.1 cur_state

```
enum SyntaxState cur_state
```

6.37.3.2 last_state

```
enum SyntaxState last_state
```

6.37.3.3 parse_stack

```
enum SyntaxState parse_stack[MAX_PARSE_STACK_SIZE]
```

6.37.3.4 stack_size

```
int stack_size = 0
```

6.38 /home/maximillian/Desktop/MAMA/term/args.h File Reference

Classes

- struct parsed_args

Typedefs

- typedef struct parsed_args parsed_args

Functions

- parsed_args ∗ parse_args (char ∗)

6.38.1 Typedef Documentation

6.38.1.1 parsed_args

```
typedef struct parsed_args parsed_args
```

6.38.2 Function Documentation

6.38.2.1 parse_args()

```
parsed_args * parse_args (
                char * arg_str )
```

6.39 args.h

Go to the documentation of this file.

```
#ifndef ARGS_H
#define ARGS_H

typedef struct parsed_args {
    int flag_count;
    int named_arg_count;
    int unnamed_arg_count;
    int unnamed_args_used_so_far;

    char flags[MAX_CMD_FLAG_COUNT][MAX_CMD_ARG_NAME_LEN + 1];
    char named_arg_names[MAX_CMD_NAMED_ARG_COUNT][MAX_CMD_ARG_NAME_LEN + 1];
    char named_arg_values[MAX_CMD_NAMED_ARG_COUNT][MAX_CMD_ARG_VALUE_LEN + 1];
    char unnamed_args[MAX_CMD_UNNAMED_ARG_COUNT][MAX_CMD_ARG_VALUE_LEN + 1];
} parsed_args;

parsed_args *parse_args(char *);

#endif
```

6.40 /home/maximillian/Desktop/MAMA/term/ascii/mama.c File Reference

```
#include "mama.h"
#include "term/dnt/dnt.h"
```

Functions

- void mama ()

  mama ascii art

6.40.1 Function Documentation

6.40.1.1 mama()

```
void mama ( )
```

mama ascii art

One of the intro ascii art.

6.41 /home/maximillian/Desktop/MAMA/term/ascii/mama.h File Reference

Functions

- void mama ()

  mama ascii art

6.41.1 Function Documentation

6.41.1.1 mama()

```
void mama ( )
```

mama ascii art

One of the intro ascii art.

6.42 mama.h

Go to the documentation of this file.

```
void mama();
```

6.43 /home/maximillian/Desktop/MAMA/term/cmds/argtest.c File Reference

```
#include "../args.h"
#include "../args.c"
#include <lib/out.h>
```

Functions

- int cmd_argtest (char *arg_str)

6.43.1 Function Documentation

6.43.1.1 cmd_argtest()

```
int cmd_argtest (
            char * arg_str )
```

6.44 /home/maximillian/Desktop/MAMA/term/cmds/echo.c File Reference

```
#include <lib/out.h>
```

Functions

- int cmd_echo (char ∗arg_str)

6.44.1 Function Documentation

6.44.1.1 cmd_echo()

```
int cmd_echo (
            char ∗ arg_str )
```

6.45 /home/maximillian/Desktop/MAMA/help.c File Reference

```
#include <lib/out.h>
```

Functions

- int cmd_help (char ∗command)

  Prints help message for command.
- void helpList ()

  Displays a list of common system commands.
- void shutdownHelp ()

  Help page for the shutdown command.
- void helpHelp ()

  Help page for the help command.
- void setdateHelp ()

  Help page for the setdate() method.
- void getdateHelp ()

  Help page for the getdate() method.
- void gettimeHelp ()

  Help page for gettime() method.
- void settimeHelp ()

  Help page for settime() method.
- void versionOs ()

6.45.1 Function Documentation

6.45.1.1 cmd_help()

```
int cmd_help (
            char ∗ command )
```

Prints help message for command.

Prints out a help message and basic syntax for a specific command

Parameters

command	Command which the user needs basic information and syntax for

Returns

1 upon success, -1 upon error

6.45.1.2 getdateHelp()

```
void getdateHelp ( )
```

Help page for the getdate() method.

Prints out the name, usage, return and description for the getdate() method.

6.45.1.3 gettimeHelp()

```
void gettimeHelp ( )
```

Help page for gettime() method.

Prints out the name, usage, return and description for the gettime() method.

6.45.1.4 helpHelp()

```
void helpHelp ( )
```

Help page for the help command.

Prints out the name, usage, return and description for the help command.

6.45.1.5 helpList()

```
void helpList ( )
```

Displays a list of common system commands.

Displays a list of common system commands for the user.

6.45.1.6 setdateHelp()

```
void setdateHelp ( )
```

Help page for the setdate() method.

Prints out the name, usage, and description for the setdate() method.

6.45.1.7 settimeHelp()

```
void settimeHelp ( )
```

Help page for settime() method.

Prints out the name, usage, and description for the settime() method.

6.45.1.8 shutdownHelp()

```
void shutdownHelp ( )
```

Help page for the shutdown command.

Prints out the name, usage, and description for the shutdown system command.

6.45.1.9 versionOs()

```
void versionOs ( )
```

6.46 /home/maximillian/Desktop/MAMA/term/cmds/help.c File Reference

```
#include <lib/out.h>
```

Functions

- int cmd_help (char ∗command)

  Prints help message for command.
- void versionHelp ()

  Help page for the version command.
- void helpList ()

  Displays a list of common system commands.
- void shutdownHelp ()

  Help page for the shutdown command.
- void helpHelp ()

  Help page for the help command.
- void clearHelp ()

  Help page for clear.
- void aliasHelp ()

  Help page for alias.
- void setdateHelp ()

  Help page for the setdate() method.
- void getdateHelp ()

  Help page for the getdate() method.
- void gettimeHelp ()

  Help page for gettime() method.

- void settimeHelp ()

  Help page for settime() method.
- void createpcbHelp ()

  Help page for createpcb.
- void deletepcbHelp ()

  Help page for deletepcb.
- void showpcbHelp ()

  Help page for showpcb.
- void showallpcbHelp ()

  Help page for showallpcb.
- void showreadypcbHelp ()

  Help page for showreadypcb.
- void showblockedpcbHelp ()

  Help page for showblockedpcb.
- void blockHelp ()

  Help page for block.
- void unblockHelp ()

  Help page for unblock.
- void setpriorityHelp ()

  Help page for setpriority.
- void resumeHelp ()

  Help page for resume.
- void suspendHelp ()

  Help page for suspend.
- void loadr3Help ()

  Help page for loadr3.
- void setalarmHelp ()

  Help page for setalarm.
- void showalarmsHelp ()

  Help page for showalarm.
- void freealarmHelp ()

  Help page for freealarm.
- void resumeallHelp ()

  Help page for resumeallpcb.
- void showallocHelp ()

  Help page for showalloc.
- void showfreeHelp ()

  Help page for showfree.
- void isemptyHelp ()

  Help page for isempty.

6.46.1 Function Documentation

6.46.1.1 aliasHelp()

```
void aliasHelp ( )
```

Help page for alias.

Displays the alias help pages

6.46.1.2 blockHelp()

```
void blockHelp ( )
```

Help page for block.

Displays the block help page

6.46.1.3 clearHelp()

```
void clearHelp ( )
```

Help page for clear.

Displays the clear help pages

6.46.1.4 cmd_help()

```
int cmd_help (
             char * command )
```

Prints help message for command.

Prints out a help message and basic syntax for a specific command

Parameters

command	Command which the user needs basic information and syntax for

Returns

1 upon success, -1 upon error

6.46.1.5 createpcbHelp()

```
void createpcbHelp ( )
```

Help page for createpcb.

Displays the createpcb help page

6.46.1.6 deletepcbHelp()

```
void deletepcbHelp ( )
```

Help page for deletepcb.

Displays the deletepcb help page

6.46.1.7 freealarmHelp()

```
void freealarmHelp ( )
```

Help page for freealarm.

Displays the freealarm help page

6.46.1.8 getdateHelp()

```
void getdateHelp ( )
```

Help page for the getdate() method.

Prints out the name, usage, return and description for the getdate() method.

6.46.1.9 gettimeHelp()

```
void gettimeHelp ( )
```

Help page for gettime() method.

Prints out the name, usage, return and description for the gettime() method.

6.46.1.10 helpHelp()

```
void helpHelp ( )
```

Help page for the help command.

Prints out the name, usage, return and description for the help command.

6.46.1.11 helpList()

```
void helpList ( )
```

Displays a list of common system commands.

Displays a list of common system commands for the user.

6.46.1.12 isemptyHelp()

```
void isemptyHelp ( )
```

Help page for isempty.

Displays the isempty help pages

6.46.1.13 loadr3Help()

```
void loadr3Help ( )
```

Help page for loadr3.

Displays the loadr3 help page

6.46.1.14 resumeallHelp()

```
void resumeallHelp ( )
```

Help page for resumeallpcb.

Displays the resumeallpcb help page

6.46.1.15 resumeHelp()

```
void resumeHelp ( )
```

Help page for resume.

Displays the resume help page

6.46.1.16 setalarmHelp()

```
void setalarmHelp ( )
```

Help page for setalarm.

Displays the setalarm help page

6.46.1.17 setdateHelp()

```
void setdateHelp ( )
```

Help page for the setdate() method.

Prints out the name, usage, and description for the setdate() method.

6.46.1.18 setpriorityHelp()

```
void setpriorityHelp ( )
```

Help page for setpriority.

Displays the setpriority help page

6.46.1.19 settimeHelp()

```
void settimeHelp ( )
```

Help page for settime() method.

Prints out the name, usage, and description for the settime() method.

6.46.1.20 showalarmsHelp()

```
void showalarmsHelp ( )
```

Help page for showalarm.

Displays the showalarm help page

6.46.1.21 showallocHelp()

```
void showallocHelp ( )
```

Help page for showalloc.

Displays the showalloc help page

6.46.1.22 showallpcbHelp()

```
void showallpcbHelp ( )
```

Help page for showallpcb.

Displays the showallpcb help page

6.46.1.23 showblockedpcbHelp()

```
void showblockedpcbHelp ( )
```

Help page for showblockedpcb.

Displays the showblockedpcb help page

6.46.1.24 showfreeHelp()

```
void showfreeHelp ( )
```

Help page for showfree.

Displays the showfree help pages

6.46.1.25 showpcbHelp()

```
void showpcbHelp ( )
```

Help page for showpcb.

Displays the showpcb help page

6.46.1.26 showreadypcbHelp()

```
void showreadypcbHelp ( )
```

Help page for showreadypcb.

Displays the showreadypcb help page

6.46.1.27 shutdownHelp()

```
void shutdownHelp ( )
```

Help page for the shutdown command.

Prints out the name, usage, and description for the shutdown system command.

6.46.1.28 suspendHelp()

```
void suspendHelp ( )
```

Help page for suspend.

Displays the suspend help page

6.46.1.29 unblockHelp()

```
void unblockHelp ( )
```

Help page for unblock.

Displays te unblock help page

6.46.1.30 versionHelp()

```
void versionHelp ( )
```

Help page for the version command.

Displays the current verson of the system.

6.47 /home/maximillian/Desktop/MAMA/term/cmds/shutdown.c File Reference

```
#include <lib/out.h>
```

Functions

- int cmd_shutdown (char ∗arg_str)

  Handler for calls to the shutdown command.

6.47.1 Function Documentation

6.47.1.1 cmd_shutdown()

```
int cmd_shutdown (
            char ∗ arg_str )
```

Handler for calls to the shutdown command.

Prompts for user confirmation before shutting the system down.

Parameters

arg_str	The arguments passed to the shutdown command. Unused by the handler.

Returns

The exit code of the command, indicating whether or not the user confirmed the request to shutdown the system. Returns 0 if the user confirmed the request, 1 otherwise.

6.48 /home/maximillian/Desktop/MAMA/term/cmds/version.c File Reference

```
#include <lib/out.h>
```

Functions

- int cmd_version (char ∗arg_str)

  Handler for the version command.

6.48.1 Function Documentation

6.48.1.1 cmd_version()

```
int cmd_version (
            char * arg_str )
```

Handler for the version command.

Prints the current version of the operating system.

Parameters

arg_str	The arguments passed to the version command. Unused by the handler.

Returns

The exit code of the command, always 0.

6.49 /home/maximillian/Desktop/MAMA/term/commands.h File Reference

```
#include "cmds/help.c"
#include "cmds/shutdown.c"
#include "cmds/echo.c"
#include "cmds/version.c"
#include "cmds/argtest.c"
#include "cmds/pcb.c"
#include "cmds/clear.c"
```

6.50 commands.h

Go to the documentation of this file.

```
#ifndef COMMANDS_H
#define COMMANDS_H

#include "cmds/help.c"
#include "cmds/shutdown.c"
#include "cmds/echo.c"
#include "cmds/version.c"
#include "cmds/argtest.c"
#include "cmds/pcb.c"
#include "cmds/clear.c"

#endif
```

6.51 /home/maximillian/Desktop/MAMA/term/commhand.c File Reference

```
#include <include/string.h>
#include <modules/mpx_supt.h>
#include "visuals/colorize.c"
#include "visuals/clear.c"
#include "history.c"
#include "commhand.h"
#include "commands.h"
#include "visuals/syntax_highlight.h"
#include "visuals/hints.h"
#include <lib/out.c>
#include "dnt/dnt.c"
#include "utils.h"
#include "ascii/mama.c"
#include "dispatch/context.c"
#include "pcb/pcb.c"
#include "memory_management/mm.c"
#include <term/args.h>
```

Classes

- struct cmd_mapping

Typedefs

- typedef int(∗ cmd_func_t) (char ∗)
- typedef struct cmd_mapping cmd_mapping

Functions

- int cmd_alias (char ∗)
- int is_name_char (char)

  Returns whether or not the specified character is a valid character in an identifier, such as a command or argument name.
- void extract_cmd_name (char ∗, char ∗, int ∗, int ∗)
- cmd_mapping ∗ fetch_cmd_mapping (char ∗)
- void commhand ()

  Displays command line and interprets inputted commands.

Variables

- cmd_mapping cmd_mappings [MAX_CMD_COUNT]
- pcb_queue_t ∗ priority_queue

6.51.1 Typedef Documentation

6.51.1.1 cmd_func_t

```
typedef int(* cmd_func_t) (char *)
```

6.51.1.2 cmd_mapping

```
typedef struct cmd_mapping cmd_mapping
```

6.51.2 Function Documentation

6.51.2.1 cmd_alias()

```
int cmd_alias (
             char * arg_str )
```

6.51.2.2 commhand()

```
void commhand ( )
```

Displays command line and interprets inputted commands.

Parses through the input that was polled from the command line and interprets the command that was inputted (typically the first word)

Returns

Returns 0 upon success, -1 upon error

6.51.2.3 extract_cmd_name()

```
void extract_cmd_name (
             char * cmd_str,
             char * cmd_name,
             int * cmd_name_len,
             int * args_start_index )
```

6.51.2.4 fetch_cmd_mapping()

```
cmd_mapping * fetch_cmd_mapping (
             char * cmd_name )
```

6.51.2.5 is_name_char()

```
int is_name_char (
             char c )
```

Returns whether or not the specified character is a valid character in an identifier, such as a command or argument name.

Parameters

c	The character to test.

Returns

True if the specified character c is valid in an identifier, false otherwise.

6.51.3 Variable Documentation

6.51.3.1 cmd_mappings

```
cmd_mapping cmd_mappings[MAX_CMD_COUNT]
```

6.51.3.2 priority_queue

```
pcb_queue_t* priority_queue  [extern]
```

6.52 /home/maximillian/Desktop/MAMA/term/commhand.h File Reference

Macros

- #define MAX_CMD_STRING_LEN 100
- #define MAX_CMD_NAME_LEN 30
- #define MAX_CMD_HIST_LEN 20
- #define MAX_CMD_ARG_NAME_LEN 30
- #define MAX_CMD_ARG_VALUE_LEN 40
- #define MAX_CMD_FLAG_COUNT 10
- #define MAX_CMD_NAMED_ARG_COUNT 10
- #define MAX_CMD_UNNAMED_ARG_COUNT 10
- #define MAX_CMD_COUNT 200

Functions

- void commhand ()

  Displays command line and interprets inputted commands.

6.52.1 Macro Definition Documentation

6.52.1.1 MAX_CMD_ARG_NAME_LEN

```
#define MAX_CMD_ARG_NAME_LEN 30
```

6.52.1.2 MAX_CMD_ARG_VALUE_LEN

```
#define MAX_CMD_ARG_VALUE_LEN 40
```

6.52.1.3 MAX_CMD_COUNT

```
#define MAX_CMD_COUNT 200
```

6.52.1.4 MAX_CMD_FLAG_COUNT

```
#define MAX_CMD_FLAG_COUNT 10
```

6.52.1.5 MAX_CMD_HIST_LEN

```
#define MAX_CMD_HIST_LEN 20
```

6.52.1.6 MAX_CMD_NAME_LEN

```
#define MAX_CMD_NAME_LEN 30
```

6.52.1.7 MAX_CMD_NAMED_ARG_COUNT

```
#define MAX_CMD_NAMED_ARG_COUNT 10
```

6.52.1.8 MAX_CMD_STRING_LEN

```
#define MAX_CMD_STRING_LEN 100
```

6.52.1.9 MAX_CMD_UNNAMED_ARG_COUNT

```
#define MAX_CMD_UNNAMED_ARG_COUNT 10
```

6.52.2 Function Documentation

6.52.2.1 commhand()

```
void commhand ( )
```

Displays command line and interprets inputted commands.

Parses through the input that was polled from the command line and interprets the command that was inputted (typically the first word)

Returns

Returns 0 upon success, -1 upon error

6.53 commhand.h

Go to the documentation of this file.

```
/* the logic for each command the user has to run is contained in a separate file in term/cmds
 * each file should contain a function to run this command and possibly any helper functions the command
      needs to run
 * include each of these files below - make sure to add an #include directive if you write a new command
 */
#ifndef COMMHAND_H
#define COMMHAND_H

#define MAX_CMD_STRING_LEN 100
#define MAX_CMD_NAME_LEN 30
#define MAX_CMD_HIST_LEN 20
#define MAX_CMD_ARG_NAME_LEN 30
#define MAX_CMD_ARG_VALUE_LEN 40
#define MAX_CMD_FLAG_COUNT 10
#define MAX_CMD_NAMED_ARG_COUNT 10
#define MAX_CMD_UNNAMED_ARG_COUNT 10

#define MAX_CMD_COUNT 200

void commhand();
#endif
```

6.54 /home/maximillian/Desktop/MAMA/term/dispatch/context.c File Reference

```
#include "context.h"
#include "term/pcb/pcb.h"
#include "procsr3.c"
#include <lib/out.h>
```

Functions

- void yield ()

  Causes commhand to yield.
- int loadr3 (char ∗p)

  Loads r3 'processes'.
- pcb_t ∗ dispatcher (char ∗name, void(∗func)(void))

  Stores context on the stack.

6.54.1 Function Documentation

6.54.1.1 dispatcher()

```
pcb_t * dispatcher (
            char * pcb,
            void(*)(void) func )
```

Stores context on the stack.

With a given pcb and method to run, the dispatcher will store context registers onto the PCB stack.

Parameters

pcb	PCB where context is stored
func	Method that is ran within the process

6.54.1.2 loadr3()

```
int loadr3 (
            char * p )
```

Loads r3 'processes'.

Loads all r3 'processes' into memory in a suspended ready state at any priority of the users choosing

Parameters

p	Empty parameter

Returns

Returns 0 upon success, 1 upon error ∗∗This may change

6.54.1.3 yield()

```
void yield ( )
```

Causes commhand to yield.

Forces commhand to yield to other processes. If any processes are in the ready queue, they will be executed.

6.55 /home/maximillian/Desktop/MAMA/term/dispatch/context.h File Reference

```
#include "term/pcb/pcb.h"
```

Classes

- struct context

  Context of the currently operating process.

Typedefs

- typedef struct context context

  Context of the currently operating process.

Functions

- void yield ()

  Causes commhand to yield.
- int loadr3 (char ∗p)

  Loads r3 'processes'.
- pcb_t ∗ dispatcher (char ∗pcb, void(∗func)(void))

  Stores context on the stack.

6.55.1 Typedef Documentation

6.55.1.1 context

```
typedef struct context context
```

Context of the currently operating process.

6.55.2 Function Documentation

6.55.2.1 dispatcher()

```
pcb_t * dispatcher (
            char * pcb,
            void(*)(void) func )
```

Stores context on the stack.

With a given pcb and method to run, the dispatcher will store context registers onto the PCB stack.

Parameters

pcb	PCB where context is stored
func	Method that is ran within the process

6.55.2.2 loadr3()

```
int loadr3 (
            char * p )
```

Loads r3 'processes'.

Loads all r3 'processes' into memory in a suspended ready state at any priority of the users choosing

Parameters

p	Empty parameter

Returns

Returns 0 upon success, 1 upon error ∗∗This may change

6.55.2.3 yield()

```
void yield ( )
```

Causes commhand to yield.

Forces commhand to yield to other processes. If any processes are in the ready queue, they will be executed.

6.56 context.h

Go to the documentation of this file.

```
#ifndef CONTEXT_H
#define CONTEXT_H

#include "term/pcb/pcb.h"

typedef struct context {
    u32int gs, fs, es, ds;

     u32int edi, esi, ebp, esp, ebx, edx, ecx, eax;

     // Other special registers
     u32int eip, cs, eflags;
} context;

void yield();

int loadr3(char * p);

pcb_t * dispatcher(char * pcb, void (* func) (void));



#endif
```

6.57 /home/maximillian/Desktop/MAMA/term/dispatch/procsr3.c File Reference

```
#include "../include/system.h"
#include "../include/core/serial.h"
#include "../modules/mpx_supt.h"
#include "procsr3.h"
```

Macros

- #define RC_1 1
- #define RC_2 2
- #define RC_3 3
- #define RC_4 4
- #define RC_5 5

Functions

- void proc1 ()
- void proc2 ()
- void proc3 ()
- void proc4 ()
- void proc5 ()

Variables

- char ∗ msg1 = "proc1 dispatched\n"
- char ∗ msg2 = "proc2 dispatched\n"
- char ∗ msg3 = "proc3 dispatched\n"
- char ∗ msg4 = "proc4 dispatched\n"
- char ∗ msg5 = "proc5 dispatched\n"
- int msgSize = 18
- char ∗ er1 = "proc1 ran after it was terminated"
- char ∗ er2 = "proc2 ran after it was terminated"
- char ∗ er3 = "proc3 ran after it was terminated"
- char ∗ er4 = "proc4 ran after it was terminated"
- char ∗ er5 = "proc5 ran after it was terminated"
- int erSize = 34

6.57.1 Macro Definition Documentation

6.57.1.1 RC_1

```
#define RC_1 1
```

6.57.1.2 RC_2

```
#define RC_2 2
```

6.57.1.3 RC_3

```
#define RC_3 3
```

6.57.1.4 RC_4

```
#define RC_4 4
```

6.57.1.5 RC_5

```
#define RC_5 5
```

6.57.2 Function Documentation

6.57.2.1 proc1()

```
void proc1 ( )
```

6.57.2.2 proc2()

```
void proc2 ( )
```

6.57.2.3 proc3()

```
void proc3 ( )
```

6.57.2.4 proc4()

```
void proc4 ( )
```

6.57.2.5 proc5()

```
void proc5 ( )
```

6.57.3 Variable Documentation

6.57.3.1 er1

```
char* er1 = "proc1 ran after it was terminated"
```

6.57.3.2 er2

```
char* er2 = "proc2 ran after it was terminated"
```

6.57.3.3 er3

```
char* er3 = "proc3 ran after it was terminated"
```

6.57.3.4 er4

```
char* er4 = "proc4 ran after it was terminated"
```

6.57.3.5 er5

```
char* er5 = "proc5 ran after it was terminated"
```

6.57.3.6 erSize

```
int erSize = 34
```

6.57.3.7 msg1

```
char* msg1 = "proc1 dispatched\n"
```

6.57.3.8 msg2

```
char* msg2 = "proc2 dispatched\n"
```

6.57.3.9 msg3

```
char* msg3 = "proc3 dispatched\n"
```

6.57.3.10 msg4

```
char* msg4 = "proc4 dispatched\n"
```

6.57.3.11 msg5

```
char* msg5 = "proc5 dispatched\n"
```

6.57.3.12 msgSize

```
int msgSize = 18
```

6.58 /home/maximillian/Desktop/MAMA/term/dispatch/procsr3.h File Reference

Functions

- void proc1 ()
- void proc2 ()
- void proc3 ()
- void proc4 ()
- void proc5 ()

6.58.1 Function Documentation

6.58.1.1 proc1()

```
void proc1 ( )
```

6.58.1.2 proc2()

```
void proc2 ( )
```

6.58.1.3 proc3()

```
void proc3 ( )
```

6.58.1.4 proc4()

```
void proc4 ( )
```

6.58.1.5 proc5()

```
void proc5 ( )
```

6.59 procsr3.h

Go to the documentation of this file.

```
#ifndef PROCSR3_H
#define PROCSR3_H

void proc1();

void proc2();

void proc3();

void proc4();

void proc5();

#endif
```

6.60 /home/maximillian/Desktop/MAMA/term/dnt/dnt.c File Reference

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

Functions

- int setdate (char ∗date)

  Sets the date of the system.
- int setDateInMemory (int month, int day, int year)

  Sets the date in memory.
- int getdate (char ∗p)

  Gets the date of the system.
- int settime (char ∗time)

  Sets the time of the system.
- void setTimeInMemory (int hour, int minute, int second)

  Sets the time into memory.
- int gettime (char ∗p)

  Gets the system time.
- unsigned char ItoBCD (unsigned int value)

  Converts 32-bit integer to 8-bit BCD.
- unsigned int BCDtoI (unsigned char value)

  Converts 8-bit BCD to 32-bit integer.
- char ∗ intToMonth (int value)

  Converts masked int into string month.
- char ∗ intToDayOfWeek (int value)

  Converts integer to string day of the week.
- int daysInMonth (int month, int year)

  Calculates the number of days in a month.
- int setAlarm (char ∗args)

  Set an alarm.
- int showAlarms (char ∗p)

  Show all alarms.
- int freeAlarm (char ∗time)

  Remove alarm from alarms.
- void currentTime ()

  Current time.
- void dispatchAlarm ()

  Alarm process.

Variables

- char alarms [10][6] = { "\0", "\0", "\0", "\0", "\0", "\0", "\0", "\0", "\0", "\0" }
- char messages [10][32] = { "\0", "\0", "\0", "\0", "\0", "\0", "\0", "\0", "\0", "\0" }
- char current_time [6]

6.60.1 Function Documentation

6.60.1.1 BCDtoI()

```
unsigned int BCDtoI (
             unsigned char value )
```

Converts 8-bit BCD to 32-bit integer.

Converts an 8-bit BCD unsigned char to a 32-bit unsigned integer.

Parameters

value	8-bit BCD value that will be converted to 32-bit int

Returns

Returns 32-bit unsigned int

6.60.1.2 currentTime()

```
void currentTime ( )
```

Current time.

Gets the current time and stores it into string.

6.60.1.3 daysInMonth()

```
int daysInMonth (
             int month,
             int year )
```

Calculates the number of days in a month.

Calculates the number of days in the month based upon which month it is. If year is divisible by four then it is a leap year and will add 1 day for February for a total of 29 days. Otherwise, February will be 28 days.

Parameters

month	The month in the year (January = 1...December = 12)
year	The year that was being set

Returns

Returns the number of days in the month

6.60.1.4 dispatchAlarm()

```
void dispatchAlarm ( )
```

Alarm process.

The function that will be used during context switching. This will check all alarm times against the current time

6.60.1.5 freeAlarm()

```
int freeAlarm (
             char * alarm )
```

Remove alarm from alarms.

Removes the alarm from the alarm list and 'frees' the spot

Parameters

time	Alarm to remove from list

Returns

Returns 0 upon success, -1 upon error

6.60.1.6 getdate()

```
int getdate (
             char * p )
```

Gets the date of the system.

Returns a string that represents the current date of the system. This is in the format DayOfWeek, Month Day, Year Ex: Wednesday, August 25, 2021

Parameters

p	Empty paremeter that is required to call this method. Does not do anything.

Returns

Returns 1 upon success, -1 upon error

6.60.1.7 gettime()

```
int gettime (
            char * p )
```

Gets the system time.

Gets the system time from memory by reading from the corresponding memory address. Time will be writtin to the interface in the syntax of Hour:Minute:Second Ex: 10:06:23

Parameters

Empty	parameter that does not do anything. Required in order to call from commhand

Returns

Returns 1 upon success, -1 upon error

6.60.1.8 intToDayOfWeek()

```
char * intToDayOfWeek (
            int value )
```

Converts integer to string day of the week.

Converts a masked integer into an unmasked string day of the week. The days of the week are Sunday to Saturday and are 1 to 7 respectivley. 1 = Sunday 2 = Monday ... 7 = Saturday

Parameters

value	The masked integer value of month

Returns

Returns the unasked string value of month

6.60.1.9 intToMonth()

```
char * intToMonth (
            int value )
```

Converts masked int into string month.

Converts integer to a string month.

Converts the masked integer value of month and converts it into an unmasked string of month

Parameters

value	Masked integer month

Returns

Returns unmasked string of month

6.60.1.10 ItoBCD()

```
unsigned char ItoBCD (
            unsigned int value )
```

Converts 32-bit integer to 8-bit BCD.

Uses basic arithmetic and bit shifting to convert from 32-bit integer to 8-bit BCD.

Parameters

value	The 32-bit integer that is converted to BCD

Returns

8-bit BCD number as an unsigned char

6.60.1.11 setAlarm()

```
int setAlarm (
            char * args )
```

Set an alarm.

Sets an alarm which will print a user defined message. Alarm will go off at specified time.

Parameters

args	Time and (optional) message

Returns

Returns 0 upon success and -1 upon error

6.60.1.12 setdate()

```
int setdate (
            char * date )
```

Sets the date of the system.

Parses the parameter to setdate, breaking the parameter into month, day and year before passing it to setDateIn↩
Memory. The basic syntax is month.day.year

Parameters

date	The parameter that is passed with setdate. This string is parsed and each segment is converted to a 32-bit int.

Returns

Returns 1 upon success, -1 upon error

6.60.1.13 setDateInMemory()

```
int setDateInMemory (
            int month,
            int day,
            int year )
```

Sets the date in memory.

Sets the date in memory by assigning the values to the appropriate places in memory. This method is called by the setdate method.

Parameters

month	The month (1 = January ... 12 = December)
day	The day in the month. Can be between 0 and 32
year	The current year. This is a 2-digit number

Returns

Returns 1 upon success, -1 upon error

6.60.1.14 settime()

```
int settime (
             char * time )
```

Sets the time of the system.

Takes the parameter which will be parsed into 32-bit int (later converted to BCD) and sets it into memory. The syntax is Hour.Minute.Second Ex: 10.23.00

Parameters

The	parameter passed with the settime call

Returns

Returns 1 upon success, -1 upon error

6.60.1.15 setTimeInMemory()

```
void setTimeInMemory (
             int hour,
             int minute,
             int second )
```

Sets the time into memory.

This method is called by the settime method. Writes the data into memory. First converts all parameter from 32-bit int to 8-bit BCD and then writes to the appropriate address.

Parameters

hour	32-bit int hour
minute	32-bit int minute
second	32-bit int second

6.60.1.16 showAlarms()

```
int showAlarms (
             char * p )
```

Show all alarms.

Print all alarms currently in the alarm list

Parameters

p	Empty parameters

Returns

Returns 0 upon success, -1 upon error

6.60.2 Variable Documentation

6.60.2.1 alarms

```
char alarms[10][6] = { "\0", "\0", "\0", "\0", "\0", "\0", "\0", "\0", "\0", "\0" }
```

6.60.2.2 current_time

```
char current_time[6]
```

6.60.2.3 messages

```
char messages[10][32] = { "\0", "\0", "\0", "\0", "\0", "\0", "\0", "\0", "\0", "\0" }
```

6.61 /home/maximillian/Desktop/MAMA/term/dnt/dnt.h File Reference

Macros

- #define MAX_HOURS 23

  The largest value that the user can set their hours to.
- #define MAX_MINUTES 59

  The largest value that the user can set their minutes to.
- #define MAX_SECONDS 59

  The largest value that the user can set their seconds to.
- #define MAX_YEAR 99

  The largest value that the user can set their year to.
- #define MAX_MONTH 12

  The largest value that the user can set their month to.

- #define MAX_DAY 31

  The largest value that the user can set their day to.
- #define MIN_YEAR 10

  Minimum year that can be set in memory.
- #define MIN_MONTH 1

  Minimum month that can be set in memory.
- #define MIN_DAY 1

  Minimum day that can be set in memory.
- #define EPOCH_YEAR 1970

  Unix Epoch year.
- #define EPOCH_FIRST_DAY_OF_YEAR 1

  Unix Epoch first day of the year.
- #define EPOCH_FIRST_MONTH_OF_YEAR 1

  Unix Epoch first month of the year.
- #define EPOCH_FIRST_DAY_OF_WEEK_OF_YEAR 5

  Unix Epoch first day of the week in the year.
- #define DAYS_IN_YEAR 365

  Number of days in a normal year.
- #define DAYS_IN_LEAP_YEAR 366

  Number of days in a leap year.
- #define MIN 0

  Minimum value that can be set for hours, minutes, and seconds.

Functions

- int setdate (char ∗date)

  Sets the date of the system.
- int setDateInMemory (int month, int day, int year)

  Sets the date in memory.
- int getdate (char ∗p)

  Gets the date of the system.
- int settime (char ∗time)

  Sets the time of the system.
- void setTimeInMemory (int hour, int minute, int second)

  Sets the time into memory.
- int gettime (char ∗p)

  Gets the system time.
- unsigned char ItoBCD (unsigned int value)

  Converts 32-bit integer to 8-bit BCD.
- unsigned int BCDtoI (unsigned char value)

  Converts 8-bit BCD to 32-bit integer.
- char ∗ intToMonth (int value)

  Converts integer to a string month.
- char ∗ intToDayOfWeek (int value)

  Converts integer to string day of the week.
- int daysInMonth (int month, int year)

  Calculates the number of days in a month.
- int setAlarm (char ∗args)

  Set an alarm.
- int showAlarms (char ∗p)

Show all alarms.
- int freeAlarm (char ∗alarm)

    Remove alarm from alarms.
- void dispatchAlarm ()

    Alarm process.
- void currentTime ()

    Current time.

6.61.1 Macro Definition Documentation

6.61.1.1 DAYS_IN_LEAP_YEAR

```
#define DAYS_IN_LEAP_YEAR 366
```

Number of days in a leap year.

6.61.1.2 DAYS_IN_YEAR

```
#define DAYS_IN_YEAR 365
```

Number of days in a normal year.

6.61.1.3 EPOCH_FIRST_DAY_OF_WEEK_OF_YEAR

```
#define EPOCH_FIRST_DAY_OF_WEEK_OF_YEAR 5
```

Unix Epoch first day of the week in the year.

6.61.1.4 EPOCH_FIRST_DAY_OF_YEAR

```
#define EPOCH_FIRST_DAY_OF_YEAR 1
```

Unix Epoch first day of the year.

6.61.1.5 EPOCH_FIRST_MONTH_OF_YEAR

```
#define EPOCH_FIRST_MONTH_OF_YEAR 1
```

Unix Epoch first month of the year.

6.61.1.6 EPOCH_YEAR

```
#define EPOCH_YEAR 1970
```

Unix Epoch year.

6.61.1.7 MAX_DAY

```
#define MAX_DAY 31
```

The largest value that the user can set their day to.

6.61.1.8 MAX_HOURS

```
#define MAX_HOURS 23
```

The largest value that the user can set their hours to.

6.61.1.9 MAX_MINUTES

```
#define MAX_MINUTES 59
```

The largest value that the user can set their minutes to.

6.61.1.10 MAX_MONTH

```
#define MAX_MONTH 12
```

The largest value that the user can set their month to.

6.61.1.11 MAX_SECONDS

```
#define MAX_SECONDS 59
```

The largest value that the user can set their seconds to.

6.61.1.12 MAX_YEAR

```
#define MAX_YEAR 99
```

The largest value that the user can set their year to.

6.61.1.13 MIN

```
#define MIN 0
```

Minimum value that can be set for hours, minutes, and seconds.

6.61.1.14 MIN_DAY

```
#define MIN_DAY 1
```

Minimum day that can be set in memory.

6.61.1.15 MIN_MONTH

```
#define MIN_MONTH 1
```

Minimum month that can be set in memory.

6.61.1.16 MIN_YEAR

```
#define MIN_YEAR 10
```

Minimum year that can be set in memory.

6.61.2 Function Documentation

6.61.2.1 BCDtoI()

```
unsigned int BCDtoI (
             unsigned char value )
```

Converts 8-bit BCD to 32-bit integer.

Converts an 8-bit BCD unsigned char to a 32-bit unsigned integer.

Parameters

value	8-bit BCD value that will be converted to 32-bit int

Returns

Returns 32-bit unsigned int

6.61.2.2 currentTime()

```
void currentTime ( )
```

Current time.

Gets the current time and stores it into string.

6.61.2.3 daysInMonth()

```
int daysInMonth (
             int month,
             int year )
```

Calculates the number of days in a month.

Calculates the number of days in the month based upon which month it is. If year is divisible by four then it is a leap year and will add 1 day for February for a total of 29 days. Otherwise, February will be 28 days.

Parameters

month	The month in the year (January = 1...December = 12)
year	The year that was being set

Returns

Returns the number of days in the month

6.61.2.4 dispatchAlarm()

```
void dispatchAlarm ( )
```

Alarm process.

The function that will be used during context switching. This will check all alarm times against the current time

6.61.2.5 freeAlarm()

```
int freeAlarm (
            char * alarm )
```

Remove alarm from alarms.

Removes the alarm from the alarm list and 'frees' the spot

Parameters

time	Alarm to remove from list

Returns

Returns 0 upon success, -1 upon error

6.61.2.6 getdate()

```
int getdate (
            char * p )
```

Gets the date of the system.

Returns a string that represents the current date of the system. This is in the format DayOfWeek, Month Day, Year Ex: Wednesday, August 25, 2021

Parameters

p	Empty paremeter that is required to call this method. Does not do anything.

Returns

Returns 1 upon success, -1 upon error

6.61.2.7 gettime()

```
int gettime (
            char * p )
```

Gets the system time.

Gets the system time from memory by reading from the corresponding memory address. Time will be writtin to the interface in the syntax of Hour:Minute:Second Ex: 10:06:23

Parameters

Empty	parameter that does not do anything. Required in order to call from commhand

Returns

Returns 1 upon success, -1 upon error

6.61.2.8 intToDayOfWeek()

```
char * intToDayOfWeek (
            int value )
```

Converts integer to string day of the week.

Converts a masked integer into an unmasked string day of the week. The days of the week are Sunday to Saturday and are 1 to 7 respectivley. 1 = Sunday 2 = Monday ... 7 = Saturday

Parameters

value	The masked integer value of month

Returns

Returns the unasked string value of month

6.61.2.9 intToMonth()

```
char * intToMonth (
            int value )
```

Converts integer to a string month.

Converts masked int into string month.

Converts a masked integer into an unmasked string month. The months are January to December and are 1 to 12 respectivley. 1 = January 2 = February ... 13 = December

Parameters

value	The masked month

Returns

Returns unmasked string month

Converts the masked integer value of month and converts it into an unmasked string of month

Parameters

value	Masked integer month

Returns

Returns unmasked string of month

Converts integer to a string month.

Converts the masked integer value of month and converts it into an unmasked string of month

Parameters

value	Masked integer month

Returns

Returns unmasked string of month

6.61.2.10 ItoBCD()

```
unsigned char ItoBCD (
            unsigned int value )
```

Converts 32-bit integer to 8-bit BCD.

Uses basic arithmetic and bit shifting to convert from 32-bit integer to 8-bit BCD.

Parameters

value	The 32-bit integer that is converted to BCD

Returns

8-bit BCD number as an unsigned char

6.61.2.11 setAlarm()

```
int setAlarm (
            char * args )
```

Set an alarm.

Sets an alarm which will print a user defined message. Alarm will go off at specified time.

Parameters

args	Time and (optional) message

Returns

Returns 0 upon success and -1 upon error

6.61.2.12 setdate()

```
int setdate (
            char * date )
```

Sets the date of the system.

Parses the parameter to setdate, breaking the parameter into month, day and year before passing it to setDateIn↩ Memory. The basic syntax is month.day.year

Parameters

date	The parameter that is passed with setdate. This string is parsed and each segment is converted to a 32-bit int.

Returns

Returns 1 upon success, -1 upon error

6.61.2.13 setDateInMemory()

```
int setDateInMemory (
            int month,
            int day,
            int year )
```

Sets the date in memory.

Sets the date in memory by assigning the values to the appropriate places in memory. This method is called by the setdate method.

Parameters

month	The month (1 = January ... 12 = December)
day	The day in the month. Can be between 0 and 32
year	The current year. This is a 2-digit number

Returns

Returns 1 upon success, -1 upon error

6.61.2.14 settime()

```
int settime (
            char * time )
```

Sets the time of the system.

Takes the parameter which will be parsed into 32-bit int (later converted to BCD) and sets it into memory. The syntax is Hour.Minute.Second Ex: 10.23.00

Parameters

The	parameter passed with the settime call

Returns

Returns 1 upon success, -1 upon error

6.61.2.15 setTimeInMemory()

```
void setTimeInMemory (
            int hour,
            int minute,
            int second )
```

Sets the time into memory.

This method is called by the settime method. Writes the data into memory. First converts all parameter from 32-bit int to 8-bit BCD and then writes to the appropriate address.

Parameters

hour	32-bit int hour
minute	32-bit int minute
second	32-bit int second

6.61.2.16 showAlarms()

```
int showAlarms (
              char * p )
```

Show all alarms.

Print all alarms currently in the alarm list

Parameters

p	Empty parameters

Returns

Returns 0 upon success, -1 upon error

6.62 dnt.h

Go to the documentation of this file.

```
#define MAX_HOURS 23
#define MAX_MINUTES 59
#define MAX_SECONDS 59

#define MAX_YEAR 99
#define MAX_MONTH 12
#define MAX_DAY 31

#define MIN_YEAR 10
#define MIN_MONTH 1
#define MIN_DAY 1

#define EPOCH_YEAR 1970
#define EPOCH_FIRST_DAY_OF_YEAR 1
#define EPOCH_FIRST_MONTH_OF_YEAR 1
#define EPOCH_FIRST_DAY_OF_WEEK_OF_YEAR 5
#define DAYS_IN_YEAR 365
#define DAYS_IN_LEAP_YEAR 366

#define MIN 0

int setdate(char * date);

int setDateInMemory(int month,int day,int year);

int getdate(char * p);

int settime(char * time);

void setTimeInMemory(int hour, int minute, int second);

int gettime(char * p);

unsigned char ItoBCD(unsigned int value);

unsigned int BCDtoI(unsigned char value);

char * intToMonth(int value);

char * intToDayOfWeek(int value);

char * intToMonth(int value);

int daysInMonth(int month, int year);

int setAlarm(char * args);

int showAlarms(char * p);

int freeAlarm(char * alarm);

void dispatchAlarm();

void currentTime();
```

6.63 /home/maximillian/Desktop/MAMA/term/history.c File Reference

```
#include "commhand.h"
#include "visuals/cursor.c"
#include "visuals/syntax_highlight.h"
#include <lib/out.h>
```

Functions

- int circular_next_index (int)

  Whether or not the most recent entry in the user's command history has been discarded by calling hist_discard_↩ last_frame.
- int circular_prev_index (int i)

  Returns the index immediately preceding the specified index in cmd_hist, an array-based circular queue containing entries in the user's command history.
- void write_hist_to_buf (char ∗buf, int ∗index, int ∗len)

  Writes the history entry pointed to by cmd_hist_current_index to the specified buffer and prints the new buffer to the terminal.
- void hist_rewind (char ∗internal_buf, int ∗internal_index, int ∗internal_buf_len)

  Moves backwards 1 entry in the user's command history.
- void hist_forward (char ∗internal_buf, int ∗internal_index, int ∗internal_buf_len)

  Moves forwards 1 entry in the user's command history.
- void hist_discard_last_frame ()

  Removes the most recent command input from the user from the user's command history.
- char ∗ hist_next_frame ()

  Requests a buffer to write user input to that will become the most recent entry in the user's command history.

6.63.1 Function Documentation

6.63.1.1 circular_next_index()

```
int circular_next_index (
            int i )
```

Whether or not the most recent entry in the user's command history has been discarded by calling hist_discard_↩ last_frame.

Returns the index immediately following the specified index in cmd_hist, an array-based circular queue containing entries in the user's command history.

Parameters

i	An index in cmd_hist.

Returns

The index of the slot immediately following the slot at index i in cmd_hist.

6.63.1.2 circular_prev_index()

```
int circular_prev_index (
             int i )
```

Returns the index immediately preceding the specified index in cmd_hist, an array-based circular queue containing entries in the user's command history.

Parameters

i	An index in cmd_hist.

Returns

The index of the slot immediately preceding the slot at index i in cmd_hist.

6.63.1.3 hist_discard_last_frame()

```
void hist_discard_last_frame ( )
```

Removes the most recent command input from the user from the user's command history.

6.63.1.4 hist_forward()

```
void hist_forward (
             char * internal_buf,
             int * internal_index,
             int * internal_buf_len )
```

Moves forwards 1 entry in the user's command history.

Parameters

internal_buf	The buffer managed by low-level read operations containing user input from the terminal. Contents will be overwritten with the next entry in the user's command history.
internal_index	A pointer to the position of the cursor, managed by low-level read operations. The cursor position will be adjusted to point to the end of the line.
internal_buf_len	A pointer to the length of the buffer containing user input to the terminal. Will be adjusted to contain the length of the history entry being written to the buffer.

6.63.1.5 hist_next_frame()

```
char * hist_next_frame ( )
```

Requests a buffer to write user input to that will become the most recent entry in the user's command history.

Returns

A pointer to the first slot in a character buffer representing the next entry in the user's command history.

6.63.1.6 hist_rewind()

```
void hist_rewind (
             char * internal_buf,
             int * internal_index,
             int * internal_buf_len )
```

Moves backwards 1 entry in the user's command history.

Parameters

internal_buf	The buffer managed by low-level read operations containing user input from the terminal. Contents will be overwritten with the previous entry in the user's command history.
internal_index	A pointer to the position of the cursor, managed by low-level read operations. The cursor position will be adjusted to point to the end of the line.
internal_buf_len	A pointer to the length of the buffer containing user input to the terminal. Will be adjusted to contain the length of the history entry being written to the buffer.

6.63.1.7 write_hist_to_buf()

```
void write_hist_to_buf (
             char * buf,
             int ∗ index,
             int ∗ len )
```

Writes the history entry pointed to by cmd_hist_current_index to the specified buffer and prints the new buffer to the terminal.

Used internally by hist_rewind and hist_forward.

Parameters

buf	The buffer to write the current history entry to.
index	A pointer to the position of the cursor in the user's terminal.
len	A pointer to the length of the buffer.

6.64 /home/maximillian/Desktop/MAMA/term/history.h File Reference

Functions

- void hist_rewind (char ∗, int ∗, int ∗)

  Moves backwards 1 entry in the user's command history.
- void hist_forward (char ∗, int ∗, int ∗)

  Moves forwards 1 entry in the user's command history.
- char ∗ hist_next_frame ()

  Requests a buffer to write user input to that will become the most recent entry in the user's command history.

6.64.1 Function Documentation

6.64.1.1 hist_forward()

```
void hist_forward (
            char ∗ internal_buf,
            int ∗ internal_index,
            int ∗ internal_buf_len )
```

Moves forwards 1 entry in the user's command history.

Parameters

internal_buf	The buffer managed by low-level read operations containing user input from the terminal. Contents will be overwritten with the next entry in the user's command history.
internal_index	A pointer to the position of the cursor, managed by low-level read operations. The cursor position will be adjusted to point to the end of the line.
internal_buf_len	A pointer to the length of the buffer containing user input to the terminal. Will be adjusted to contain the length of the history entry being written to the buffer.

6.64.1.2 hist_next_frame()

```
char * hist_next_frame ( )
```

Requests a buffer to write user input to that will become the most recent entry in the user's command history.

Returns

A pointer to the first slot in a character buffer representing the next entry in the user's command history.

6.64.1.3 hist_rewind()

```
void hist_rewind (
             char * internal_buf,
             int * internal_index,
             int * internal_buf_len )
```

Moves backwards 1 entry in the user's command history.

Parameters

internal_buf	The buffer managed by low-level read operations containing user input from the terminal. Contents will be overwritten with the previous entry in the user's command history.
internal_index	A pointer to the position of the cursor, managed by low-level read operations. The cursor position will be adjusted to point to the end of the line.
internal_buf_len	A pointer to the length of the buffer containing user input to the terminal. Will be adjusted to contain the length of the history entry being written to the buffer.

6.65 history.h

Go to the documentation of this file.

```
#ifndef HISTORY_H
#define HISTORY_H

void hist_rewind(char *, int *, int *);
void hist_forward(char *, int *, int *);
char *hist_next_frame();

#endif
```

6.66 /home/maximillian/Desktop/MAMA/term/memory_↩ management/mm.c File Reference

```
#include "mm.h"
#include <term/utils.h>
```

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

Functions

- int initHeap (u32int size)

  Allocate all memory available for the MPX.
- u32int allocateMemory (u32int size)

  Allocate additional memory from the heap.
- void removeFMCB (cmcb_s ∗cmcb)
- void removeAMCB (cmcb_s ∗cmcb)
- void insertAMCB (cmcb_s ∗mcb)
- void insertFMCB (cmcb_s ∗mcb)
- int showAllocated (char ∗discard)

  Shows addresses and block size of all blocks in allocated list.
- int freeMemory (void ∗addr)

  Free a block of memory.
- int isEmpty (char ∗p)
- int showFree (char ∗p)

  Shows the addresses and block size of all block in free list.

Variables

- u32int start_addr

  Start address of the heap.
- mcb_queue_s allocated
- mcb_queue_s free
- mcb_queue_s ∗ amcb = &allocated

  Allocated Memory Control List.
- mcb_queue_s ∗ fmcb = &free

  Free Memory Control List.

6.66.1 Function Documentation

6.66.1.1 allocateMemory()

```
u32int allocateMemory (
            u32int size )
```

Allocate additional memory from the heap.

Allocates additional memory from the heap in a first-fit method.

@params size Amount of bytes to be allocated from the heap

Returns

Returns 0 upon success, -1 otherwise

6.66.1.2 freeMemory()

```
int freeMemory (
            void * addr )
```

Free a block of memory.

Frees a particular block of memory that was previously allocated. Searches for the block of memory, removes it from the allocated list and places it into the free list. If there are any adjacent blocks then merge.

@params addr Address of the block that will be free

Returns

Returns 0 upon success, -1 otherwise

6.66.1.3 initHeap()

```
int initHeap (
            u32int size )
```

Allocate all memory available for the MPX.

Allocates memory for both CMBC and LMCB. This will put a CMCB at the top of the heap and a LMCB at the bottom of the heap, both of type free. This method also intializes the free and allocated lists

Parameters

size	Size that will be allocated for the heap in bytes

Returns

Return 0 upon success, -1 otherwise

6.66.1.4 insertAMCB()

```
void insertAMCB (
            cmcb_s * mcb )
```

6.66.1.5 insertFMCB()

```
void insertFMCB (
            cmcb_s * mcb )
```

6.66.1.6 isEmpty()

```
int isEmpty (
             char * p )
```

6.66.1.7 removeAMCB()

```
void removeAMCB (
             cmcb_s * cmcb )
```

6.66.1.8 removeFMCB()

```
void removeFMCB (
             cmcb_s * cmcb )
```

6.66.1.9 showAllocated()

```
int showAllocated (
             char * discard )
```

Shows addresses and block size of all blocks in allocated list.

Traverses the allocated list and shows the addresses and the size of the block. Shown in the order of address.

6.66.1.10 showFree()

```
int showFree (
             char * p )
```

Shows the addresses and block size of all block in free list.

Traverses the free list and shows the addresses and the size of the block. Shown in the order of address.

6.66.2 Variable Documentation

6.66.2.1 allocated

```
mcb_queue_s allocated
```

6.66.2.2 amcb

```
mcb_queue_s* amcb = &allocated
```

Allocated Memory Control List.

6.66.2.3 fmcb

```
mcb_queue_s* fmcb = &free
```

Free Memory Control List.

6.66.2.4 free

```
mcb_queue_s free
```

6.66.2.5 start_addr

```
u32int start_addr
```

Start address of the heap.

6.67 /home/maximillian/Desktop/MAMA/term/memory_↩ management/mm.h File Reference

Classes

- struct cmcb_s

  Complete Memory Control Block (CMBC)
- struct mcb_queue_s

  "Master" controller of the MCB queue

Typedefs

- typedef struct cmcb_s cmcb_s

  Complete Memory Control Block (CMBC)
- typedef struct mcb_queue_s mcb_queue_s

  "Master" controller of the MCB queue

Enumerations

- enum mcb_state_e { ALLOCATED , FREE }

  Indicates the type of CMCB.

Functions

- int initHeap (u32int size)

  Allocate all memory available for the MPX.
- u32int allocateMemory (u32int size)

  Allocate additional memory from the heap.
- int freeMemory (void ∗addr)

  Free a block of memory.
- int showAllocated (char ∗)

  Shows addresses and block size of all blocks in allocated list.
- int showFree (char ∗p)

  Shows the addresses and block size of all block in free list.
- int isEmpty ()

  Does the heap only contain free memory.
- void removeFMCB (cmcb_s ∗mcb)
- void removeAMCB (cmcb_s ∗cmcb)
- void insertAMCB (cmcb_s ∗mcb)
- void insertFMCB (cmcb_s ∗mcb)

6.67.1 Typedef Documentation

6.67.1.1 cmcb_s

```
typedef struct cmcb_s cmcb_s
```

Complete Memory Control Block (CMBC)

6.67.1.2 mcb_queue_s

```
typedef struct mcb_queue_s mcb_queue_s
```

"Master" controller of the MCB queue

6.67.2 Enumeration Type Documentation

6.67.2.1 mcb_state_e

```
enum mcb_state_e
```

Indicates the type of CMCB.

Enumerator

ALLOCATED	Allocated CMCB.
FREE	Freed CMCB.

6.67.3 Function Documentation

6.67.3.1 allocateMemory()

```
u32int allocateMemory (
            u32int size )
```

Allocate additional memory from the heap.

Allocates additional memory from the heap in a first-fit method.

@params size Amount of bytes to be allocated from the heap

Returns

Returns 0 upon success, -1 otherwise

6.67.3.2 freeMemory()

```
int freeMemory (
            void * addr )
```

Free a block of memory.

Frees a particular block of memory that was previously allocated. Searches for the block of memory, removes it from the allocated list and places it into the free list. If there are any adjacent blocks then merge.

@params addr Address of the block that will be free

Returns

Returns 0 upon success, -1 otherwise

6.67.3.3 initHeap()

```
int initHeap (
            u32int size )
```

Allocate all memory available for the MPX.

Allocates memory for both CMBC and LMCB. This will put a CMCB at the top of the heap and a LMCB at the bottom of the heap, both of type free. This method also intializes the free and allocated lists

Parameters

size	Size that will be allocated for the heap in bytes

Returns

Return 0 upon success, -1 otherwise

6.67.3.4 insertAMCB()

```
void insertAMCB (
            cmcb_s * mcb )
```

6.67.3.5 insertFMCB()

```
void insertFMCB (
            cmcb_s * mcb )
```

6.67.3.6 isEmpty()

```
int isEmpty ( )
```

Does the heap only contain free memory.

Tells whether the heap contains only free memory (True) or not (false).

Returns

Returns 1 (True) if heap contains only free memory, Return 0 (False) if there is something within the heap.

6.67.3.7 removeAMCB()

```
void removeAMCB (
            cmcb_s * cmcb )
```

6.67.3.8 removeFMCB()

```
void removeFMCB (
            cmcb_s * mcb )
```

6.67.3.9 showAllocated()

```
int showAllocated (
            char * discard )
```

Shows addresses and block size of all blocks in allocated list.

Traverses the allocated list and shows the addresses and the size of the block. Shown in the order of address.

6.67.3.10 showFree()

```
int showFree (
            char * p )
```

Shows the addresses and block size of all block in free list.

Traverses the free list and shows the addresses and the size of the block. Shown in the order of address.

6.68 mm.h

Go to the documentation of this file.

```
#ifndef MM_H
#define MM_H

/*********************************************/
/*************** Structures ****************/
/*********************************************/

typedef enum {
    ALLOCATED,

    FREE
} mcb_state_e;

typedef struct cmcb_s { // This is 52 bytes long
    mcb_state_e type;

    u32int addr;

    u32int size;

    char name[32];

    struct cmcb_s * next;

    struct cmcb_s * prev;
} cmcb_s;

typedef struct mcb_queue_s {
    cmcb_s * mcbq_head;

    mcb_state_e mcb_queue_type;
} mcb_queue_s;

/*********************************************/
/************ Function Headers **************/
/*********************************************/
```

```
int initHeap(u32int size);

u32int allocateMemory(u32int size);

int freeMemory(void * addr);

int showAllocated(char *);

int showFree(char * p);

int isEmpty();

void removeFMCB(cmcb_s * mcb);

void removeAMCB(cmcb_s * cmcb);

void insertAMCB(cmcb_s * mcb);

void insertFMCB(cmcb_s * mcb);

#endif
```

6.69 /home/maximillian/Desktop/MAMA/term/cmds/pcb.c File Reference

6.70 /home/maximillian/Desktop/MAMA/term/pcb/pcb.c File Reference

```
#include "pcb.h"
#include <modules/mpx_supt.h>
#include <include/string.h>
#include <lib/out.h>
#include <term/utils.h>
#include <term/args.h>
#include <term/dispatch/context.h>
```

Functions

- void initPCB ()

  Initialize PCB Queue.
- pcb_t ∗ allocatePCB ()

  Allocate memory for a new PCB.
- int freePCB (pcb_t ∗pcb)

  Free's memory associated with PCB.
- pcb_t ∗ setupPCB (char ∗name, int process_class, int priority)

  Creates a PCB.
- pcb_t ∗ findPCB (char ∗name)

  Searches for PCB.
- int insertPCB (pcb_t ∗pcb)

  Insert PCB into queue.
- int removePCB (pcb_t ∗pcb)

  Removes PCB from Queue.
- int createPCB (char ∗args)

  Create a PCB.
- int setPriority (char ∗args)

  Set a new priority to a PCB.
- int showPCB (char ∗args)

Show informatino of PCB.
- int showReady (char ∗p)

  Show PCBs in ready queue.
- int showBlocked (char ∗args)

  Show PCBs in blocked queue.
- int showAll (char ∗args)

  Show all PCBs.
- int suspendPCB (char ∗args)

  Set PCB state to suspended.
- int resumePCB (char ∗args)

  Set PCB state to resume.
- int deletePCB (char ∗args)

  Delete PCB.
- int blockPCB (char ∗args)

  Set PCB state to be blocked.
- int unblockPCB (char ∗name)

  Set PCB state to unblocked.
- int resumeAll (char ∗p)

  Resume all suspended processes.
- int isSystemProcess (char ∗name)

  Checks whether specified process is a system process or not.

Variables

- pcb_queue_t p_queue
- pcb_queue_t f_queue
- pcb_queue_t ∗ priority_queue = &p_queue
- pcb_queue_t ∗ fifo_queue = &f_queue

6.70.1 Function Documentation

6.70.1.1 allocatePCB()

```
pcb_t * allocatePCB ( )
```

Allocate memory for a new PCB.

Allocates memory for a new PCB in the stack and performs actions to initialize PCB

Returns

Pointer to newly created PCB, NULL otherwise

6.70.1.2 blockPCB()

```
int blockPCB (
            char * name )
```

Set PCB state to be blocked.

Find the PCB name in queue and sets its state to blocked and reinserts it into the appropriate queue.

Parameters

name	Name of PCB to block

6.70.1.3 createPCB()

```
int createPCB (
            char * user_input )
```

Create a PCB.

Creates a new, unique PCB in memory.

Parameters

name	Give name of the PCB
process_class	The type of process class that will be used
priority	Priority of the PCB

Returns

Returns 0 upon success, 1 upon error

Parse the user input

Error Handling

6.70.1.4 deletePCB()

```
int deletePCB (
            char * name )
```

Delete PCB.

Will remove a PCB from the appropriate queue and free all associated memory. Will find the PCB in the queue, unlink it and free it.

Parameters

name	Name of the PCB to delete

Returns

Return 0 upon success, 1 upon failure

6.70.1.5 findPCB()

```
pcb_t * findPCB (
            char * name )
```

Searches for PCB.

Given a PCB name, will search all queues for a process.

Parameters

name	Name of the PCB being searched

Returns

Returns pointer to PCB upon success, NULL if PCB was not found

6.70.1.6 freePCB()

```
int freePCB (
            pcb_t * freed_pcb )
```

Free's memory associated with PCB.

Free's the memory associated with the PCB such as the stack and the PCB itself

Parameters

freed_pcb	Pointer to the PCB being freed

Returns

Returns 1 upon success, 0 upon error

6.70.1.7 initPCB()

```
void initPCB ( )
```

Initialize PCB Queue.

Initialize the PCB queue's by assigning values for the two queues that exist. This method is called upon startup in the commhand

6.70.1.8 insertPCB()

```
int insertPCB (
            pcb_t * pcb )
```

Insert PCB into queue.

Inserts a PCB into the appropriate queue

Parameters

pcb	Pointer to the PCB being inserted

Returns

0 on success, 1 on error

6.70.1.9 isSystemProcess()

```
int isSystemProcess (
            char * name )
```

Checks whether specified process is a system process or not.

Checks if the user supplied process name is a system process or an application.

Parameters

name	Name of the process

Returns

Returns 1 if the process is a system process, 0 if the process is an application

6.70.1.10 removePCB()

```
int removePCB (
            pcb_t * pcb )
```

Removes PCB from Queue.

Removes specified PCB from queue it is stored in.

Parameters

pcb	Pointer to the PCB being removed

Returns

Returns 1 upon success, 0 upon error

6.70.1.11 resumeAll()

```
int resumeAll (
            char * p )
```

Resume all suspended processes.

Iterates through READY queue and sets the state of the each PCB to READY

Parameters

p	Empty params

Returns

Returns 0 upon success, -1 otherwise.

6.70.1.12 resumePCB()

```
int resumePCB (
            char * name )
```

Set PCB state to resume.

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

Parameters

name	Name of PCB to resume

Returns

Returns 0 upon success, 1 upon error

6.70.1.13 setPriority()

```
int setPriority (
            char * args )
```

Set a new priority to a PCB.

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

Parameters

args	Name of the PCB and new priority (PCB_NAME.PRIORITY)

Returns

Returns 0 upon success, 1 upon error

Parse the user input

Error Handling

6.70.1.14 setupPCB()

```
pcb_t * setupPCB (
            char * name,
            int process_class,
            int priority )
```

Creates a PCB.

Allocates and fill memory associated with the PCB being created. This is accomplished by calling allocatePCB() to initialize the memory and the fills the data with the parameters.

Parameters

name	Name of the PCB
process_class	Type of process being created
priority	The priority of the PCB being created

Returns

Returns pointer to PCB upon success, NULL otherwise

6.70.1.15 showAll()

```
int showAll (
            char * args )
```

Show all PCBs.

Display information for each PCB in the ready and blocked queue. The information that is displayed is: Process Name, Class, State, Suspended Status, Priority.

@params args Empty params

Returns

Returns 0 upon success, 1 upon error

6.70.1.16 showBlocked()

```
int showBlocked (
            char * args )
```

Show PCBs in blocked queue.

Display information for each PCB in the blocked queue. The information that is displayed is: Process Name, Class, State, Suspended Status, Priority.

6.70.1.17 showPCB()

```
int showPCB (
            char * name )
```

Show informatino of PCB.

Display information of the PCB. The information that is displayed is: Process Name, Class, State, Suspended Status and Priority

Parameters

name	Name of PCB to have its information displayed

6.70.1.18 showReady()

```
int showReady (
            char * p )
```

Show PCBs in ready queue.

Display information for each PCB in the ready queue. The information that is displayed is: Process Name, Class, State, Suspended Status, Priority

Parameters

p	Empty parameters.

Returns

0 upon success, 1 upon failure

6.70.1.19 suspendPCB()

```
int suspendPCB (
            char * name )
```

Set PCB state to suspended.

Places a PCB state into suspended and reinserts into appropriate queue

Parameters

name	Name of PCB to suspend

Returns

Returns 0 upon success, 1 upon error

6.70.1.20 unblockPCB()

```
int unblockPCB (
              char * name )
```

Set PCB state to unblocked.

Sets PCB state into unblocked and reinserts it into the appropriate queue

Parameters

name	Name of the PCB to unblock

Returns

Returns 0 upon success, 1 upon error

6.70.2 Variable Documentation

6.70.2.1 f_queue

```
pcb_queue_t f_queue
```

6.70.2.2 fifo_queue

```
pcb_queue_t* fifo_queue = &f_queue
```

6.70.2.3 p_queue

```
pcb_queue_t p_queue
```

6.70.2.4 priority_queue

```
pcb_queue_t* priority_queue = &p_queue
```

6.71 /home/maximillian/Desktop/MAMA/term/pcb/pcb.h File Reference

Classes

- struct pcb_t

  Process Control Block Structure.
- struct pcb_node_t

  Individual PCB nodes. Each PCB is associated with one node.
- struct pcb_queue

  "Master" controller of the PCB queue

Macros

- #define MAX_STACK_SIZE 1024

  The maximum size the stack can be. May change.
- #define MAX_PRIORITY 9

  Maximum priority a PCB can be given.
- #define MIN_PRIORITY 0

  Minimum priority a PCB can be given.
- #define MAX_NAME_SIZE 32

  Maximum name size that can be given to a pcb.

Typedefs

- typedef struct pcb_node_t pcb_node_t

  Individual PCB nodes. Each PCB is associated with one node.
- typedef struct pcb_queue pcb_queue_t

  "Master" controller of the PCB queue

Enumerations

- enum pcb_queue_order_t { PRIORITY , FIFO }

  Type of Queue Ordering.
- enum p_state_t {
  RUNNING , READY , BLOCKED , SUSPENDED_READY ,
  SUSPENDED_BLOCKED }

  Types of process states.
- enum p_protection_mode_t { DELETABLE , DELETABLE_WHEN_SUSPENDED , NOT_DELETABLE }

Functions

- void initPCB ()

    Initialize PCB Queue.
- pcb_t ∗ allocatePCB ()

    Allocate memory for a new PCB.
- int freePCB (pcb_t ∗freed_pcb)

    Free's memory associated with PCB.
- pcb_t ∗ setupPCB (char ∗name, int process_class, int priority)

    Creates a PCB.
- pcb_t ∗ findPCB (char ∗name)

    Searches for PCB.
- int insertPCB (pcb_t ∗pcb)

    Insert PCB into queue.
- int removePCB (pcb_t ∗pcb)

    Removes PCB from Queue.
- int createPCB (char ∗user_input)

    Create a PCB.
- int deletePCB (char ∗name)

    Delete PCB.
- int blockPCB (char ∗name)

    Set PCB state to be blocked.
- int unblockPCB (char ∗name)

    Set PCB state to unblocked.
- int suspendPCB (char ∗name)

    Set PCB state to suspended.
- int resumePCB (char ∗name)

    Set PCB state to resume.
- int setPriority (char ∗args)

    Set a new priority to a PCB.
- int showPCB (char ∗name)

    Show informatino of PCB.
- int showReady (char ∗p)

    Show PCBs in ready queue.
- int showBlocked (char ∗args)

    Show PCBs in blocked queue.
- int showAll (char ∗args)

    Show all PCBs.
- int resumeAll (char ∗p)

    Resume all suspended processes.
- int isSystemProcess (char ∗name)

    Checks whether specified process is a system process or not.

6.71.1 Macro Definition Documentation

6.71.1.1 MAX_NAME_SIZE

```
#define MAX_NAME_SIZE 32
```

Maximum name size that can be given to a pcb.

6.71.1.2 MAX_PRIORITY

```
#define MAX_PRIORITY 9
```

Maximum priority a PCB can be given.

6.71.1.3 MAX_STACK_SIZE

```
#define MAX_STACK_SIZE 1024
```

The maximum size the stack can be. May change.

6.71.1.4 MIN_PRIORITY

```
#define MIN_PRIORITY 0
```

Minimum priority a PCB can be given.

6.71.2 Typedef Documentation

6.71.2.1 pcb_node_t

```
typedef struct pcb_node_t pcb_node_t
```

Individual PCB nodes. Each PCB is associated with one node.

6.71.2.2 pcb_queue_t

```
typedef struct pcb_queue pcb_queue_t
```

"Master" controller of the PCB queue

6.71.3 Enumeration Type Documentation

6.71.3.1 p_protection_mode_t

```
enum p_protection_mode_t
```

Enumerator

DELETABLE	
DELETABLE_WHEN_SUSPENDED	
NOT_DELETABLE	

6.71.3.2 p_state_t

```
enum p_state_t
```

Types of process states.

Enumerator

RUNNING	Running State.
READY	Ready State.
BLOCKED	Blocked State.
SUSPENDED_READY	Suspended Ready State.
SUSPENDED_BLOCKED	Suspended Blocked State.

6.71.3.3 pcb_queue_order_t

```
enum pcb_queue_order_t
```

Type of Queue Ordering.

Enumerator

PRIORITY	Priority Queue (Ready)
FIFO	FIFO Queue (Blocked)

6.71.4 Function Documentation

6.71.4.1 allocatePCB()

```
pcb_t * allocatePCB ( )
```

Allocate memory for a new PCB.

Allocates memory for a new PCB in the stack and performs actions to initialize PCB

Returns

Pointer to newly created PCB, NULL otherwise

6.71.4.2 blockPCB()

```
int blockPCB (
            char * name )
```

Set PCB state to be blocked.

Find the PCB name in queue and sets its state to blocked and reinserts it into the appropriate queue.

Parameters

name	Name of PCB to block

6.71.4.3 createPCB()

```
int createPCB (
            char * user_input )
```

Create a PCB.

Creates a new, unique PCB in memory.

Parameters

name	Give name of the PCB
process_class	The type of process class that will be used
priority	Priority of the PCB

Returns

Returns 0 upon success, 1 upon error

Parse the user input

Error Handling

6.71.4.4 deletePCB()

```
int deletePCB (
            char * name )
```

Delete PCB.

Will remove a PCB from the appropriate queue and free all associated memory. Will find the PCB in the queue, unlink it and free it.

Parameters

name	Name of the PCB to delete

Returns

Return 0 upon success, 1 upon failure

6.71.4.5 findPCB()

```
pcb_t * findPCB (
            char * name )
```

Searches for PCB.

Given a PCB name, will search all queues for a process.

Parameters

name	Name of the PCB being searched

Returns

Returns pointer to PCB upon success, NULL if PCB was not found

6.71.4.6 freePCB()

```
int freePCB (
            pcb_t * freed_pcb )
```

Free's memory associated with PCB.

Free's the memory associated with the PCB such as the stack and the PCB itself

Parameters

freed_pcb	Pointer to the PCB being freed

Returns

Returns 1 upon success, 0 upon error

6.71.4.7 initPCB()

```
void initPCB ( )
```

Initialize PCB Queue.

Initialize the PCB queue's by assigning values for the two queues that exist. This method is called upon startup in the commhand

6.71.4.8 insertPCB()

```
int insertPCB (
            pcb_t * pcb )
```

Insert PCB into queue.

Inserts a PCB into the appropriate queue

Parameters

pcb	Pointer to the PCB being inserted

Returns

0 on success, 1 on error

6.71.4.9 isSystemProcess()

```
int isSystemProcess (
            char * name )
```

Checks whether specified process is a system process or not.

Checks if the user supplied process name is a system process or an application.

Parameters

name	Name of the process

Returns

Returns 1 if the process is a system process, 0 if the process is an application

6.71.4.10 removePCB()

```
int removePCB (
            pcb_t * pcb )
```

Removes PCB from Queue.

Removes specified PCB from queue it is stored in.

Parameters

pcb	Pointer to the PCB being removed

Returns

Returns 1 upon success, 0 upon error

6.71.4.11 resumeAll()

```
int resumeAll (
            char * p )
```

Resume all suspended processes.

Iterates through READY queue and sets the state of the each PCB to READY

Parameters

p	Empty params

Returns

Returns 0 upon success, -1 otherwise.

6.71.4.12 resumePCB()

```
int resumePCB (
            char * name )
```

Set PCB state to resume.

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

Parameters

name	Name of PCB to resume

Returns

Returns 0 upon success, 1 upon error

6.71.4.13 setPriority()

```
int setPriority (
            char * args )
```

Set a new priority to a PCB.

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

Parameters

args	Name of the PCB and new priority (PCB_NAME.PRIORITY)

Returns

Returns 0 upon success, 1 upon error

Parse the user input

Error Handling

6.71.4.14 setupPCB()

```
pcb_t * setupPCB (
            char * name,
            int process_class,
            int priority )
```

Creates a PCB.

Allocates and fill memory associated with the PCB being created. This is accomplished by calling allocatePCB() to initialize the memory and the fills the data with the parameters.

Parameters

name	Name of the PCB
process_class	Type of process being created
priority	The priority of the PCB being created

Returns

Returns pointer to PCB upon success, NULL otherwise

6.71.4.15 showAll()

```
int showAll (
            char * args )
```

Show all PCBs.

Display information for each PCB in the ready and blocked queue. The information that is displayed is: Process Name, Class, State, Suspended Status, Priority.

@params args Empty params

Returns

Returns 0 upon success, 1 upon error

6.71.4.16 showBlocked()

```
int showBlocked (
            char * args )
```

Show PCBs in blocked queue.

Display information for each PCB in the blocked queue. The information that is displayed is: Process Name, Class, State, Suspended Status, Priority.

6.71.4.17 showPCB()

```
int showPCB (
            char * name )
```

Show informatino of PCB.

Display information of the PCB. The information that is displayed is: Process Name, Class, State, Suspended Status and Priority

Parameters

name	Name of PCB to have its information displayed

6.71.4.18 showReady()

```
int showReady (
            char * p )
```

Show PCBs in ready queue.

Display information for each PCB in the ready queue. The information that is displayed is: Process Name, Class, State, Suspended Status, Priority

Parameters

p	Empty parameters.

Returns

0 upon success, 1 upon failure

6.71.4.19 suspendPCB()

```
int suspendPCB (
            char * name )
```

Set PCB state to suspended.

Places a PCB state into suspended and reinserts into appropriate queue

Parameters

name	Name of PCB to suspend

Returns

Returns 0 upon success, 1 upon error

6.71.4.20 unblockPCB()

```
int unblockPCB (
            char * name )
```

Set PCB state to unblocked.

Sets PCB state into unblocked and reinserts it into the appropriate queue

Parameters

name	Name of the PCB to unblock

Returns

Returns 0 upon success, 1 upon error

6.72 pcb.h

Go to the documentation of this file.

```c
#ifndef PCB_H
#define PCB_H

#define MAX_STACK_SIZE 1024

#define MAX_PRIORITY 9
#define MIN_PRIORITY 0

#define MAX_NAME_SIZE 32

/*********************************************/
/*************** Structures ***************/
/*********************************************/

typedef enum {
    PRIORITY,

    FIFO
} pcb_queue_order_t;

typedef enum {
    RUNNING,

    READY,

    BLOCKED,

    SUSPENDED_READY,

    SUSPENDED_BLOCKED
} p_state_t;

typedef enum {
    DELETABLE,
    DELETABLE_WHEN_SUSPENDED,
    NOT_DELETABLE
} p_protection_mode_t;

typedef struct {
    char pcb_name[32];          // Can change size in the future

    int pcb_process_class;     // I've decided that process class will be an int. SYS_PROCESS = 0,
    APPLICATION = 1

    int pcb_priority;

    p_state_t pcb_process_state;

    p_protection_mode_t pcb_protection_mode;

    unsigned char * pcb_stack_top;

    unsigned char * pcb_stack_bottom;
} pcb_t;

typedef struct pcb_node_t {
    struct pcb_node_t *pcbn_next_pcb;

    struct pcb_node_t *pcbn_prev_pcb;

    pcb_t *pcb;
} pcb_node_t;

typedef struct pcb_queue {
    int pcbq_count;

    pcb_node_t *pcbq_head;

    pcb_node_t *pcbq_tail;

    pcb_queue_order_t queue_order;
} pcb_queue_t;


```

```

/*********************************************/
/************ Function Headers **************/
/*********************************************/

void initPCB();

pcb_t * allocatePCB();

int freePCB(pcb_t * freed_pcb);

pcb_t * setupPCB(char * name, int process_class, int priority);

pcb_t * findPCB(char * name);

int insertPCB(pcb_t * pcb);

int removePCB(pcb_t * pcb);


int createPCB(char * user_input);

int deletePCB(char * name);

int blockPCB(char * name);

int unblockPCB(char * name);

int suspendPCB(char * name);

int resumePCB(char * name);

int setPriority(char * args);

int showPCB(char * name);

int showReady(char * p);

int showBlocked(char * args);

int showAll(char * args);

/********************************************************/
/******************** R4 Stuff Here *******************/
/********************************************************/

int resumeAll(char * p);

/********************************************************/
/******************** R6 Stuff Here *******************/
/********************************************************/

int isSystemProcess(char * name);


#endif
```

6.73 /home/maximillian/Desktop/MAMA/term/syntax.c File Reference

```
#include "syntax.h"
#include "utils.h"
```

Functions

- int changes_state (char, enum SyntaxState, enum SyntaxState ∗)
- enum SyntaxState get_state (char c, enum SyntaxState cur_state)

6.73.1 Function Documentation

6.73.1.1 changes_state()

```
int changes_state (
            char c,
            enum SyntaxState cur_state,
            enum SyntaxState * next_state )
```

6.73.1.2 get_state()

```
enum SyntaxState get_state (
            char c,
            enum SyntaxState cur_state )
```

6.74 /home/maximillian/Desktop/MAMA/term/syntax.h File Reference

Enumerations

- enum SyntaxState {
  CMD_NAME_OR_LEADING_WHITESPACE , CMD_NAME , PARAM_NAME , PARAM_VALUE ,
  DOUBLE_QUOTE_STRING , DOUBLE_QUOTE_STRING_END_QUOTE , SINGLE_QUOTE_STRING ,
  SINGLE_QUOTE_STRING_END_QUOTE ,
  END_OF_INPUT , DEFAULT }

Functions

- enum SyntaxState get_state (char, enum SyntaxState)
- int changes_state (char, enum SyntaxState, enum SyntaxState ∗)

6.74.1 Enumeration Type Documentation

6.74.1.1 SyntaxState

```
enum SyntaxState
```

Enumerator

CMD_NAME_OR_LEADING_WHITESPACE	
CMD_NAME	
PARAM_NAME	
PARAM_VALUE	
DOUBLE_QUOTE_STRING	
DOUBLE_QUOTE_STRING_END_QUOTE	
SINGLE_QUOTE_STRING	
SINGLE_QUOTE_STRING_END_QUOTE	
END_OF_INPUT	
DEFAULT	

6.74.2 Function Documentation

6.74.2.1 changes_state()

```
int changes_state (
            char c,
            enum SyntaxState,
            enum SyntaxState * next_state )
```

6.74.2.2 get_state()

```
enum SyntaxState get_state (
            char c,
            enum SyntaxState )
```

6.75 syntax.h

Go to the documentation of this file.

```
#ifndef SYNTAX_H
#define SYNTAX_H

enum SyntaxState {
    CMD_NAME_OR_LEADING_WHITESPACE,
    CMD_NAME,
    PARAM_NAME,
    PARAM_VALUE,
    DOUBLE_QUOTE_STRING,
    DOUBLE_QUOTE_STRING_END_QUOTE,
    SINGLE_QUOTE_STRING,
    SINGLE_QUOTE_STRING_END_QUOTE,
    END_OF_INPUT,
    DEFAULT
};

enum SyntaxState get_state(char, enum SyntaxState);
int changes_state(char, enum SyntaxState, enum SyntaxState *);

#endif
```

6.76 /home/maximillian/Desktop/MAMA/term/utils.c File Reference

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

Functions

- int is_name_char (char c)

  Returns whether or not the specified character is a valid character in an identifier, such as a command or argument name.
- void skip_ws (char **c)

  Moves the specified pointer to a character buffer forward until it points to the next non-whitespace character.

6.76.1 Function Documentation

6.76.1.1 is_name_char()

```
int is_name_char (
             char c )
```

Returns whether or not the specified character is a valid character in an identifier, such as a command or argument name.

Parameters

c	The character to test.

Returns

True if the specified character c is valid in an identifier, false otherwise.

6.76.1.2 skip_ws()

```
void skip_ws (
             char ** c )
```

Moves the specified pointer to a character buffer forward until it points to the next non-whitespace character.

Parameters

c	A pointer to a pointer to an entry in a character buffer. Will be modified to point to the next non-whitespace character in the buffer.

6.77 /home/maximillian/Desktop/MAMA/term/utils.h File Reference

Functions

- int is_name_char (char)

  Returns whether or not the specified character is a valid character in an identifier, such as a command or argument name.
- void skip_ws (char ∗∗)

  Moves the specified pointer to a character buffer forward until it points to the next non-whitespace character.

6.77.1 Function Documentation

6.77.1.1 is_name_char()

```
int is_name_char (
            char c )
```

Returns whether or not the specified character is a valid character in an identifier, such as a command or argument name.

Parameters

c	The character to test.

Returns

True if the specified character c is valid in an identifier, false otherwise.

6.77.1.2 skip_ws()

```
void skip_ws (
            char ** c )
```

Moves the specified pointer to a character buffer forward until it points to the next non-whitespace character.

Parameters

c	A pointer to a pointer to an entry in a character buffer. Will be modified to point to the next non-whitespace character in the buffer.

6.78 utils.h

Go to the documentation of this file.

```
#ifndef UTILS_H
#define UTILS_H

int is_name_char(char);
void skip_ws(char **);

#endif
```

6.79 /home/maximillian/Desktop/MAMA/term/cmds/clear.c File Reference

```
#include <term/visuals/cursor.h>
#include <term/visuals/clear.h>
```

Functions

- int cmd_clear (char ∗args)

6.79.1 Function Documentation

6.79.1.1 cmd_clear()

```
int cmd_clear (
             char * args )
```

6.80 /home/maximillian/Desktop/MAMA/term/visuals/clear.c File Reference

```
#include <lib/out.h>
```

Functions

- void display_clear ()

6.80.1 Function Documentation

6.80.1.1 display_clear()

```
void display_clear ( )
```

6.81 /home/maximillian/Desktop/MAMA/term/visuals/clear.h File Reference

Functions

- void display_clear ()

6.81.1 Function Documentation

6.81.1.1 display_clear()

```
void display_clear ( )
```

6.82 clear.h

Go to the documentation of this file.

```
void display_clear();
```

6.83 /home/maximillian/Desktop/MAMA/term/visuals/colorize.c File Reference

```
#include <lib/out.h>
```

Macros

- #define START_SEQ "\e["

Enumerations

- enum Color {
  BLACK , RED , GREEN , YELLOW ,
  BLUE , MAGENTA , CYAN , WHITE ,
  BLACK , RED , GREEN , YELLOW ,
  BLUE , MAGENTA , CYAN , WHITE }

Functions

- void print_color_code (enum Color color)

  Description: Prints part of the escape sequence needed to switch the foreground or background color to the specified color.
- void display_fg_color (enum Color color)

  Switches the text color in the terminal so that subsequent text written to the screen will be the specified color.
- void display_bg_color (enum Color color)

  Switches the background color in the terminal so that subsequent text written to the screen will appear on a backdrop of the specified color.
- void display_reset ()

  Resets any formatting so that subsequent text written to the screen will use the default appearance.
- void display_italicize ()

  Description: Causes subsequent text written to the screen to be displayed in italics.

6.83.1 Macro Definition Documentation

6.83.1.1 START_SEQ

```
#define START_SEQ "\e["
```

6.83.2 Enumeration Type Documentation

6.83.2.1 Color

```
enum Color
```

Enumerator

BLACK	
RED	
GREEN	
YELLOW	
BLUE	
MAGENTA	
CYAN	
WHITE	
BLACK	
RED	
GREEN	
YELLOW	
BLUE	
MAGENTA	
CYAN	
WHITE	

6.83.3 Function Documentation

6.83.3.1 display_bg_color()

```
void display_bg_color (
            enum Color color )
```

Switches the background color in the terminal so that subsequent text written to the screen will appear on a backdrop of the specified color.

Parameters

color	The color to switch to.

6.83.3.2 display_fg_color()

```
void display_fg_color (
            enum Color color )
```

Switches the text color in the terminal so that subsequent text written to the screen will be the specified color.

Parameters

color	The color to switch to.

6.83.3.3 display_italicize()

```
void display_italicize ( )
```

Description: Causes subsequent text written to the screen to be displayed in italics.

6.83.3.4 display_reset()

```
void display_reset ( )
```

Resets any formatting so that subsequent text written to the screen will use the default appearance.

6.83.3.5 print_color_code()

```
void print_color_code (
            enum Color color )
```

Description: Prints part of the escape sequence needed to switch the foreground or background color to the specified color.

Used internally by display_fg_color and display_bg_color.

Parameters

color	The color being switched to.

6.84 /home/maximillian/Desktop/MAMA/term/visuals/colorize.h File Reference

Enumerations

- enum Color {
  BLACK , RED , GREEN , YELLOW ,
  BLUE , MAGENTA , CYAN , WHITE ,
  BLACK , RED , GREEN , YELLOW ,
  BLUE , MAGENTA , CYAN , WHITE }

Functions

- void display_fg_color (enum Color)

Switches the text color in the terminal so that subsequent text written to the screen will be the specified color.

- void display_bg_color (enum Color)

  Switches the background color in the terminal so that subsequent text written to the screen will appear on a backdrop of the specified color.

- void display_italicize ()

  Description: Causes subsequent text written to the screen to be displayed in italics.

- void display_reset ()

  Resets any formatting so that subsequent text written to the screen will use the default appearance.

6.84.1 Enumeration Type Documentation

6.84.1.1 Color

```
enum Color
```

Enumerator

BLACK	
RED	
GREEN	
YELLOW	
BLUE	
MAGENTA	
CYAN	
WHITE	
BLACK	
RED	
GREEN	
YELLOW	
BLUE	
MAGENTA	
CYAN	
WHITE	

6.84.2 Function Documentation

6.84.2.1 display_bg_color()

```
void display_bg_color (
            enum Color color )
```

Switches the background color in the terminal so that subsequent text written to the screen will appear on a backdrop of the specified color.

Parameters

color	The color to switch to.

6.84.2.2 display_fg_color()

```
void display_fg_color (
            enum Color color )
```

Switches the text color in the terminal so that subsequent text written to the screen will be the specified color.

Parameters

color	The color to switch to.

6.84.2.3 display_italicize()

```
void display_italicize ( )
```

Description: Causes subsequent text written to the screen to be displayed in italics.

6.84.2.4 display_reset()

```
void display_reset ( )
```

Resets any formatting so that subsequent text written to the screen will use the default appearance.

6.85 colorize.h

Go to the documentation of this file.

```
#ifndef COLORIZE_H
#define COLORIZE_H

enum Color {
        BLACK,
        RED,
        GREEN,
        YELLOW,
        BLUE,
        MAGENTA,
        CYAN,
        WHITE
};

void display_fg_color(enum Color);
void display_bg_color(enum Color);
void display_italicize();
void display_reset();

#endif
```

6.86 /home/maximillian/Desktop/MAMA/term/visuals/cursor.c File Reference

```
#include <lib/out.h>
```

Functions

- void cursor_left (int steps)

  Moves the visual cursor to the left a specified number of steps.
- void cursor_right (int steps)

  Moves the visual cursor to the right a specified number of steps.
- void cursor_down (int steps)

  Moves the visual cursor down a specified number of steps.
- void cursor_up (int steps)

  Moves the visual cursor up a specified number of steps.
- void cursor_return ()

  Moves the visual cursor to the beginning of the line.

6.86.1 Function Documentation

6.86.1.1 cursor_down()

```
void cursor_down (
             int steps )
```

Moves the visual cursor down a specified number of steps.

Parameters

steps	The number of steps to move the cursor down.

6.86.1.2 cursor_left()

```
void cursor_left (
             int steps )
```

Moves the visual cursor to the left a specified number of steps.

Parameters

steps	The number of steps to move the cursor to the left.

6.86.1.3 cursor_return()

```
void cursor_return ( )
```

Moves the visual cursor to the beginning of the line.

6.86.1.4 cursor_right()

```
void cursor_right (
             int steps )
```

Moves the visual cursor to the right a specified number of steps.

Parameters

steps	The number of steps to move the cursor to the right.

6.86.1.5 cursor_up()

```
void cursor_up (
             int steps )
```

Moves the visual cursor up a specified number of steps.

Parameters

steps	The number of steps to move the cursor up.

6.87 /home/maximillian/Desktop/MAMA/term/visuals/cursor.h File Reference

Functions

- void cursor_left (int)
  Moves the visual cursor to the left a specified number of steps.
- void cursor_right (int)
  Moves the visual cursor to the right a specified number of steps.
- void cursor_up (int)
  Moves the visual cursor up a specified number of steps.

- void cursor_down (int)

  Moves the visual cursor down a specified number of steps.
- void cursor_return ()

  Moves the visual cursor to the beginning of the line.

6.87.1 Function Documentation

6.87.1.1 cursor_down()

```
void cursor_down (
             int steps )
```

Moves the visual cursor down a specified number of steps.

Parameters

steps	The number of steps to move the cursor down.

6.87.1.2 cursor_left()

```
void cursor_left (
             int steps )
```

Moves the visual cursor to the left a specified number of steps.

Parameters

steps	The number of steps to move the cursor to the left.

6.87.1.3 cursor_return()

```
void cursor_return ( )
```

Moves the visual cursor to the beginning of the line.

6.87.1.4 cursor_right()

```
void cursor_right (
             int steps )
```

Moves the visual cursor to the right a specified number of steps.

Parameters

steps	The number of steps to move the cursor to the right.

6.87.1.5 cursor_up()

```
void cursor_up (
             int steps )
```

Moves the visual cursor up a specified number of steps.

Parameters

steps	The number of steps to move the cursor up.

6.88 cursor.h

Go to the documentation of this file.

```
#ifndef CURSOR_H
#define CURSOR_H

void cursor_left(int);
void cursor_right(int);
void cursor_up(int);
void cursor_down(int);
void cursor_return();

#endif
```

6.89 /home/maximillian/Desktop/MAMA/term/visuals/hints.c File Reference

```
#include <lib/out.h>
#include "cursor.h"
```

Functions

- void hint_under_prompt (char ∗str, int len, int ret_index)

  Writes a line of text under the user's prompt in the terminal.

6.89.1 Function Documentation

6.89.1.1 hint_under_prompt()

```
void hint_under_prompt (
            char * str,
            int len,
            int ret_index )
```

Writes a line of text under the user's prompt in the terminal.

Recommended for providing hints or warnings to the user as they type.

Parameters

str	The text to write under the user's prompt.
len	The length of the text to write under the user's prompt.
ret_index	The position to return the user's cursor to after writing the text.

6.90 /home/maximillian/Desktop/MAMA/term/visuals/hints.h File Reference

Functions

- void hint_under_prompt (char ∗, int, int)

  Writes a line of text under the user's prompt in the terminal.

6.90.1 Function Documentation

6.90.1.1 hint_under_prompt()

```
void hint_under_prompt (
            char * str,
            int len,
            int ret_index )
```

Writes a line of text under the user's prompt in the terminal.

Recommended for providing hints or warnings to the user as they type.

Parameters

str	The text to write under the user's prompt.
len	The length of the text to write under the user's prompt.
ret_index	The position to return the user's cursor to after writing the text.

6.91 hints.h

Go to the documentation of this file.

```
#ifndef HINTS_H
#define HINTS_H

void hint_under_prompt(char *, int, int);

#endif
```

6.92 /home/maximillian/Desktop/MAMA/term/visuals/syntax_highlight.c File Reference

```
#include "../syntax.h"
#include "../syntax.c"
#include "syntax_highlight.h"
#include "../commhand.h"
#include "colorize.h"
#include "hints.c"
#include "../utils.c"
#include <include/string.h>
```

Functions

- void switch_to (enum SyntaxState, int, int)

  Whether or not syntax highlighting is enabled as the user types.
- void color_for (enum SyntaxState state)

  Prints the ANSI color code for the specified syntax state.
- void get_state_at (int index, int ∗index_of_state_in_record)

  Retrieves the index in the states and switch_indexes data structures corresponding to the specified cursor index.
- void syntax_init ()

  Initializes internal data structures needed for syntax highlighting.
- void syntax_enable_highlighting ()

  Enables syntax highlighting as the user types.
- void syntax_disable_highlighting ()

  Disables syntax highlighting as the user types.
- void syntax_handle_char (char c, int index)

  Adjusts the terminal color assuming the specified character will immediately be written to the screen at the specified index.

Variables

- enum SyntaxState states [MAX_SYNTAX_SWITCHES]
- int switch_indexes [MAX_SYNTAX_SWITCHES]

  Array of all the states the cursor has been in as the user has typed. Entries correspond to entries in switch_indexes.
- int newest_switch

  Array of indexes the cursor was at when the corresponding syntax state in states was switched to.
- int enabled = 0

  The largest and most recent valid index in states and switch_indexes.

6.92.1 Function Documentation

6.92.1.1 color_for()

```
void color_for (
             enum SyntaxState state )
```

Prints the ANSI color code for the specified syntax state.

Used internally by syntax_handle_char.

Parameters

state	The syntax state for which to print the correct color code to the terminal for.

6.92.1.2 get_state_at()

```
void get_state_at (
             int index,
             int * index_of_state_in_record )
```

Retrieves the index in the states and switch_indexes data structures corresponding to the specified cursor index.

Used internally by syntax_handle_char.

Parameters

index	The index of the cursor.
index_of_state_in_record	A pointer to the index in the states and switch_indexes data structures corresponding to the specified cursor index. Will be updated to point to the correct index in the data structures.

6.92.1.3 switch_to()

```
void switch_to (
             enum SyntaxState state,
             int index,
             int record_index )
```

Whether or not syntax highlighting is enabled as the user types.

Switches to the specified syntax state.

Used internally by syntax_handle_char.

Parameters

state	The syntax state being switched to.
index	The index in the user's input at which this switch occurs.
record_index	The index in the internal data structures states and switch_indexes at which to write this switch to.

6.92.1.4 syntax_disable_highlighting()

```
void syntax_disable_highlighting ( )
```

Disables syntax highlighting as the user types.

6.92.1.5 syntax_enable_highlighting()

```
void syntax_enable_highlighting ( )
```

Enables syntax highlighting as the user types.

6.92.1.6 syntax_handle_char()

```
void syntax_handle_char (
             char c,
             int index )
```

Adjusts the terminal color assuming the specified character will immediately be written to the screen at the specified index.

Parameters

c	The next character that will be output to the screen.
index	The index of the cursor.

6.92.1.7 syntax_init()

```
void syntax_init ( )
```

Initializes internal data structures needed for syntax highlighting.

6.92.2 Variable Documentation

6.92.2.1 enabled

```
int enabled = 0
```

The largest and most recent valid index in states and switch_indexes.

6.92.2.2 newest_switch

```
int newest_switch
```

Array of indexes the cursor was at when the corresponding syntax state in states was switched to.

6.92.2.3 states

```
enum SyntaxState states[MAX_SYNTAX_SWITCHES]
```

6.92.2.4 switch_indexes

```
int switch_indexes[MAX_SYNTAX_SWITCHES]
```

Array of all the states the cursor has been in as the user has typed. Entries correspond to entries in switch_indexes.

6.93 /home/maximillian/Desktop/MAMA/term/visuals/syntax_highlight.h File Reference

Macros

- #define MAX_SYNTAX_SWITCHES 40
- #define SYNTAX_COLOR_CMD_NAME CYAN
- #define SYNTAX_COLOR_PARAM_NAME MAGENTA
- #define SYNTAX_COLOR_PARAM_VALUE WHITE
- #define SYNTAX_COLOR_DOUBLE_QUOTE_STRING YELLOW
- #define SYNTAX_COLOR_SINGLE_QUOTE_STRING YELLOW
- #define SYNTAX_COLOR_DEFAULT WHITE

Functions

- void syntax_init ()

  Initializes internal data structures needed for syntax highlighting.
- void syntax_enable_highlighting ()

  Enables syntax highlighting as the user types.
- void syntax_disable_highlighting ()

  Disables syntax highlighting as the user types.
- void syntax_handle_char (char, int)

  Adjusts the terminal color assuming the specified character will immediately be written to the screen at the specified index.

6.93.1 Macro Definition Documentation

6.93.1.1 MAX_SYNTAX_SWITCHES

```
#define MAX_SYNTAX_SWITCHES 40
```

6.93.1.2 SYNTAX_COLOR_CMD_NAME

```
#define SYNTAX_COLOR_CMD_NAME CYAN
```

6.93.1.3 SYNTAX_COLOR_DEFAULT

```
#define SYNTAX_COLOR_DEFAULT WHITE
```

6.93.1.4 SYNTAX_COLOR_DOUBLE_QUOTE_STRING

```
#define SYNTAX_COLOR_DOUBLE_QUOTE_STRING YELLOW
```

6.93.1.5 SYNTAX_COLOR_PARAM_NAME

```
#define SYNTAX_COLOR_PARAM_NAME MAGENTA
```

6.93.1.6 SYNTAX_COLOR_PARAM_VALUE

```
#define SYNTAX_COLOR_PARAM_VALUE WHITE
```

6.93.1.7 SYNTAX_COLOR_SINGLE_QUOTE_STRING

```
#define SYNTAX_COLOR_SINGLE_QUOTE_STRING YELLOW
```

6.93.2 Function Documentation

6.93.2.1 syntax_disable_highlighting()

```
void syntax_disable_highlighting ( )
```

Disables syntax highlighting as the user types.

6.93.2.2 syntax_enable_highlighting()

```
void syntax_enable_highlighting ( )
```

Enables syntax highlighting as the user types.

6.93.2.3 syntax_handle_char()

```
void syntax_handle_char (
             char c,
             int index )
```

Adjusts the terminal color assuming the specified character will immediately be written to the screen at the specified index.

Parameters

c	The next character that will be output to the screen.
index	The index of the cursor.

6.93.2.4 syntax_init()

```
void syntax_init ( )
```

Initializes internal data structures needed for syntax highlighting.

6.94 syntax_highlight.h

Go to the documentation of this file.

```
#ifndef SYNTAX_HIGHLIGHT_H
#define SYNTAX_HIGHLIGHT_H

#define MAX_SYNTAX_SWITCHES 40

#define SYNTAX_COLOR_CMD_NAME CYAN
#define SYNTAX_COLOR_PARAM_NAME MAGENTA
#define SYNTAX_COLOR_PARAM_VALUE WHITE
#define SYNTAX_COLOR_DOUBLE_QUOTE_STRING YELLOW
#define SYNTAX_COLOR_SINGLE_QUOTE_STRING YELLOW
#define SYNTAX_COLOR_DEFAULT WHITE

void syntax_init();
void syntax_enable_highlighting();
void syntax_disable_highlighting();
void syntax_handle_char(char, int);

#endif
```

6.95 /home/maximillian/Desktop/MAMA/WhoDidWhat.md File Reference

Index