mpx-spring2017-npennf

NPE-NPX.R6.04282017

Generated by Doxygen 1.8.11

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 boot_sector Struct Reference

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

Data Fields

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

3.1.1 Detailed Description

Struct representing the boot sector

3.1.2 Field Documentation

3.1.2.1 uint8_t bootSig

3.1.2.2 uint16_t bytesPerSector

3.1.2.3 unsigned char fileSystemType[9]

3.1.2.4 uint8_t ignore_1[11]

3.1.2.5 uint8_t ignore_2[1]

3.1.2.6 uint8_t ignore_3[4]

3.1.2.7 uint8_t ignore_4[2]

3.1.2.8 uint8_t ignore_5[450]

3.1.2.9 uint16_t maxRootDirEntries

3.1.2.10 uint8_t numFATCopies

3.1.2.11 uint16_t numHeads

3.1.2.12 uint16_t numReservedSectors

3.1.2.13 uint16_t numSectors

3.1.2.14 uint32_t sectorCountFAT32

3.1.2.15 uint8_t sectorsPerCluster

3.1.2.16 uint16_t sectorsPerFAT

3.1.2.17 uint16_t sectorsPerTrack

3.1.2.18 uint32_t volId

3.1.2.19 unsigned char volName[12]

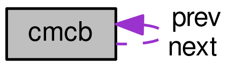
The documentation for this struct was generated from the following file:

- r6/fat.h

3.2 cmcb Struct Reference

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

Collaboration diagram for cmcb:



Data Fields

- int type
- void ∗ beginningAddr
- int size
- int memSize
- const char ∗ name
- struct cmcb ∗ next
- struct cmcb ∗ prev

3.2.1 Field Documentation

3.2.1.1 void∗ beginningAddr

3.2.1.2 int memSize

3.2.1.3 const char∗ name

3.2.1.4 struct cmcb∗ next

3.2.1.5 struct cmcb∗ prev

3.2.1.6 int size

3.2.1.7 int type

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

- include/mem/memoryControl.h

3.3 context Struct Reference

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

Data Fields

- u32int gs
- u32int fs
- u32int es
- u32int ds
- u32int edi
- u32int esi
- u32int ebp
- u32int esp
- u32int ebx
- u32int edx
- u32int ecx
- u32int eax
- u32int eip
- u32int cs
- u32int eflags

3.3.1 Field Documentation

3.3.1.1 u32int cs

3.3.1.2 u32int ds

3.3.1.3 u32int eax

3.3.1.4 u32int ebp

3.3.1.5 u32int ebx

3.3.1.6 u32int ecx

3.3.1.7 u32int edi

3.3.1.8 u32int edx

3.3.1.9 u32int eflags

3.3.1.10 u32int eip

3.3.1.11 u32int es

3.3.1.12 u32int esi

3.3.1.13 u32int esp

3.3.1.14 u32int fs

3.3.1.15 u32int gs

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

- include/modules/mpx_supt.h

3.4 date_time Struct Reference

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

Data Fields

- int sec
- int min
- int hour
- int day_w
- int day_m
- int day_y
- int mon
- int year

3.4.1 Detailed Description

Structure representing a date and time.

3.4.2 Field Documentation

3.4.2.1 int day_m

3.4.2.2 int day_w

3.4.2.3 int day_y

3.4.2.4 int hour

3.4.2.5 int min

3.4.2.6 int mon

3.4.2.7 int sec

3.4.2.8 int year

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

- include/time.h

3.5 dir_entry Struct Reference

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

Data Fields

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

3.5.1 Detailed Description

Struct representing a directory entry

3.5.2 Field Documentation

3.5.2.1 uint8_t attributes

3.5.2.2 uint16_t creationDate

3.5.2.3 uint16_t creationTime

3.5.2.4 unsigned char extension[4]

3.5.2.5 unsigned char filename[9]

3.5.2.6 uint32_t fileSize

3.5.2.7 uint16_t firstLogicalCluster

3.5.2.8 uint16_t ignore

3.5.2.9 uint16_t lastAccess

3.5.2.10 uint16_t lastWriteDate

3.5.2.11 uint16_t lastWriteTime

3.5.2.12 uint16_t reserved

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

- r6/fat.h

3.6 fat_tables Struct Reference

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

Data Fields

- int numEntries
- uint16_t ∗ fat1
- uint16_t ∗ fat2

3.6.1 Detailed Description

Struct containing the FAT table information

3.6.2 Field Documentation

3.6.2.1 uint16_t∗ fat1

3.6.2.2 uint16_t∗ fat2

3.6.2.3 int numEntries

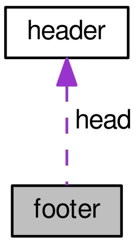
The documentation for this struct was generated from the following file:

- r6/fat.h

3.7 footer Struct Reference

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

Collaboration diagram for footer:



Data Fields

- header head

3.7.1 Detailed Description

Heap allocation footer.

3.7.2 Field Documentation

3.7.2.1 header head

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

- include/mem/heap.h

3.8 functionDef Struct Reference

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

Data Fields

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

3.8.1 Field Documentation

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

3.8.1.2 const char$*$ helpString

3.8.1.3 char$*$ name

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

- include/core/comHandler.h

3.9 gdt_descriptor_struct Struct Reference

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

Data Fields

- u16int limit
- u32int base

3.9.1 Field Documentation

3.9.1.1 u32int base

3.9.1.2 u16int limit

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

- include/core/tables.h

3.10 gdt_entry_struct Struct Reference

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

Data Fields

- u16int limit_low
- u16int base_low
- u8int base_mid
- u8int access
- u8int flags
- u8int base_high

3.10.1 Field Documentation

3.10.1.1 u8int access

3.10.1.2 u8int base_high

3.10.1.3 u16int base_low

3.10.1.4 u8int base_mid

3.10.1.5 u8int flags

3.10.1.6 u16int limit_low

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

- include/core/tables.h

3.11 header Struct Reference

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

Data Fields

- int size
- int index_id

3.11.1 Detailed Description

Heap allocation header.

3.11.2 Field Documentation

3.11.2.1 int index_id

3.11.2.2 int size

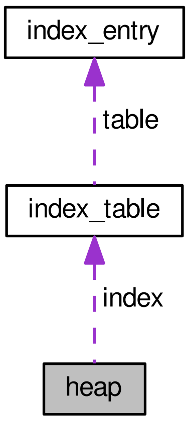
The documentation for this struct was generated from the following file:

- include/mem/heap.h

3.12 heap Struct Reference

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

Collaboration diagram for heap:



Data Fields

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

3.12.1 Detailed Description

Heap structure

3.12.2 Field Documentation

3.12.2.1 u32int base

3.12.2.2 index_table index

3.12.2.3 u32int max_size

3.12.2.4 u32int min_size

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

- include/mem/heap.h

3.13 idt_entry_struct Struct Reference

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

Data Fields

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

3.13.1 Field Documentation

3.13.1.1 u16int base_high

3.13.1.2 u16int base_low

3.13.1.3 u8int flags

3.13.1.4 u16int sselect

3.13.1.5 u8int zero

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

- include/core/tables.h

3.14 idt_struct Struct Reference

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

Data Fields

- u16int limit
- u32int base

3.14.1 Field Documentation

3.14.1.1 u32int base

3.14.1.2 u16int limit

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

- include/core/tables.h

3.15 index_entry Struct Reference

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

Data Fields

- int size
- int empty
- u32int block

3.15.1 Field Documentation

3.15.1.1 u32int block

3.15.1.2 int empty

3.15.1.3 int size

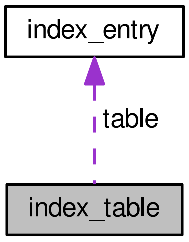
The documentation for this struct was generated from the following file:

- include/mem/heap.h

3.16 index_table Struct Reference

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

Collaboration diagram for index_table:



Data Fields

- index_entry table [TABLE_SIZE]
- int id

3.16.1 Detailed Description

Kernel heap index table.

3.16.2 Field Documentation

3.16.2.1 int id

3.16.2.2 index_entry table[TABLE_SIZE]

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

- include/mem/heap.h

3.17 lmcb Struct Reference

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

Data Fields

- int type
- int size
- int memSize

3.17.1 Field Documentation

3.17.1.1 int memSize

3.17.1.2 int size

3.17.1.3 int type

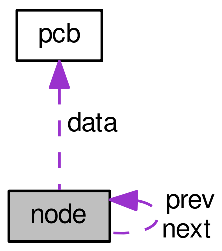
The documentation for this struct was generated from the following file:

- include/mem/memoryControl.h

3.18 node Struct Reference

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

Collaboration diagram for node:



Data Fields

- struct pcb ∗ data
- struct node ∗ next
- struct node ∗ prev

3.18.1 Detailed Description

The struct representing a node in a queue

3.18.2 Field Documentation

3.18.2.1 struct pcb∗ data

3.18.2.2 struct node∗ next

3.18.2.3 struct node∗ prev

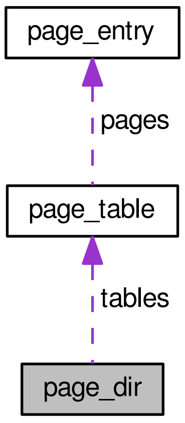
The documentation for this struct was generated from the following file:

- include/core/queue.h

3.19 page_dir Struct Reference

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

Collaboration diagram for page_dir:



Data Fields

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

3.19.1 Detailed Description

Page directory structure Limited to 1024 tables for now

3.19.2 Field Documentation

3.19.2.1 page_table∗ tables[1024]

3.19.2.2 u32int tables_phys[1024]

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

- include/mem/paging.h

3.20 page_entry Struct Reference

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

Data Fields

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

3.20.1 Detailed Description

Page entry structure Describes a single page in memory

3.20.2 Field Documentation

3.20.2.1 u32int accessed

3.20.2.2 u32int dirty

3.20.2.3 u32int frameaddr

3.20.2.4 u32int present

3.20.2.5 u32int reserved

3.20.2.6 u32int usermode

3.20.2.7 u32int writeable

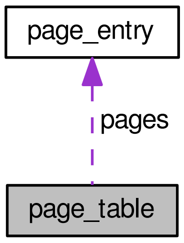
The documentation for this struct was generated from the following file:

- include/mem/paging.h

3.21 page_table Struct Reference

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

Collaboration diagram for page_table:



Data Fields

- page_entry pages [1024]

3.21.1 Detailed Description

Page table structure Contains 1024 pages/frames

3.21.2 Field Documentation

3.21.2.1 page_entry pages[1024]

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

- include/mem/paging.h

3.22 param Struct Reference

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

Data Fields

- int op_code
- int device_id

3.22.1 Field Documentation

3.22.1.1 int device_id

3.22.1.2 int op_code

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

- include/modules/mpx_supt.h

3.23 pcb Struct Reference

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

Data Fields

- char ∗ processName
- int processClass
- int priority
- int isSuspended
- int state
- unsigned char ∗ stackTop
- unsigned char ∗ stackBottom

3.23.1 Field Documentation

3.23.1.1 int isSuspended

3.23.1.2 int priority

3.23.1.3 int processClass

3.23.1.4 char∗ processName

3.23.1.5 unsigned char∗ stackBottom

3.23.1.6 unsigned char∗ stackTop

3.23.1.7 int state

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

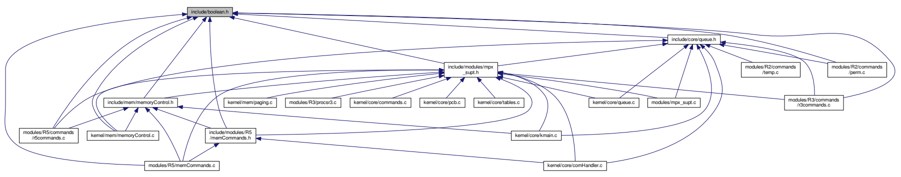
- include/core/pcb.h

Chapter 4

File Documentation

4.1 include/boolean.h File Reference

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



Enumerations

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

4.1.1 Enumeration Type Documentation

4.1.1.1 enum boolean

Enumerator

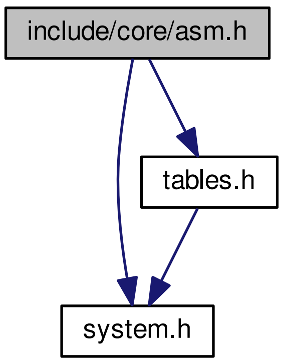
false

true

4.2 include/core/asm.h File Reference

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

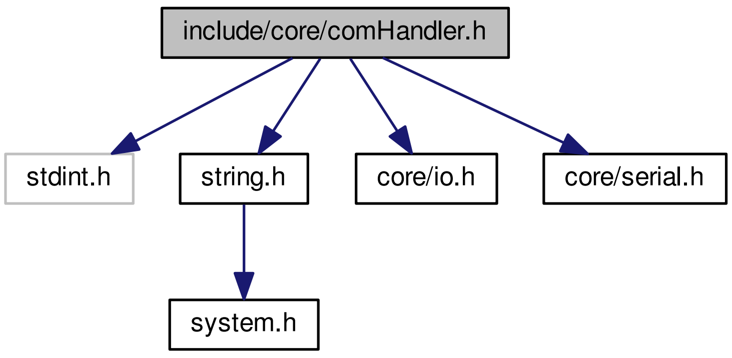
Include dependency graph for asm.h:



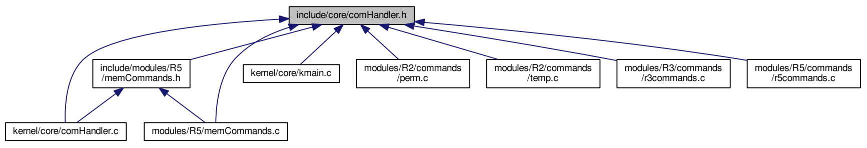
4.3 include/core/comHandler.h File Reference

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

Include dependency graph for comHandler.h:



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



Data Structures

- struct functionDef

Functions

- void addFunctionDef (char ∗name, const char ∗helpString, const char ∗(funcPointer)(char ∗∗args, int num←
  Args))
- functionDef getFunctionDef (char ∗name)
- const char ∗ getHelpString (char ∗name)
- char ∗ getComHistory (int isPrev)
- void addComHistory (char ∗newCom)
- void printStart ()
- void returnToInsertionPoint (int endIndex, int insertionIndex)
- void eraseCurrentRow (int endIndex, int insertionIndex)
- char ∗ getInput ()
- void executeCommand (char ∗commandString)
- void setupCommands ()
- void initCommandHandler ()

4.3.1 Function Documentation

4.3.1.1 void addComHistory (char ∗ *newCom*)

Helper function to add a command to the command history array

Parameters

newCom	string to add to the command history

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

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

Parameters

name	- string representation of the function

Parameters

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

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

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

Parameters

endIndex	- index of last char printed
insertionIndex	- index of where insertion point should be

4.3.1.4 void executeCommand (char ∗ *commandString*)

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

Parameters

commandString	string contianing the command name and any args

4.3.1.5 char∗ getComHistory (int *isPrev*)

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

Parameters

isPrev	integer denoting if to get the previous command

Returns

string of the command

4.3.1.6 functionDef getFunctionDef (char ∗ *name*)

Gets the functionDef struct corresponding to the name provided, returns a functionDef with null funcPointer if none are found

Parameters

name	- name of the functionDef

Returns

functionDef

4.3.1.7 const char∗ getHelpString (char ∗ *name*)

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

Parameters

name	- name associated witht he struct from which to get the help string

Returns

const char∗ help string

4.3.1.8 char∗ getInput ( )

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

Returns

string that was input

4.3.1.9 void initCommandHandler ( )

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

4.3.1.10 void printStart ( )

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

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

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

Parameters

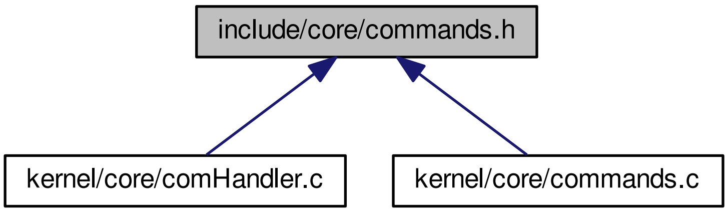
endIndex	- index of last char printed
insertionIndex	- index of where insertion point should be

4.3.1.12 void setupCommands ( )

Initialization function to add commands ot the functionDefs array

4.4 include/core/commands.h File Reference

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



Functions

- const char ∗ version (char ∗∗args, int numArgs)
- const char ∗ help (char ∗∗args, int numArgs)
- const char ∗ shutdown (char ∗∗args, int numArgs)
- const char ∗ date (char ∗∗args, int numArgs)

4.4.1 Function Documentation

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

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

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

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

The ISO-8601 formatted date

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

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

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

Parameters

args	The arguments to pass to the function

Returns

The ISO-8601 formatted date

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

Returns help for the specified commands.

Usage: help commandName

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

The help string

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

Shuts down the OS after asking for confirmation.

Usage: shutdown [–confirm]

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

True if shutdown was confirmed

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

Returns the current version of the OS.

Usage: version

Args: [no args] - Returns the version

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

The version of the OS.

Returns the current version of the OS.

Usage: version

Args: [no args] - Returns the version

Parameters

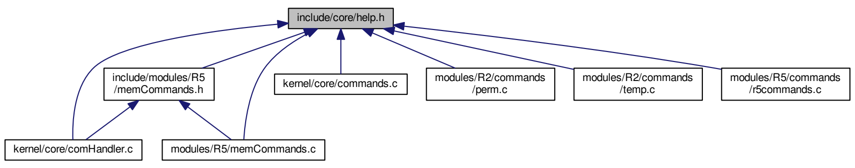
args	The arguments to pass to the function

Returns

The version of the OS.

4.5 include/core/help.h File Reference

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



Macros

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

4.5.1 Macro Definition Documentation

4.5.1.1 #define HELP_COMMAND_DATE

Value:

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

4.5.1.2 #define HELP_COMMAND_HELP

Value:

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

4.5.1.3 #define HELP_COMMAND_SHUTDOWN

Value:

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

4.5.1.4 #define HELP_COMMAND_VERSION

Value:

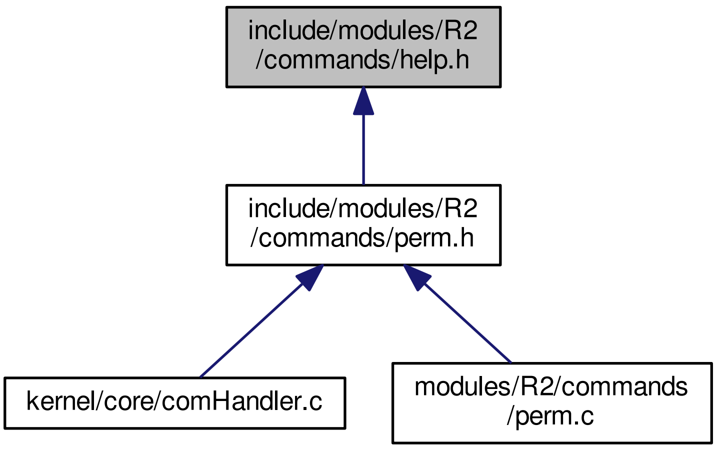
```
((const char*) \
    "Prints the current version of the OS.\n"\
    "\n"\
    "Usage: version\n"\
    "\n"\
    "Args:\n"\
    "     [no args] - Returns the version")
```

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

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

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

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


include/modules/R2 /commands/help.h
include/modules/R2 /commands/perm.h
kernel/core/comHandler.c
modules/R2/commands /perm.c

Macros

- #define HELP_R2_COMMAND_SPCB
- #define HELP_R2_COMMAND_RPCB
- #define HELP_R2_COMMAND_PPCB
- #define HELP_R2_COMMAND_SHOWPCB

4.6.1 Macro Definition Documentation

4.6.1.1 #define HELP_R2_COMMAND_PPCB

Value:

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

4.6.1.2 #define HELP_R2_COMMAND_RPCB

Value:

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

4.6.1.3 #define HELP_R2_COMMAND_SHOWPCB

Value:

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

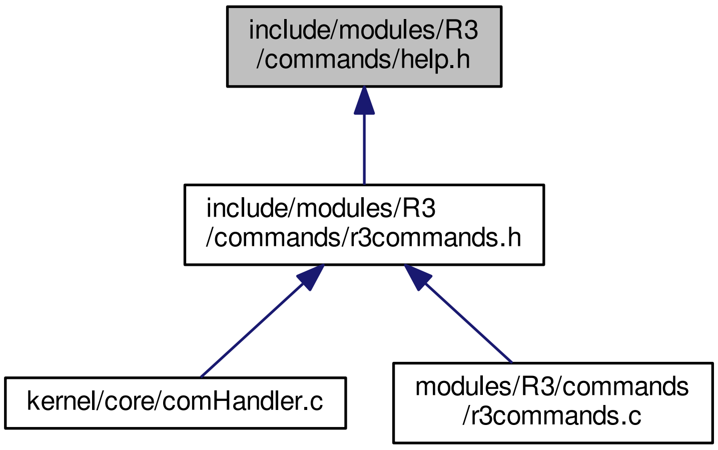
4.6.1.4 #define HELP_R2_COMMAND_SPCB

Value:

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

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

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



Macros

- #define HELP_R3_COMMAND_YIELD
- #define HELP_R3_COMMAND_LOAD

4.7.1 Macro Definition Documentation

4.7.1.1 #define HELP_R3_COMMAND_LOAD

Value:

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

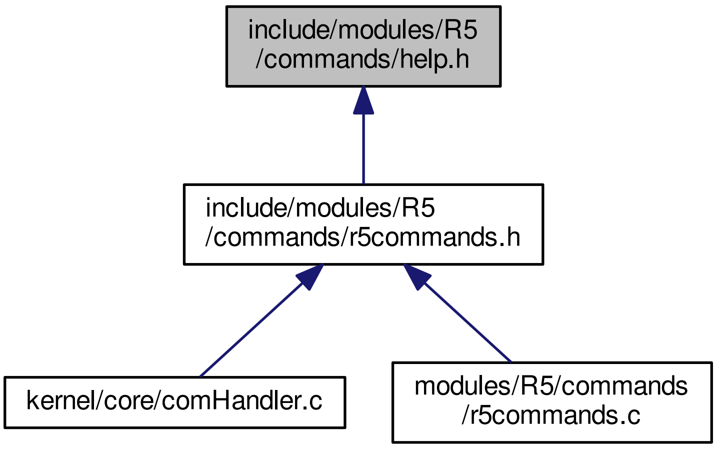
4.7.1.2 #define HELP_R3_COMMAND_YIELD

Value:

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

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

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



Macros

- #define HELP_R5_COMMAND_SHOWMEMORY

4.8.1 Macro Definition Documentation

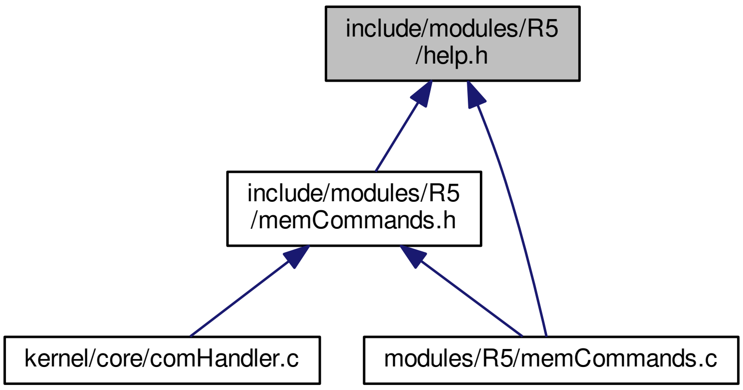
4.8.1.1 #define HELP_R5_COMMAND_SHOWMEMORY

Value:

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

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

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



Macros

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

4.9.1 Macro Definition Documentation

4.9.1.1 #define HELP_R5_COMMAND_ALLOC

Value:

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

4.9.1.2 #define HELP_R5_COMMAND_EMPTY

Value:

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

4.9.1.3 #define HELP_R5_COMMAND_FREE

Value:

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

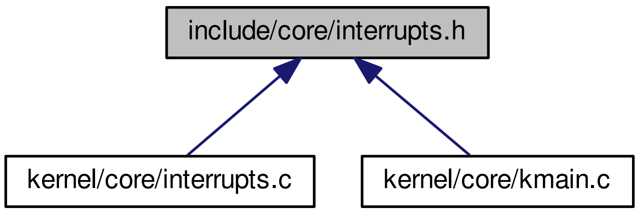
4.9.1.4 #define HELP_R5_COMMAND_HEAP

Value:

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

4.10 include/core/interrupts.h File Reference

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



Functions

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

4.10.1 Function Documentation

4.10.1.1 void init_irq (void)

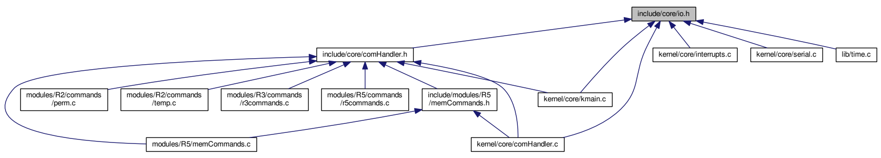
Installs the initial interrupt handlers for the first 32 irq lines. Most do a panic for now.

4.10.1.2 void init_pic (void)

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

4.11 include/core/io.h File Reference

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


include/core/io.h
include/core/comHandler.h
kernel/core/interrupts.c
kernel/core/serial.c
lib/time.c
modules/R2/commands /perm.c
modules/R2/commands /temp.c
modules/R3/commands /r3commands.c
modules/R5/commands /r5commands.c
include/modules/R5 /memCommands.h
kernel/core/kmain.c
modules/R5/memCommands.c
kernel/core/comHandler.c

Macros

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

4.11.1 Macro Definition Documentation

4.11.1.1 #define inb(*port*)

Value:

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

Reads a byte of data from a port.

Parameters

port	The port to read the data from

Returns

The byte from the port

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

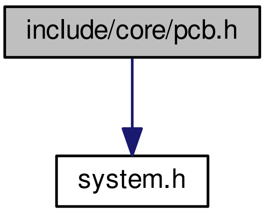
Writes a byte of data to a port.

Parameters

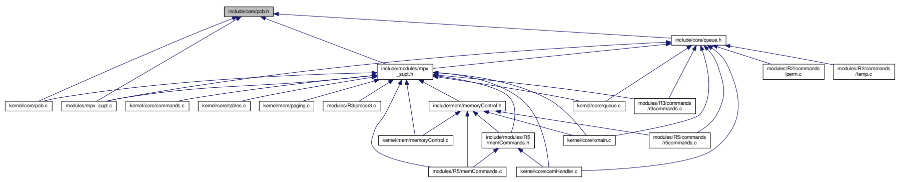
port	The port to write the data to
data	The byte to write

4.12 include/core/pcb.h File Reference

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



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



Data Structures

- struct pcb

Macros

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

Typedefs

- typedef struct pcb pcb

Functions

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

4.12.1 Macro Definition Documentation

4.12.1.1 #define APPLICATION 1

4.12.1.2 #define BLOCKED 0

4.12.1.3 #define READY 1

4.12.1.4 #define RUNNING 2

4.12.1.5 #define SYSTEM 0

4.12.2 Typedef Documentation

4.12.2.1 typedef struct pcb pcb

4.12.3 Function Documentation

4.12.3.1 pcb∗ allocatePCB ( )

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

Returns

PCB pointer or Null if error occurs

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

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

Returns

PCB pointer or Null if error occurs

4.12.3.2 int checkParamClass (int *processClass*)

Validates that the processClass is valid

Parameters

processClass	- int

Returns

integer 0 or 1 if valid

4.12.3.3 int checkParamName (const char ∗ *processName*)

Validates that the processName is valid

Parameters

processName	- const char ∗ processName

Returns

integer 0 or 1 if valid

4.12.3.4 int checkParamPriority (int *priority*)

Validates that the priority is valid

Parameters

priority	- int

Returns

integer 0 or 1 if valid

4.12.3.5 int freePCB (pcb ∗ *pcbPtr*)

Frees memory that is allocated for the pcb provided

Parameters

pcbPtr	pointer to pcb to be freed

Returns

integer code - 1 if successful, -1 otherwise

Frees memory that is allocated for the pcb provided

Parameters

pcbPtr	pointer to pcb to be freed

Returns

integer code - 1 if successful, 0 otherwise

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

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

Parameters

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

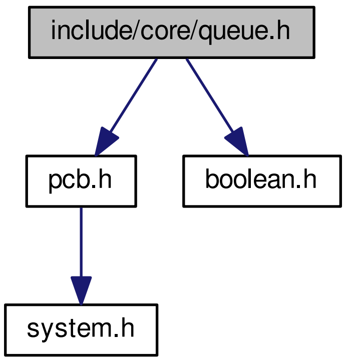
Returns

PCB pointer to new pcb or NULL if there were errors

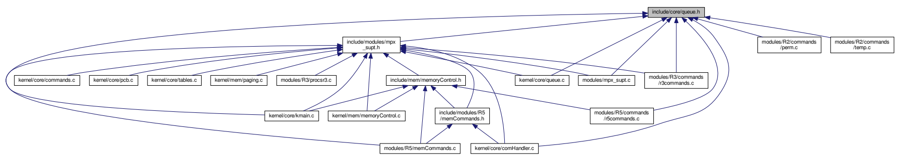
4.13 include/core/queue.h File Reference

```
#include "pcb.h"
#include "boolean.h"
```

Include dependency graph for queue.h:



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



Data Structures

- struct node

Typedefs

- typedef struct node node

Functions

- node ∗ getReadyQueue ()
- node ∗ getBlockedQueue ()
- node ∗ getSuspendedReadyQueue ()
- node ∗ getSuspendedBlockedQueue ()
- pcb ∗ popReady ()
- pcb ∗ popBlocked ()
- pcb ∗ popSuspendedReady ()
- pcb ∗ popSuspendedBlocked ()
- boolean insertPCB (pcb ∗p)
- boolean removePCB (pcb ∗p)
- pcb ∗ findPCB (const char ∗processName)

4.13.1 Typedef Documentation

4.13.1.1 typedef struct **node node**

The struct representing a node in a queue

4.13.2 Function Documentation

4.13.2.1 **pcb**∗ findPCB (const char ∗ *processName*)

Finds the PCB with the given process name.

Parameters

processName	The name of the process to search for

Returns

A pointer to the PCB, or null if not found

4.13.2.2 node∗ getBlockedQueue ( )

Gets the head node of the blocked queue.

Returns

The head node of the blocked queue

4.13.2.3 node∗ getReadyQueue ( )

Gets the head node of the ready queue.

Returns

The head node of the ready queue

4.13.2.4 node∗ getSuspendedBlockedQueue ( )

Gets the head node of the suspended-blocked queue.

Returns

The head node of the suspended-blocked queue

4.13.2.5 node∗ getSuspendedReadyQueue ( )

Gets the head node of the suspended-ready queue.

Returns

The head node of the suspended-ready queue

4.13.2.6 boolean insertPCB (pcb ∗ *p*)

Inserts the PCB into the appropriate queue.

Parameters

p	The PCB to insert.

Returns

true if the PCB was inserted, false otherwise

4.13.2.7 pcb∗ popBlocked ( )

Pops the next node off of the blocked queue.

Returns

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

4.13.2.8 pcb∗ popReady ( )

Pops the next node off of the ready queue.

Returns

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

4.13.2.9 pcb∗ popSuspendedBlocked ( )

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

Returns

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

4.13.2.10 pcb∗ popSuspendedReady ( )

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

Returns

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

4.13.2.11 boolean removePCB (pcb ∗ *p*)

Removes the given PCB from it's queue.

Parameters

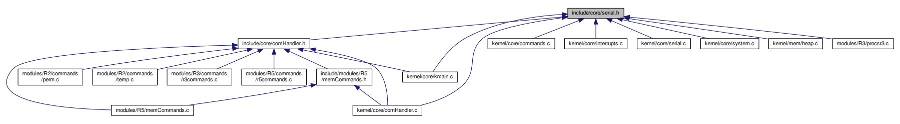
p	The PCB to remove

Returns

true if the PCB was removed, false otherwise

4.14 include/core/serial.h File Reference

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



Macros

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

Functions

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

4.14.1 Macro Definition Documentation

4.14.1.1 #define COM1 0x3f8

4.14.1.2 #define COM2 0x2f8

4.14.1.3 #define COM3 0x3e8

4.14.1.4 #define COM4 0x2e8

4.14.2 Function Documentation

4.14.2.1 int init_serial (int *device*)

Initializes devices for user interaction, logging, ...

Parameters

device	The device to initialize

Returns

The error code

4.14.2.2 int serial_print (const char * *msg*)

Writes a message to the active serial output device.

Parameters

msg	The message to write

Returns

The error code

4.14.2.3 int serial_println (const char * *msg*)

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

Parameters

msg	The message to write

Returns

The error code

4.14.2.4 int set_serial_in (int *device*)

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

Parameters

device	The divce to set as input

Returns

The error code

4.14.2.5 int set_serial_out (int *device*)

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

Parameters

device	The device to set as output

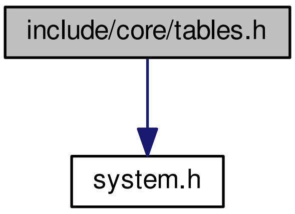
Returns

The error code

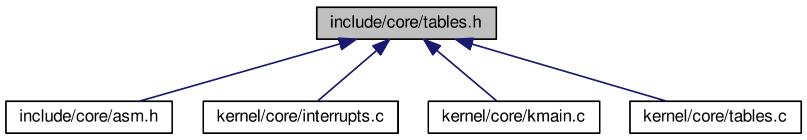
4.15 include/core/tables.h File Reference

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

Include dependency graph for tables.h:



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



Data Structures

- struct idt_entry_struct
- struct idt_struct
- struct gdt_descriptor_struct
- struct gdt_entry_struct

Functions

- struct idt_entry_struct __attribute__ ((packed)) idt_entry
- void idt_set_gate (u8int idx, u32int base, u16int sel, u8int flags)
- void gdt_init_entry (int idx, u32int base, u32int limit, u8int access, u8int flags)
- void init_idt ()
- void init_gdt ()

Variables

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

4.15.1 Function Documentation

4.15.1.1 struct **idt_entry_struct** __attribute__ ((packed))

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

Installs a new table entry into the global descriptor table.

Parameters

idx	
base	
limit	
access	
flags	

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

Installs a new gate entry into the IDT.

Parameters

idx	
base	
sel	
flags	

4.15.1.4 void init_gdt ( )

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

4.15.1.5 void init_idt ( )

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

4.15.2 Variable Documentation

4.15.2.1 u8int access

4.15.2.2 u32int base

4.15.2.3 u8int base_high

4.15.2.4 u16int base_low

4.15.2.5 u8int base_mid

4.15.2.6 u8int flags

4.15.2.7 u16int limit

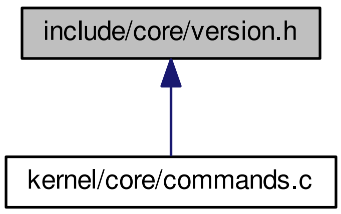
4.15.2.8 u16int limit_low

4.15.2.9 u16int sselect

4.15.2.10 u8int zero

4.16 include/core/version.h File Reference

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



Macros

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

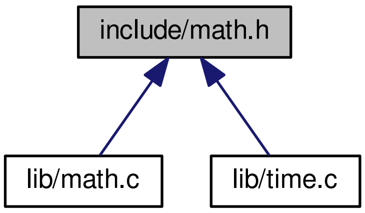
4.16.1 Macro Definition Documentation

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

The current OS version.

4.17 include/math.h File Reference

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



Functions

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

4.17.1 Function Documentation

4.17.1.1 unsigned char bcdToDec (unsigned char *bcd*)

Converts a BCD encoded byte to a decimal encoded byte

Parameters

bcd	The value to convert

Returns

The decimal value

4.17.1.2 unsigned char decToBcd (unsigned char *dec*)

Converts a decimal encoded byte to a BCD encoded byte

Parameters

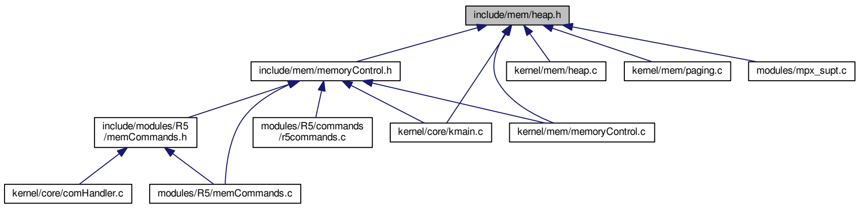
dec	The value to convert

Returns

The BCD value

4.18 include/mem/heap.h File Reference

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



Data Structures

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

Macros

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

Functions

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

Variables

- typedef __attribute__

4.18.1 Macro Definition Documentation

4.18.1.1 #define KHEAP_BASE 0xD000000

4.18.1.2 #define KHEAP_MIN 0x10000

4.18.1.3 #define KHEAP_SIZE 0x1000000

4.18.1.4 #define TABLE_SIZE 0x1000

Kernel heap.

4.18.2 Function Documentation

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

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

Parameters

size	The amount of memory to allocate
align	The page alignment
phys_addr	The physical address

Returns

The memory address

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

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

Parameters

size	The amount of memory to allocate
hp	The heap to allocate on
align	The page alignment

Returns

The memory address

4.18.2.3 void init_kheap ( )

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

4.18.2.4 u32int kfree ( )

Free kernel memory.

Returns

4.18.2.5 u32int kmalloc (u32int *size*)

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

Parameters

size	The amount of memory to allocate

Returns

The memory address

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

Create a new heap.

Parameters

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

Returns

The address of the heap

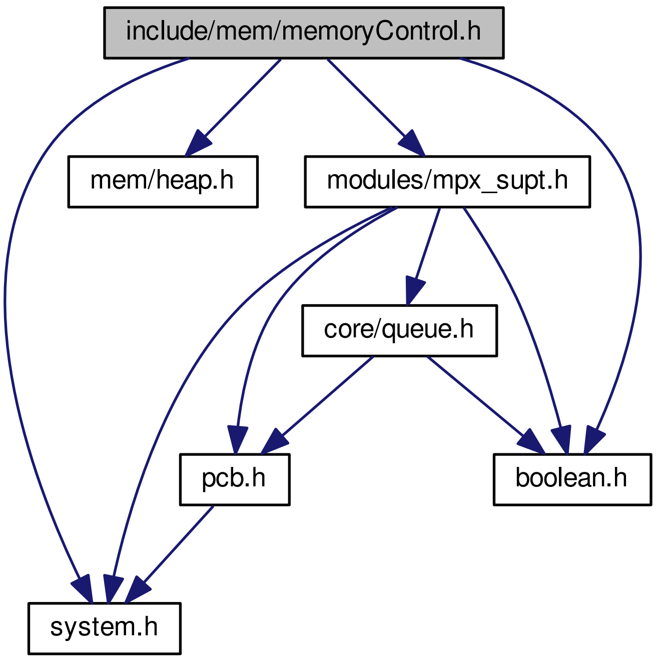
4.18.3 Variable Documentation

4.18.3.1 struct gdt_entry_struct __attribute__

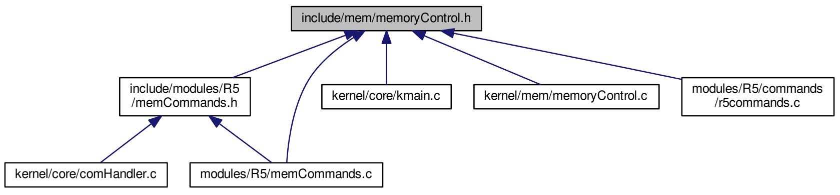
4.19 include/mem/memoryControl.h File Reference

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

Include dependency graph for memoryControl.h:



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



include/mem/memoryControl.h
include/modules/R5 /memCommands.h
kernel/core/kmain.c
kernel/mem/memoryControl.c
modules/R5/commands /r5commands.c
kernel/core/comHandler.c
modules/R5/memCommands.c

Data Structures

- struct cmcb
- struct lmcb

Macros

- #define FREE 0
- #define ALLOCATED 1

Typedefs

- typedef struct cmcb cmcb
- typedef struct lmcb lmcb

Functions

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

4.19.1 Macro Definition Documentation

4.19.1.1 #define ALLOCATED 1

4.19.1.2 #define FREE 0

4.19.2 Typedef Documentation

4.19.2.1 typedef struct cmcb cmcb

4.19.2.2 typedef struct lmcb lmcb

4.19.3 Function Documentation

4.19.3.1 void∗ allocateMemory (int *size*)

Allocates a memory block if enough memory is availabel

Parameters

size	- size of memory to allocate in bytes

Returns

pointer to the me

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

Deallocates the block of memory at the mempointer

Parameters

memPointer	- pointer to the mem block

Returns

boolean - tells whether successfull dealloc

Deallocates the block of memory at the mempointer

Parameters

memPointer	- pointer to the mem block

Returns

boolean - boolean telling whether succesful dealloc

4.19.3.3 **cmcb**∗ getAllocatedHead ( )

Returns the head to the allocated list

Returns

cmcb ∗ to the allocated list head

4.19.3.4 **cmcb**∗ getFreeHead ( )

Returns the head of the free list

Returns

cmcb ∗ to the free list head

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

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

Parameters

size	- size of heap in bytes

Returns

boolean - boolean denoting if heap was initialized

4.19.3.6 boolean isEmpty ( )

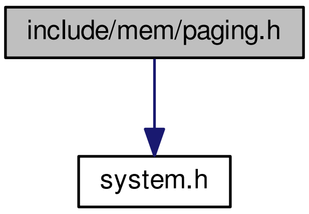
Returns a boolean telling if all the memory is empty

Returns

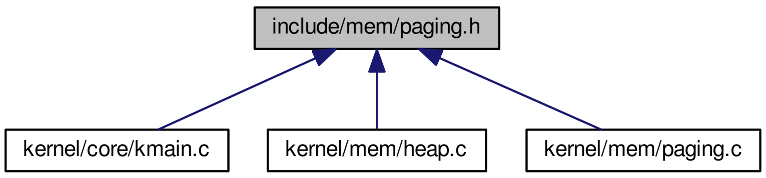
boolean

4.20 include/mem/paging.h File Reference

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



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



Data Structures

- struct page_entry
- struct page_table
- struct page_dir

Macros

- #define PAGE_SIZE 0x1000

Functions

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

4.20.1 Macro Definition Documentation

4.20.1.1 #define PAGE_SIZE 0x1000

4.20.2 Function Documentation

4.20.2.1 void clear_bit (u32int *addr*)

Marks a page frame bit as free (0).

Parameters

addr	The address of the frame

4.20.2.2 u32int first_free ( )

Finds the first free page frame.

Returns

The first free page frame

4.20.2.3 u32int get_bit (u32int *addr*)

Checks if page frame is in use.

Parameters

addr	The address of the frame

Returns

True if it is in use

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

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

Parameters

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

Returns

A pointer to the page

4.20.2.5 void init_paging ( )

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

4.20.2.6 void load_page_dir (page_dir ∗ *new_dir*)

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

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

Parameters

new_page_dir	The page directory to set as the current

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

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

Parameters

new_page_dir	The page directory to set as the current

4.20.2.7 void new_frame (page_entry * *page*)

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

Parameters

page	The page to create the frame in

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

Parameters

page	The page to create the frame in

4.20.2.8 void set_bit (u32int *addr*)

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

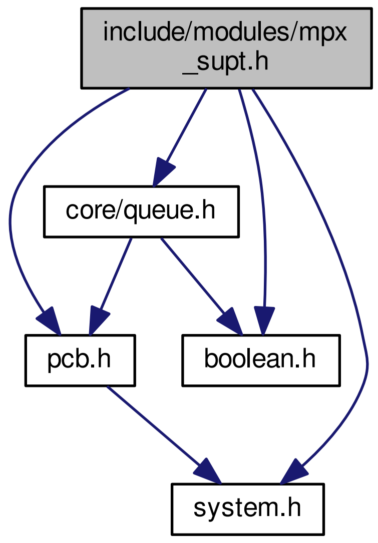
Parameters

addr	The address of the frame

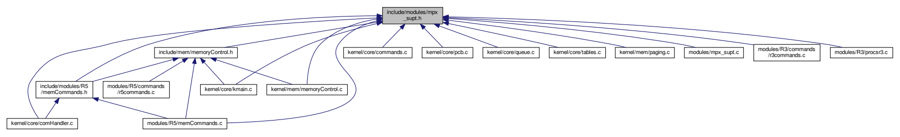
4.21 include/modules/mpx_supt.h File Reference

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

Include dependency graph for mpx_supt.h:



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



Data Structures

- struct param
- struct context

Macros

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

Typedefs

- typedef struct context context

Functions

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

4.21.1 Macro Definition Documentation

4.21.1.1 #define EXIT 0

4.21.1.2 #define IDLE 1

4.21.1.3 #define MODULE_R1 0

4.21.1.4 #define MODULE_R2 1

4.21.1.5 #define MODULE_R3 2

4.21.1.6 #define MODULE_R4 4

4.21.1.7 #define MODULE_R5 8

4.21.1.8 #define READ 2

4.21.1.9 #define WRITE 3

4.21.2 Typedef Documentation

4.21.2.1 typedef struct context context

4.21.3 Function Documentation

4.21.3.1 const char∗ getCOPName ( )

Gets the name of the COP

Returns

const char pointer name

4.21.3.2 void idle ( )

The idle process

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

Set a region of memory

Parameters

s	Destination
c	Byte to write
n	Count

Returns

s

4.21.3.4 void mpx_init (int *cur_mod*)

Initialize MPX support software

Parameters

cur_mod	(symbolic constants MODULE_R1, MODULE_R2, etc)

4.21.3.5 void∗ sys_alloc_mem (u32int *size*)

Allocates a block of memory (similar to malloc)

Parameters

size	Number of bytes to allocate

Returns

The allocated memory

4.21.3.6 u32int∗ sys_call (context ∗ *registers*)

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

Parameters

registers	- copy of register values

Returns

u32int position of stackTop

4.21.3.7 int sys_free_mem (void ∗ *ptr*)

Frees memory

Parameters

ptr	Pointer to the block of memory to free

Returns

4.21.3.8 int sys_req (int *op_code*)

Generates interrupt 60H

Parameters

op_code	(IDLE)

Returns

0

4.21.3.9 void sys_set_free (boolean(func)(void ∗))

Sets the memory free function for sys_free_mem

Parameters

func	Function pointer to the memory free-er

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

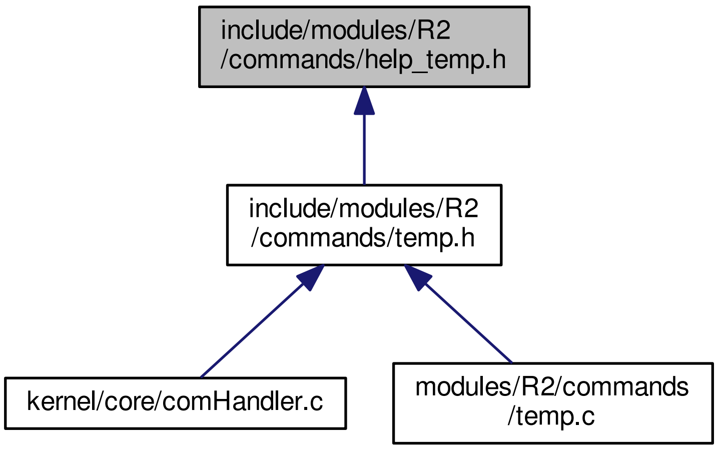
Sets the memory allocation function for sys_alloc_mem

Parameters

func	Function pointer to the memory allocator

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

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



Macros

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

4.22.1 Macro Definition Documentation

4.22.1.1 #define HELP_R2_COMMAND_BPCB

Value:

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

4.22.1.2 #define HELP_R2_COMMAND_CPCB

Value:

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

4.22.1.3 #define HELP_R2_COMMAND_DPCB

Value:

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

4.22.1.4 #define HELP_R2_COMMAND_UPCB

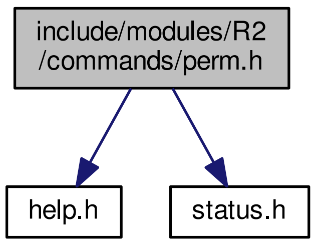
Value:

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

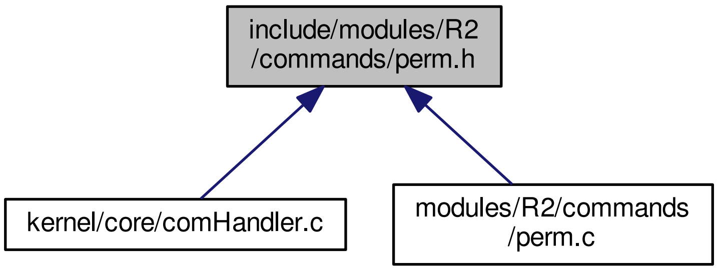
4.23 include/modules/R2/commands/perm.h File Reference

```
#include "help.h"
#include "status.h"
```

Include dependency graph for perm.h:



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



Functions

- void registerR2PermCommands ()
- const char ∗ suspendPcb (char ∗∗args, int numArgs)
- const char ∗ resumePcb (char ∗∗args, int numArgs)
- const char ∗ setPriorityPcb (char ∗∗args, int numArgs)
- const char ∗ showPcbInfo (char ∗∗args, int numArgs)

4.23.1 Function Documentation

4.23.1.1 void registerR2PermCommands ( )

Registers the permanent commands in the command handler

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

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

Usage: rpcb name

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

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

Usage: rpcb name

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

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

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

Usage: ppcb name priority

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

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

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

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

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

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

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

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

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

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

Usage: spcb name

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

Parameters

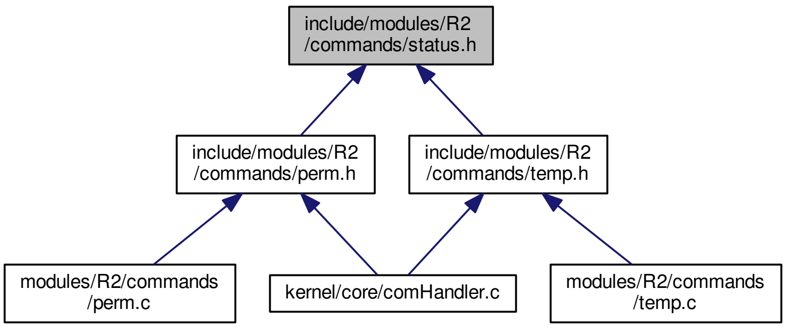
args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

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

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


include/modules/R2 /commands/status.h
include/modules/R2 /commands/perm.h
include/modules/R2 /commands/temp.h
modules/R2/commands /perm.c
kernel/core/comHandler.c
modules/R2/commands /temp.c

Macros

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

4.24.1 Macro Definition Documentation

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

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

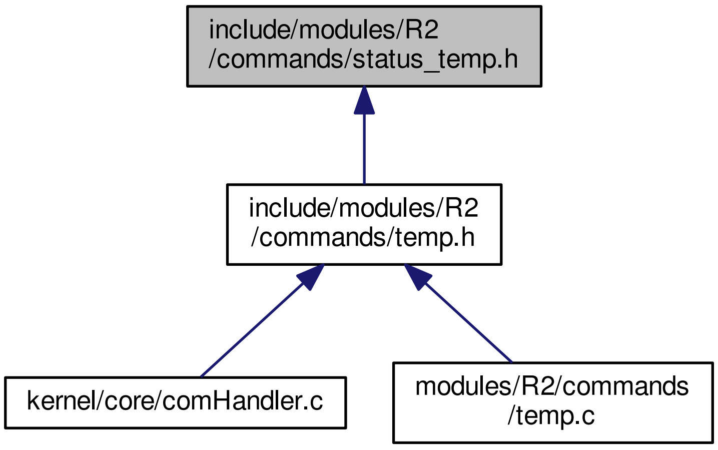
4.24.1.3 #define SUSPEND_PCB_SUCCESS ((const char∗) "Process suspended.")

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

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

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

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



Macros

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

4.25.1 Macro Definition Documentation

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

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

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

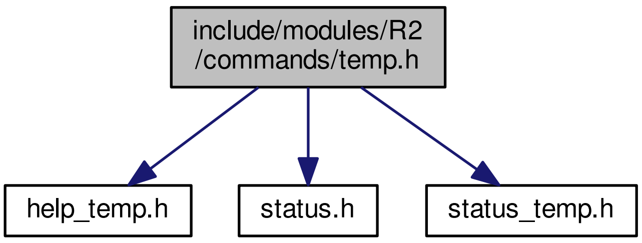
4.25.1.4 #define PROCESS_NAME_ALREADY_EXISTS ((const char∗) "This process name already exists")

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

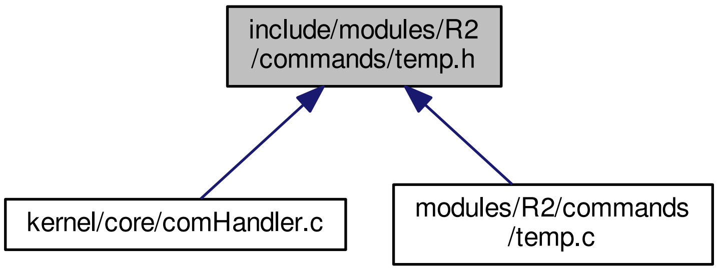
4.26 include/modules/R2/commands/temp.h File Reference

```
#include "help_temp.h"
#include "status.h"
#include "status_temp.h"
```

Include dependency graph for temp.h:



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



Functions

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

4.26.1 Function Documentation

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

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

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

Usage: bpcb name

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

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

Creates a PCB and inserts it into the appropriate queue.

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

Usage: cpcb name class priority

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

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

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

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

Usage: dpcb name

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

4.26.1.4 void registerR2TempCommands ( )

Registers the temporary commands in the command handler

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

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

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

Usage: upcb name

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

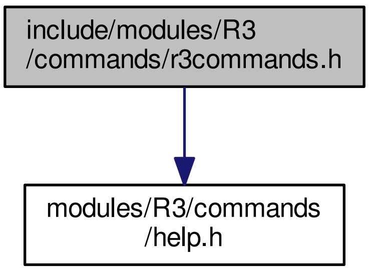
Returns

A status message indicating success/failure

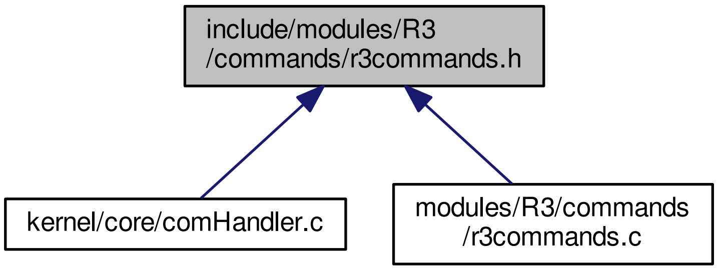
4.27 include/modules/R3/commands/r3commands.h File Reference

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

Include dependency graph for r3commands.h:



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



Functions

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

4.27.1 Function Documentation

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

Loads the r3 processes to the queue.
")

Usage: loadr3

Args: [no args] - loads processes

Parameters

args	The arguments to pass to the function

Returns

""

4.27.1.2 void registerR3Commands ( )

Registers commands in command handler

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

Yields command handler to allow other processes to run.

Usage: yield

Args: [no args] - yields command handler

Parameters

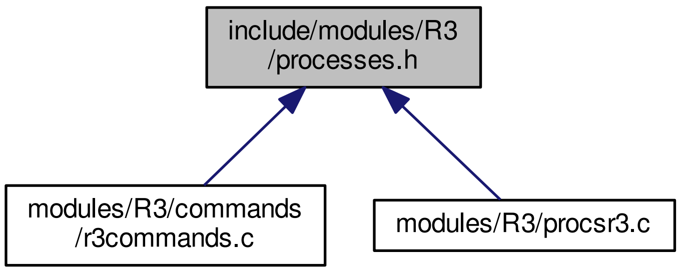
args	The arguments to pass to the function

Returns

""

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

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


include/modules/R3 /processes.h
modules/R3/commands /r3commands.c
modules/R3/procsr3.c

Functions

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

4.28.1 Function Documentation

4.28.1.1 void proc1 ( )

4.28.1.2 void proc2 ( )

4.28.1.3 void proc3 ( )

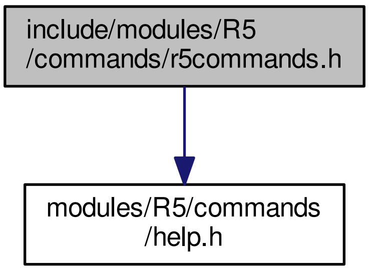
4.28.1.4 void proc4 ( )

4.28.1.5 void proc5 ( )

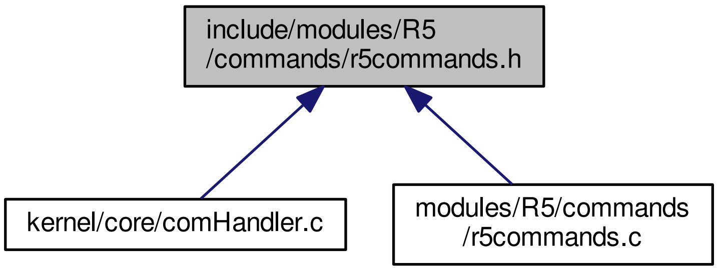
4.29 include/modules/R5/commands/r5commands.h File Reference

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

Include dependency graph for r5commands.h:



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



Functions

- void registerR5PermCommands ()
- const char ∗ showMemory (char ∗∗args, int numArgs)

4.29.1 Function Documentation

4.29.1.1 void registerR5PermCommands ( )

Registers commands in command handler

Registers the permanent commands in the command handler

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

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

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

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

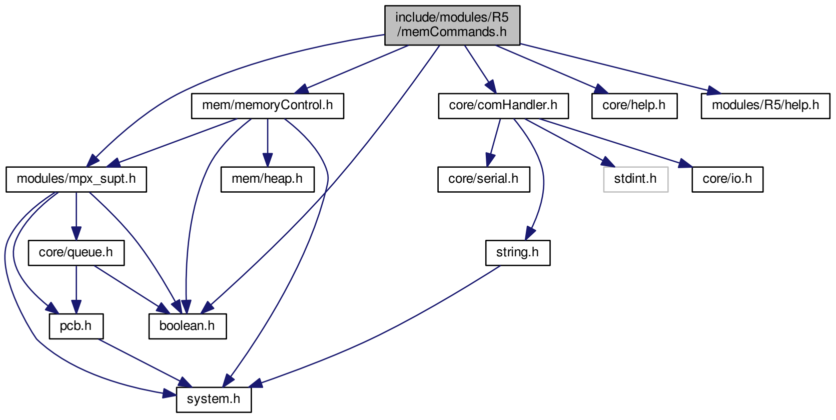
Returns

A status message indicating success/failure

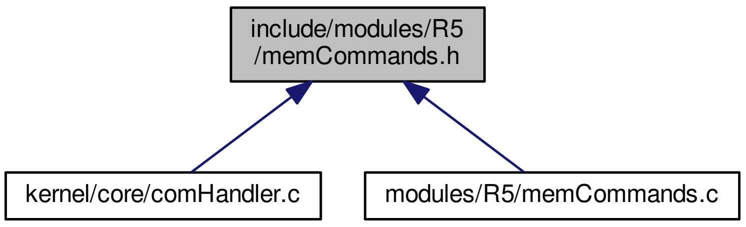
4.30 include/modules/R5/memCommands.h File Reference

```
#include <mem/memoryControl.h>
#include <boolean.h>
#include <core/comHandler.h>
#include <core/help.h>
#include <modules/R5/help.h>
#include <modules/mpx_supt.h>
```

Include dependency graph for memCommands.h:



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



Functions

- void registerR5TempCommands ()
- const char ∗ initHeap (char ∗∗args, int numArgs)
- const char ∗ allocateMem (char ∗∗args, int numArgs)
- const char ∗ freeMemory (char ∗∗args, int numArgs)
- const char ∗ isEmptyCom (char ∗∗args, int numArgs)

4.30.1 Function Documentation

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

Allocates a memory block if enough memory is availabel

Parameters

size	- size of memory to allocate in bytes

Returns

pointer to the me

Allocates a memory block if enough memory is available

Parameters

size	- size of memory to allocate in bytes

Returns

pointer to the me

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

Deallocates the block of memory at the mempointer

Parameters

memPointer	- pointer to the mem block

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

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

Returns

True or false

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

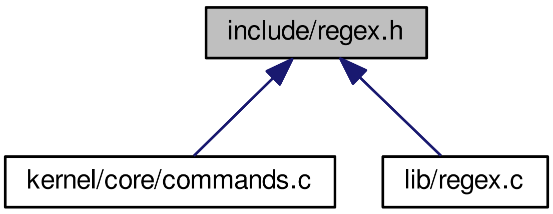
Check if memory is empty

4.30.1.5 void registerR5TempCommands ( )

Registers the permanent commands in the command handler

4.31 include/regex.h File Reference

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



Functions

- int testRegex (const char ∗regex, const char ∗stringToCheck)

4.31.1 Function Documentation

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

Tests if the stringToCheck adheres to the given regex string

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

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

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

Parameters

regex	
stringToCheck	

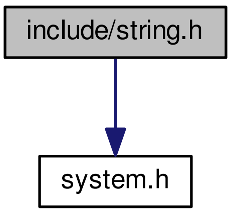
Returns

1 if adheres, 0 otherwise

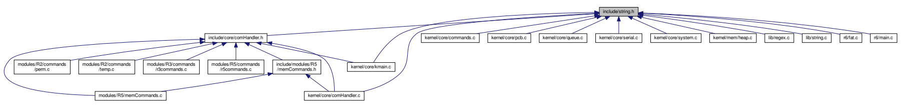
4.32 include/string.h File Reference

`#include <system.h>`

Include dependency graph for string.h:



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



Functions

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

4.32.1 Function Documentation

4.32.1.1 int atoi (const char ∗ *s*)

Convert an ASCII string to an integer

Parameters

s	The string to convert

Returns

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

4.32.1.2 int isChar (const char *c*)

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

Parameters

const	char c

Returns

1 if c is a char, 0 otherwies

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

Parameters

const	char c

Returns

1 if c is a char, 0 otherwise

4.32.1.3 int isdigit (const char *c*)

Determine if a character is a digit.

Parameters

c	The character to check

Returns

True if the character is a digit

4.32.1.4 int isLowerChar (const char *c*)

Checks if the given char is a-z

Parameters

const	char c

Returns

1 if c is a lower char, 0 otherwise

4.32.1.5 int isspace (const char ∗ *c*)

Determine if a character is whitespace.

Parameters

c	The character to check

Returns

True if the character is a whitespace character

4.32.1.6 int isUpperChar (const char *c*)

Checks if the given char is A-Z

Parameters

const	char c

Returns

1 if c is a upper char, 0 otherwise

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

Converts an integer to an ASCII string.

Parameters

num	The number to convert
str	The destination string
base	The radix

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

Reverses a string.

Parameters

str	The string to reverse
len	The length of the string

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

Concatenate the contents of one string onto another.

Parameters

s1	The destination string
s2	The source string

Returns

A pointer to the destination string

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

Compares two strings to each other

Parameters

s1	The first string
s2	The second string

Returns

The difference between the characters at the first index of indifference

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

Copy on string to another.

Parameters

cpy	The destination string
ori	The source string

Returns

A pointer to the destination string

4.32.1.12 int strlen (const char ∗ *s*)

Returns the length of a string.

Parameters

s	The input string

Returns

The length of the string

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

Split string into tokens

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

Parameters

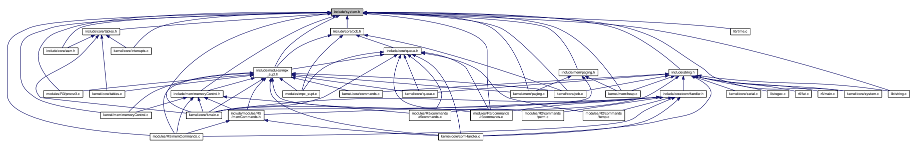
s1	The string to split
s2	The delimiter

Returns

A single token

4.33 include/system.h File Reference

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



Macros

- #define NULL 0
- #define no_warn(p) if (p) while (1) break

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

Typedefs

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

Functions

- void klogv (const char ∗msg)
- void kpanic (const char ∗msg)

4.33.1 Macro Definition Documentation

4.33.1.1 #define asm __asm__

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

Turn IRQs off.

4.33.1.3 #define GDT_CS_ID 0x01

Kernel code segment ID.

4.33.1.4 #define GDT_DS_ID 0x02

Kernel data segment ID.

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

Halt.

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

Interrupt return.

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

Suppress 'unused parameter' warnings/errors

Parameters

p	The parameter

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

Skip cycle.

4.33.1.9 #define NULL 0

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

Turn IRQs on.

4.33.1.11 #define volatile __volatile__

4.33.2 Typedef Documentation

4.33.2.1 typedef unsigned int size_t

4.33.2.2 typedef unsigned short u16int

4.33.2.3 typedef unsigned long u32int

4.33.2.4 typedef unsigned char u8int

4.33.3 Function Documentation

4.33.3.1 void klogv (const char ∗ *msg*)

Kernel log message. Sent to active serial device.

Parameters

msg	The message to log

4.33.3.2 void kpanic (const char ∗ *msg*)

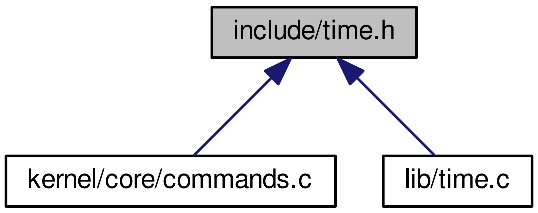
Kernel panic. Prints an error message and halts.

Parameters

msg	The error mesage to print

4.34 include/time.h File Reference

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



Data Structures

- struct date_time

Macros

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

Functions

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

Variables

- const int DAYS_IN_MONTH [13]

4.34.1 Macro Definition Documentation

4.34.1.1 #define APR ((const char∗) "April")

4.34.1.2 #define AUG ((const char∗) "August")

4.34.1.3 #define CONTROL_PORT 0x70

Registers for reading/writing data

4.34.1.4 #define DATA_PORT 0x71

4.34.1.5 #define DAY_MONTH 0x07

4.34.1.6 #define DAY_WEEK 0x06

4.34.1.7 #define DEC ((const char∗) "December")

4.34.1.8 #define FEB ((const char∗) "February")

4.34.1.9 #define FRI ((const char∗) "Friday")

4.34.1.10 #define HOURS 0x04

4.34.1.11 #define JAN ((const char∗) "January")

Month Names

4.34.1.12 #define JUL ((const char∗) "July")

4.34.1.13 #define JUN ((const char∗) "June")

4.34.1.14 #define MAR ((const char∗) "March")

4.34.1.15 #define MAY ((const char∗) "May")

4.34.1.16 #define MINUTES 0x02

4.34.1.17 #define MON ((const char∗) "Monday")

Day Names

4.34.1.18 #define MONTH 0x08

4.34.1.19 #define NMI_DISABLE 0x80

NMI Flags Disable - OR, Enable - AND

4.34.1.20 #define NMI_ENABLE 0x7F

4.34.1.21 #define NOV ((const char∗) "November")

4.34.1.22 #define OCT ((const char∗) "October")

4.34.1.23 #define SAT ((const char∗) "Saturday")

4.34.1.24 #define SECONDS 0x00

Aliases for accessing time/date

4.34.1.25 #define SEP ((const char∗) "September")

4.34.1.26 #define SUN ((const char∗) "Sunday")

4.34.1.27 #define THU ((const char∗) "Thursday")

4.34.1.28 #define TIME_DELIM ':'

The delimiter for the time

4.34.1.29 #define TUE ((const char∗) "Tuesday")

4.34.1.30 #define WED ((const char∗) "Wednesday")

4.34.1.31 #define YEAR 0x09

4.34.2 Function Documentation

4.34.2.1 date_time getDateTime ( )

Gets the date and time from the RTC registers.

Returns

The date and time stored in the RTC.

4.34.2.2 unsigned char getDayOfMonth ( )

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

Returns

The decimal-encoded day of the month.

4.34.2.3 unsigned char getDayOfWeek ( )

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

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

Returns

The decimal-encoded day of the week.

4.34.2.4 unsigned char getHours ( )

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

Returns

The decimal-encoded number of hours.

4.34.2.5 unsigned char getMinutes ( )

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

Returns

The decimal-encoded number of minutes.

4.34.2.6 unsigned char getMonth ( )

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

Returns

The decimal-encoded month.

4.34.2.7 unsigned char getSeconds ( )

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

Returns

The decimal-encoded number of seconds.

4.34.2.8 unsigned char getYear ( )

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

Returns

The decimal-encoded year.

4.34.2.9 int isLeapYear (int *year*)

Determines if the given year is a leap year.

Parameters

year	The year to check

Returns

True if the year is a leap year.

4.34.2.10 void setDateTime (date_time *dateTime*)

Sets the date and time to the specified values.

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

Parameters

dateTime	The values to set.

4.34.2.11 void setDayOfMonth (unsigned char *day*)

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

Parameters

dayOfMonth	The day of the month value to set

4.34.2.12 void setDayOfWeek (unsigned char *day*)

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

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

Parameters

dayOfWeek	The day of the week value to set

4.34.2.13 void setHours (unsigned char *hour*)

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

Parameters

hours	The hours value to set

4.34.2.14 void setMinutes (unsigned char *min*)

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

Parameters

minutes	The minutes value to set

4.34.2.15 void setMonth (unsigned char *mon*)

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

Parameters

month	The month value to set

4.34.2.16 void setSeconds (unsigned char *sec*)

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

Parameters

seconds	The seconds value to set

4.34.2.17 void setYear (unsigned char *year*)

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

Parameters

year	The year value to set

4.34.2.18 void updateDayOfWeek (date_time ∗ *dateTime*)

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

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

Parameters

dateTime	The date_time to update

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

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

Parameters

dateTime	The date_time to update.

4.34.2.19 void updateDayOfYear (date_time ∗ *dateTime*)

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

Parameters

dateTime	The date_time to update

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

Parameters

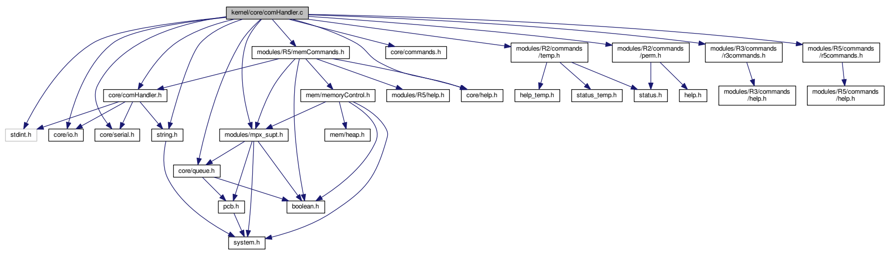
dateTime	The date_time to update.

4.34.3 Variable Documentation

4.34.3.1 const int DAYS_IN_MONTH[13]

4.35 kernel/core/comHandler.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <core/comHandler.h>
#include <core/io.h>
#include <core/serial.h>
#include <core/help.h>
#include <core/commands.h>
#include <core/queue.h>
#include <modules/R2/commands/temp.h>
#include <modules/R2/commands/perm.h>
#include <modules/R3/commands/r3commands.h>
#include <modules/R5/commands/r5commands.h>
#include <modules/R5/memCommands.h>
#include <modules/mpx_supt.h>
```
Include dependency graph for comHandler.c:



Functions

- void addFunctionDef (char ∗name, const char ∗helpString, const char ∗(funcPointer)(char ∗∗args, int num↩ Args))
- functionDef getFunctionDef (char ∗name)
- const char ∗ getHelpString (char ∗name)
- char ∗ getComHistory (int isPrev)

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

Variables

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

4.35.1 Function Documentation

4.35.1.1 void addComHistory (char ∗ *newCom*)

Helper function to add a command to the command history array

Parameters

newCom	string to add to the command history

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

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

Parameters

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

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

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

Parameters

endIndex	- index of last char printed
insertionIndex	- index of where insertion point should be

4.35.1.4 void executeCommand (char ∗ *commandString*)

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

Parameters

commandString	string contianing the command name and any args

4.35.1.5 char∗ getComHistory (int *isPrev*)

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

Parameters

isPrev	integer denoting if to get the previous command

Returns

string of the command

4.35.1.6 functionDef getFunctionDef (char ∗ *name*)

Gets the functionDef struct corresponding to the name provided, returns a functionDef with null funcPointer if none are found

Parameters

name	- name of the functionDef

Returns

functionDef

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

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

Parameters

name	- name associated witht he struct from which to get the help string

Returns

const char∗ help string

4.35.1.8 char∗ getInput ( )

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

Returns

string that was input

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

Returns help for the specified commands.

Usage: help commandName

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

The help string

4.35.1.10 void initCommandHandler ( )

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

4.35.1.11 void printStart ( )

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

4.35.1.12 void returnToInsertionPoint (int *endIndex,* int *insertionIndex*)

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

Parameters

endIndex	- index of last char printed
insertionIndex	- index of where insertion point should be

4.35.1.13 void setupCommands ( )

Initialization function to add commands ot the functionDefs array

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

Shuts down the OS after asking for confirmation.

Usage: shutdown [–confirm]

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

True if shutdown was confirmed

4.35.2 Variable Documentation

4.35.2.1 char buffer[256]

4.35.2.2 char comHistory[10][256]

4.35.2.3 int comHistoryPos = 0

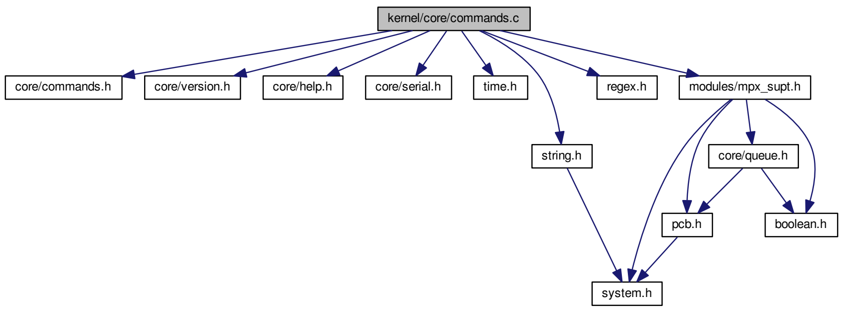
4.35.2.4 int continueHandle = 1

4.35.2.5 functionDef functionDefs[256]

4.35.2.6 int functionInsertPoint = 0

4.36 kernel/core/commands.c File Reference

```
#include <core/commands.h>
#include <core/version.h>
#include <core/help.h>
#include <core/serial.h>
#include <time.h>
#include <string.h>
#include <regex.h>
#include <modules/mpx_supt.h>
```
Include dependency graph for commands.c:



Functions

- const char ∗ version (char ∗∗args, int numArgs)
- const char ∗ date (char ∗∗args, int numArgs)

4.36.1 Function Documentation

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

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

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

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

Parameters

args	The arguments to pass to the function

Returns

The ISO-8601 formatted date

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

Returns the current version of the OS.

Usage: version

Args: [no args] - Returns the version

Parameters

args	The arguments to pass to the function

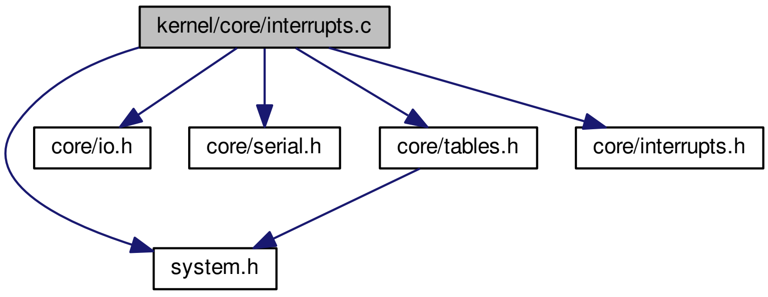
Returns

The version of the OS.

4.37 kernel/core/interrupts.c File Reference

```
#include <system.h>
#include <core/io.h>
#include <core/serial.h>
#include <core/tables.h>
#include <core/interrupts.h>
```

Include dependency graph for interrupts.c:



Macros

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

Functions

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

Variables

- idt_entry idt_entries [256]

4.37.1 Macro Definition Documentation

4.37.1.1 #define ICW1 0x11

4.37.1.2 #define ICW4 0x01

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

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

4.37.1.4 #define PIC1 0x20

4.37.1.5 #define PIC2 0xA0

4.37.2 Function Documentation

4.37.2.1 void bounds ( )

4.37.2.2 void breakpoint ( )

4.37.2.3 void coprocessor ( )

4.37.2.4 void coprocessor_segment ( )

4.37.2.5 void debug ( )

4.37.2.6 void device_not_available ( )

4.37.2.7 void divide_error ( )

4.37.2.8 void do_bounds ( )

4.37.2.9 void do_breakpoint ( )

4.37.2.10 void do_coprocessor ( )

4.37.2.11 void do_coprocessor_segment ( )

4.37.2.12 void do_debug ( )

4.37.2.13 void do_device_not_available ( )

4.37.2.14 void do_divide_error ( )

4.37.2.15 void do_double_fault ( )

4.37.2.16 void do_general_protection ( )

4.37.2.17 void do_invalid_op ( )

4.37.2.18 void do_invalid_tss ( )

4.37.2.19 void do_isr ( )

4.37.2.20 void do_nmi ( )

4.37.2.21 void do_overflow ( )

4.37.2.22 void do_page_fault ( )

4.37.2.23 void do_reserved ( )

4.37.2.24 void do_segment_not_present ( )

4.37.2.25 void do_stack_segment ( )

4.37.2.26 void double_fault ( )

4.37.2.27 void general_protection ( )

4.37.2.28 void init_irq (void)

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

4.37.2.29 void init_pic (void)

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

4.37.2.30 void invalid_op ( )

4.37.2.31 void invalid_tss ( )

4.37.2.32 void isr0 ( )

4.37.2.33 void nmi ( )

4.37.2.34 void overflow ( )

4.37.2.35 void page_fault ( )

4.37.2.36 void reserved ( )

4.37.2.37 void rtc_isr ( )

4.37.2.38 void segment_not_present ( )

4.37.2.39 void stack_segment ( )

4.37.2.40 void sys_call_isr ( )

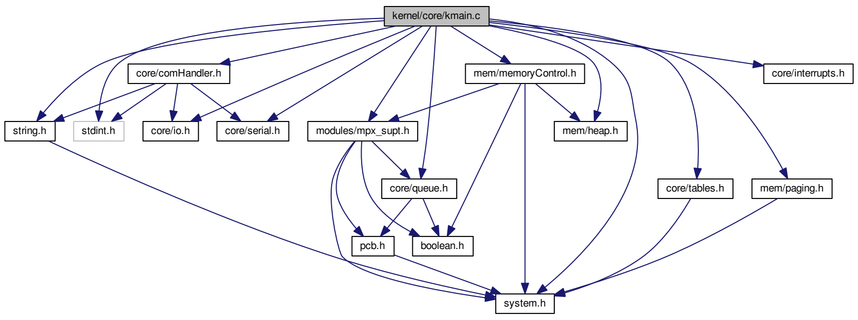
4.37.3 Variable Documentation

4.37.3.1 idt_entry idt_entries[256]

4.38 kernel/core/kmain.c File Reference

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

Include dependency graph for kmain.c:



Functions

- void kmain (void)

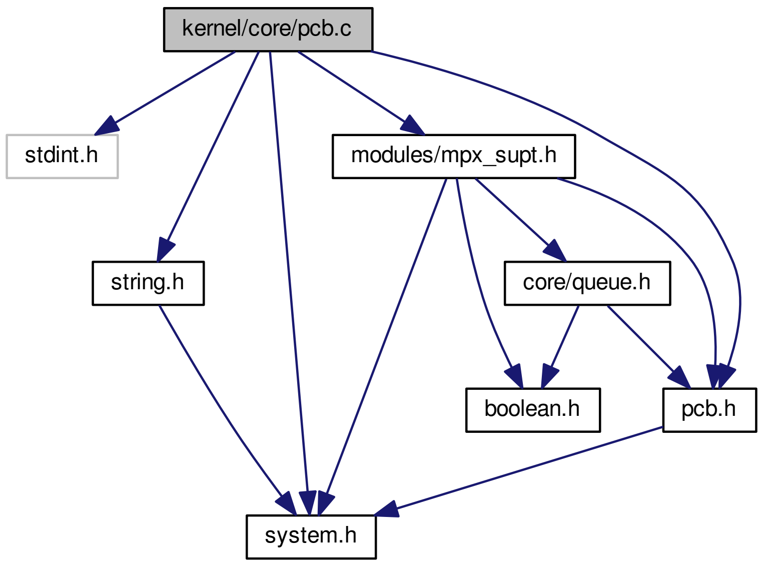
4.38.1 Function Documentation

4.38.1.1 void kmain (void)

4.39 kernel/core/pcb.c File Reference

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

Include dependency graph for pcb.c:



Functions

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

4.39.1 Function Documentation

4.39.1.1 pcb∗ allocatePCB ( )

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

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

Returns

PCB pointer or Null if error occurs

4.39.1.2 int checkParamClass (int *processClass*)

Validates that the processClass is valid

Parameters

processClass	- int

Returns

integer 0 or 1 if valid

4.39.1.3 int checkParamName (const char ∗ *processName*)

Validates that the processName is valid

Parameters

processName	- const char ∗ processName

Returns

integer 0 or 1 if valid

4.39.1.4 int checkParamPriority (int *priority*)

Validates that the priority is valid

Parameters

priority	- int

Returns

integer 0 or 1 if valid

4.39.1.5 int freePCB (pcb ∗ *pcbPtr*)

Frees memory that is allocated for the pcb provided

Parameters

pcbPtr	pointer to pcb to be freed

Returns

integer code - 1 if successful, 0 otherwise

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

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

Parameters

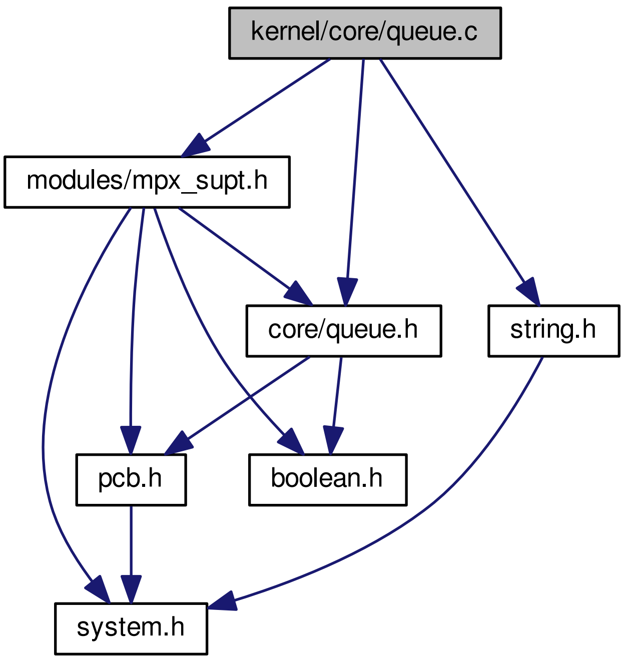
processName	- const string name
processClass	- integer identifying as system or application process (0, 1)
priority	- integer between 0 and 9 indicating priority

Returns

PCB pointer to new pcb or NULL if there were errors

4.40 kernel/core/queue.c File Reference

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



Enumerations

- enum queue { QUEUE_BLOCKED = BLOCKED, QUEUE_READY = READY, QUEUE_SUSPENDED_BLO←
CKED = BLOCKED + 0x02, QUEUE_SUSPENDED_READY = READY + 0x02 }

Functions

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

Variables

- node ∗ queues [4]

4.40.1 Enumeration Type Documentation

4.40.1.1 enum **queue**

Enumerator

QUEUE_BLOCKED

QUEUE_READY

QUEUE_SUSPENDED_BLOCKED

QUEUE_SUSPENDED_READY

4.40.2 Function Documentation

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

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

Parameters

processName	The process name to search for

Returns

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

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

Internal function for finding a node in a specific queue

Parameters

q	The queue to search in
processName	The process name to search for

Returns

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

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

Inserts a node into a FIFO queue

Parameters

q	The queue to insert into
newNode	The node to insert

Returns

true if the node was inserted, false otherwise

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

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

Parameters

q	The queue to insert into
newNode	The node to insert

Returns

true if the node was inserted, false otherwise

4.40.2.5 node ∗ _newNode (pcb ∗ *pcb*)

Internal function to create a new list node.

Parameters

pcb	The pcb to store in the node

Returns

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

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

Finds the PCB with the given process name.

Parameters

processName	The name of the process to search for

Returns

A pointer to the PCB, or null if not found

4.40.2.7 node∗ getBlockedQueue ( )

Gets the head node of the blocked queue.

Returns

The head node of the blocked queue

4.40.2.8 node∗ getReadyQueue ( )

Gets the head node of the ready queue.

Returns

The head node of the ready queue

4.40.2.9 node∗ getSuspendedBlockedQueue ( )

Gets the head node of the suspended-blocked queue.

Returns

The head node of the suspended-blocked queue

4.40.2.10 node∗ getSuspendedReadyQueue ( )

Gets the head node of the suspended-ready queue.

Returns

The head node of the suspended-ready queue

4.40.2.11 boolean insertPCB (pcb ∗ *p*)

Inserts the PCB into the appropriate queue.

Parameters

p	The PCB to insert.

Returns

true if the PCB was inserted, false otherwise

4.40.2.12 pcb∗ popBlocked ( )

Pops the next node off of the blocked queue.

Returns

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

4.40.2.13 pcb∗ popReady ( )

Pops the next node off of the ready queue.

Returns

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

4.40.2.14 pcb∗ popSuspendedBlocked ( )

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

Returns

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

4.40.2.15 pcb∗ popSuspendedReady ( )

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

Returns

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

4.40.2.16 boolean removePCB (pcb ∗ *p*)

Removes the given PCB from it's queue.

Parameters

p	The PCB to remove

Returns

true if the PCB was removed, false otherwise

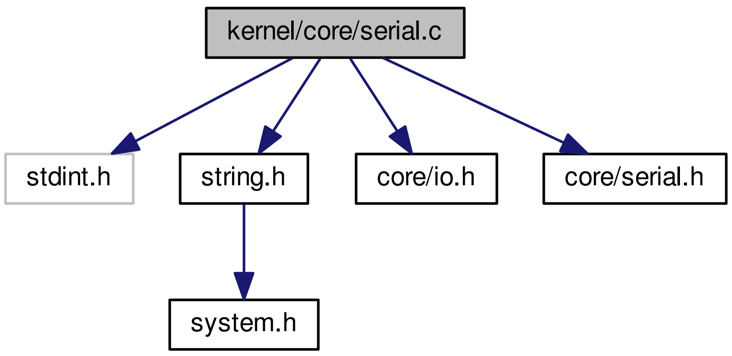
4.40.3 Variable Documentation

4.40.3.1 node∗ queues[4]

4.41 kernel/core/serial.c File Reference

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

Include dependency graph for serial.c:



Macros

- #define NO_ERROR 0

Functions

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

Variables

- int serial_port_out = 0
- int serial_port_in = 0

4.41.1 Macro Definition Documentation

4.41.1.1 #define NO_ERROR 0

4.41.2 Function Documentation

4.41.2.1 int init_serial (int *device*)

Initializes devices for user interaction, logging, ...

Parameters

device	The device to initialize

Returns

The error code

4.41.2.2 int serial_print (const char ∗ *msg*)

Writes a message to the active serial output device.

Parameters

msg	The message to write

Returns

The error code

4.41.2.3 int serial_println (const char ∗ *msg*)

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

Parameters

msg	The message to write

Returns

The error code

4.41.2.4 int set_serial_in (int *device*)

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

Parameters

device	The divce to set as input

Returns

The error code

4.41.2.5 int set_serial_out (int *device*)

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

Parameters

device	The device to set as output

Returns

The error code

4.41.3 Variable Documentation

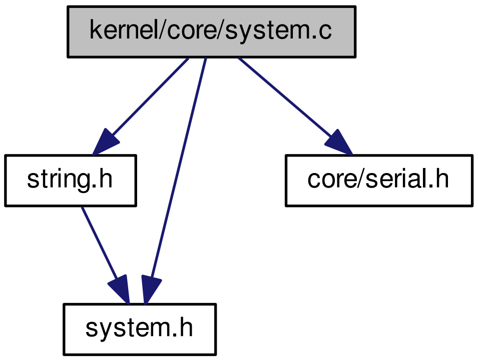
4.41.3.1 int serial_port_in = 0

4.41.3.2 int serial_port_out = 0

4.42 kernel/core/system.c File Reference

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

Include dependency graph for system.c:



Functions

- void klogv (const char ∗msg)
- void kpanic (const char ∗msg)

4.42.1 Function Documentation

4.42.1.1 void klogv (const char ∗ *msg*)

Kernel log message. Sent to active serial device.

Parameters

msg	The message to log

4.42.1.2 void kpanic (const char ∗ *msg*)

Kernel panic. Prints an error message and halts.

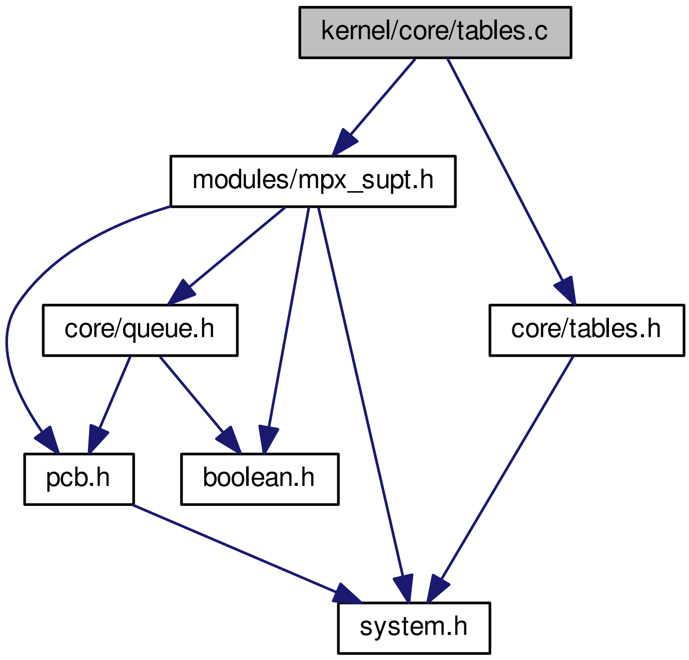
Parameters

msg	The error mesage to print

4.43 kernel/core/tables.c File Reference

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

Include dependency graph for tables.c:



Functions

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

Variables

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

4.43.1 Function Documentation

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

Installs a new table entry into the global descriptor table.

Parameters

idx	
base	
limit	
access	
flags	

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

Installs a new gate entry into the IDT.

Parameters

idx	
base	
sel	
flags	

4.43.1.3 void init_gdt ( )

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

4.43.1.4 void init_idt ( )

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

4.43.1.5 void write_gdt_ptr (u32int , size_t)

4.43.1.6 void write_idt_ptr (u32int)

4.43.2 Variable Documentation

4.43.2.1 gdt_entry gdt_entries[5]

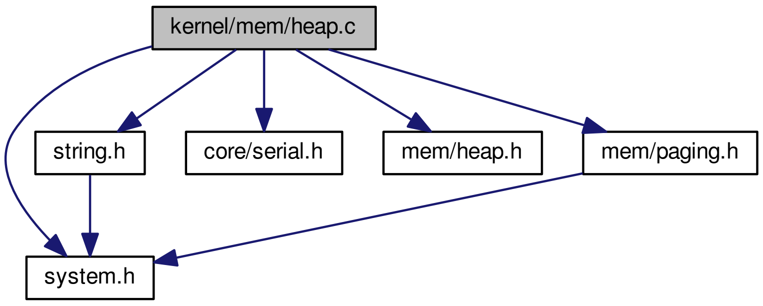
4.43.2.2 gdt_descriptor gdt_ptr

4.43.2.3 idt_entry idt_entries[256]

4.43.2.4 idt_descriptor idt_ptr

4.44 kernel/mem/heap.c File Reference

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



Functions

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

Variables

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

4.44.1 Function Documentation

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

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

Parameters

size	The amount of memory to allocate
align	The page alignment
phys_addr	The physical address

Returns

The memory address

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

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

Parameters

size	The amount of memory to allocate
hp	The heap to allocate on
align	The page alignment

Returns

The memory address

4.44.1.3 u32int kmalloc (u32int *size*)

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

Parameters

size	The amount of memory to allocate

Returns

The memory address

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

Create a new heap.

Parameters

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

Returns

The address of the heap

4.44.2 Variable Documentation

4.44.2.1 void __end

4.44.2.2 void _end

4.44.2.3 heap* curr_heap = 0

4.44.2.4 void* end

4.44.2.5 page_dir* kdir

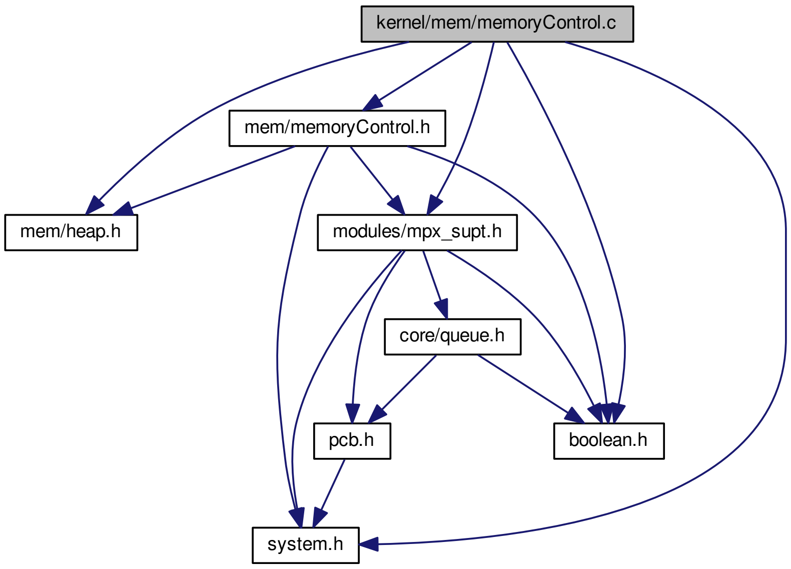
4.44.2.6 heap* kheap = 0

4.44.2.7 u32int phys_alloc_addr = (u32int) & end

4.45 kernel/mem/memoryControl.c File Reference

```
#include <system.h>
#include <mem/heap.h>
#include <mem/memoryControl.h>
#include <modules/mpx_supt.h>
#include <boolean.h>
```

Include dependency graph for memoryControl.c:



Functions

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

Variables

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

4.45.1 Function Documentation

4.45.1.1 void _mergeAdjacentFree ( )

Private helper function to merge all adjacent memory blocks

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

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

Parameters

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

Returns

pointer to create cmcb at beginning of memory block

4.45.1.3 void∗ allocateMemory (int *size*)

Allocates a memory block if enough memory is availabel

Parameters

size	- size of memory to allocate in bytes

Returns

pointer to the me

4.45.1.4 boolean deallocateMemory (void ∗ *memPointer*)

Deallocates the block of memory at the mempointer

Parameters

memPointer	- pointer to the mem block

Returns

boolean - boolean telling whether succesful dealloc

4.45.1.5 cmcb∗ getAllocatedHead ( )

Returns the head to the allocated list

Returns

cmcb ∗ to the allocated list head

4.45.1.6 cmcb∗ getFreeHead ( )

Returns the head of the free list

Returns

cmcb ∗ to the free list head

4.45.1.7 boolean initializeHeap (int *size*)

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

Parameters

size	- size of heap in bytes

Returns

boolean - boolean denoting if heap was initialized

4.45.1.8 boolean isEmpty ( )

Returns a boolean telling if all the memory is empty

Returns

boolean

4.45.2 Variable Documentation

4.45.2.1 cmcb∗ allocatedHead

4.45.2.2 cmcb∗ freeHead

4.45.2.3 int isInitialized = false

4.45.2.4 int memAllocated

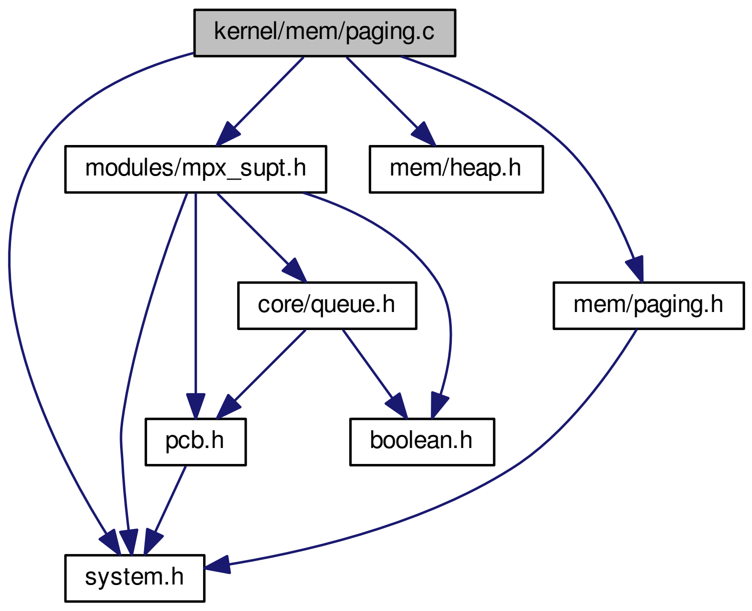
4.45.2.5 void∗ memHeap

4.45.2.6 int memSize

4.46 kernel/mem/paging.c File Reference

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

Include dependency graph for paging.c:



Functions

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

Variables

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

4.46.1 Function Documentation

4.46.1.1 void clear_bit (u32int *addr*)

Marks a page frame bit as free (0).

Parameters

addr	The address of the frame

4.46.1.2 u32int first_free ( )

Finds the first free page frame.

Returns

The first free page frame

4.46.1.3 u32int get_bit (u32int *addr*)

Checks if page frame is in use.

Parameters

addr	The address of the frame

Returns

True if it is in use

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

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

Parameters

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

Returns

A pointer to the page

4.46.1.5 void init_paging ( )

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

4.46.1.6 void load_page_dir (page_dir ∗ *new_dir*)

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

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

Parameters

new_page_dir	The page directory to set as the current

4.46.1.7 void new_frame (page_entry ∗ *page*)

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

Parameters

page	The page to create the frame in

4.46.1.8 void set_bit (u32int *addr*)

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

Parameters

addr	The address of the frame

4.46.2 Variable Documentation

4.46.2.1 page_dir∗ cdir = 0

4.46.2.2 u32int∗ frames

4.46.2.3 page_dir∗ kdir = 0

4.46.2.4 heap∗ kheap

4.46.2.5 u32int mem_size = 0x4000000

4.46.2.6 u32int nframes

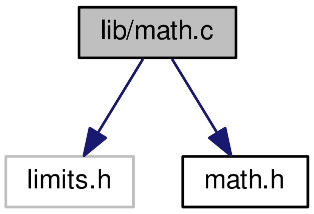
4.46.2.7 u32int page_size = 0x1000

4.46.2.8 u32int phys_alloc_addr

4.47 lib/math.c File Reference

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

Include dependency graph for math.c:



Functions

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

4.47.1 Function Documentation

4.47.1.1 unsigned char bcdToDec (unsigned char *bcd*)

Converts a BCD encoded byte to a decimal encoded byte

Parameters

bcd	The value to convert

Returns

The decimal value

4.47.1.2 unsigned char decToBcd (unsigned char *dec*)

Converts a decimal encoded byte to a BCD encoded byte

Parameters

dec	The value to convert

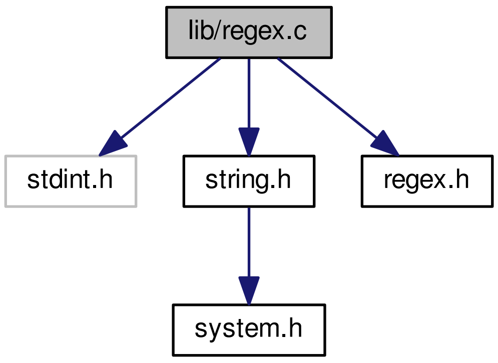
Returns

The BCD value

4.48 lib/regex.c File Reference

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

Include dependency graph for regex.c:



Functions

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

4.48.1 Function Documentation

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

Tests if the stringToCheck adheres to the given regex string

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

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

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

Parameters

regex	
stringToCheck	

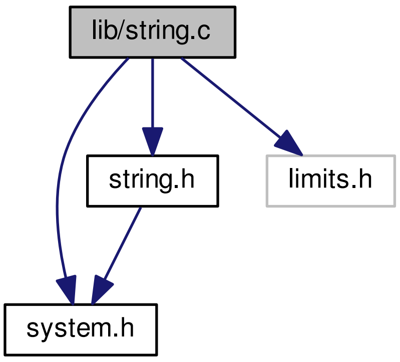
Returns

1 if adheres, 0 otherwise

4.49 lib/string.c File Reference

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

Include dependency graph for string.c:



Functions

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

4.49.1 Function Documentation

4.49.1.1 int atoi (const char $*$ *s*)

Convert an ASCII string to an integer

Parameters

s	The string to convert

Returns

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

4.49.1.2 int isChar (const char *c*)

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

Parameters

const	char c

Returns

1 if c is a char, 0 otherwise

4.49.1.3 int isdigit (const char *c*)

Determine if a character is a digit.

Parameters

c	The character to check

Returns

True if the character is a digit

4.49.1.4 int isLowerChar (const char *c*)

Checks if the given char is a-z

Parameters

const	char c

Returns

1 if c is a lower char, 0 otherwise

4.49.1.5 int isspace (const char ∗ *c*)

Determine if a character is whitespace.

Parameters

c	The character to check

Returns

True if the character is a whitespace character

4.49.1.6 int isUpperChar (const char *c*)

Checks if the given char is A-Z

Parameters

const	char c

Returns

1 if c is a upper char, 0 otherwise

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

Converts an integer to an ASCII string.

Parameters

num	The number to convert
str	The destination string
base	The radix

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

Reverses a string.

Parameters

str	The string to reverse
len	The length of the string

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

Concatenate the contents of one string onto another.

Parameters

s1	The destination string
s2	The source string

Returns

A pointer to the destination string

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

Compares two strings to each other

Parameters

s1	The first string
s2	The second string

Returns

The difference between the characters at the first index of indifference

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

Copy on string to another.

Parameters

cpy	The destination string
ori	The source string

Returns

A pointer to the destination string

4.49.1.12 int strlen (const char ∗ *s*)

Returns the length of a string.

Parameters

s	The input string

Returns

The length of the string

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

Split string into tokens

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

Parameters

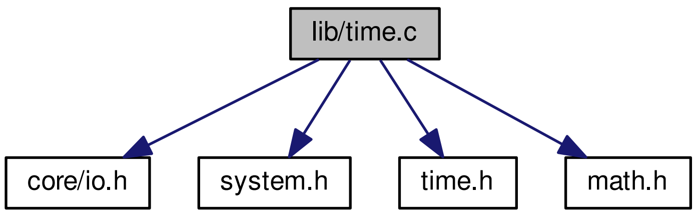
s1	The string to split
s2	The delimiter

Returns

A single token

4.50 lib/time.c File Reference

```
#include <core/io.h>
#include <system.h>
#include <time.h>
#include <math.h>
```
Include dependency graph for time.c:



Functions

- date_time getDateTime ()

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

Variables

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

4.50.1 Function Documentation

4.50.1.1 date_time getDateTime ( )

Gets the date and time from the RTC registers.

Returns

The date and time stored in the RTC.

4.50.1.2 unsigned char getDayOfMonth ( )

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

Returns

The decimal-encoded day of the month.

4.50.1.3 unsigned char getDayOfWeek ( )

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

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

Returns

The decimal-encoded day of the week.

4.50.1.4 unsigned char getHours ( )

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

Returns

The decimal-encoded number of hours.

4.50.1.5 unsigned char getMinutes ( )

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

Returns

The decimal-encoded number of minutes.

4.50.1.6 unsigned char getMonth ( )

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

Returns

The decimal-encoded month.

4.50.1.7 unsigned char getSeconds ( )

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

Returns

The decimal-encoded number of seconds.

4.50.1.8 unsigned char getYear ( )

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

Returns

The decimal-encoded year.

4.50.1.9 int isLeapYear (int *year*)

Determines if the given year is a leap year.

Parameters

year	The year to check

Returns

True if the year is a leap year.

4.50.1.10 void setDateTime (date_time *dateTime*)

Sets the date and time to the specified values.

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

Parameters

dateTime	The values to set.

4.50.1.11 void setDayOfMonth (unsigned char *day*)

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

Parameters

dayOfMonth	The day of the month value to set

4.50.1.12 void setDayOfWeek (unsigned char *day*)

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

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

Parameters

dayOfWeek	The day of the week value to set

4.50.1.13 void setHours (unsigned char *hour*)

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

Parameters

hours	The hours value to set

4.50.1.14 void setMinutes (unsigned char *min*)

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

Parameters

minutes	The minutes value to set

4.50.1.15 void setMonth (unsigned char *mon*)

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

Parameters

month	The month value to set

4.50.1.16 void setSeconds (unsigned char *sec*)

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

Parameters

seconds	The seconds value to set

4.50.1.17 void setYear (unsigned char *year*)

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

Parameters

year	The year value to set

4.50.1.18 void updateDayOfWeek (date_time ∗ *dateTime*)

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

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

Parameters

dateTime	The date_time to update.

4.50.1.19 void updateDayOfYear (date_time ∗ *dateTime*)

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

Parameters

dateTime	The date_time to update.

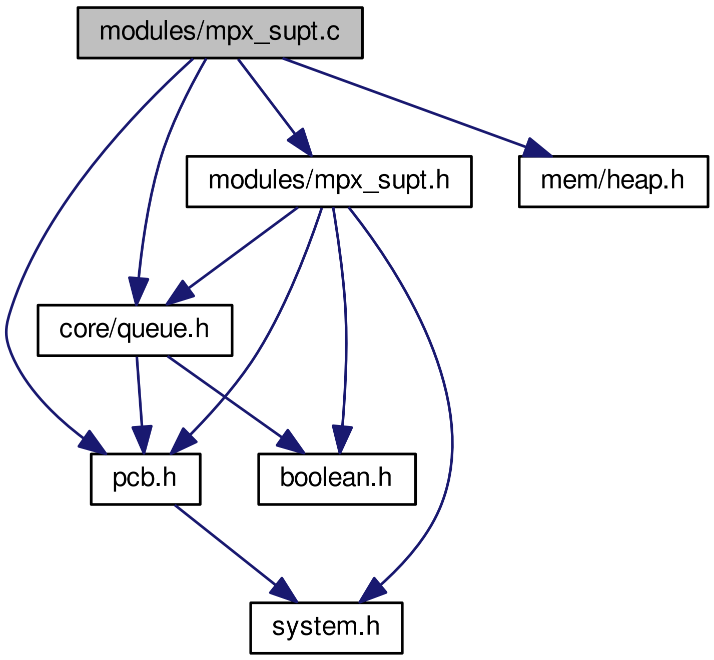
4.50.2 Variable Documentation

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

4.51 modules/mpx_supt.c File Reference

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

Include dependency graph for mpx_supt.c:



Functions

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

Variables

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

4.51.1 Function Documentation

4.51.1.1 const char∗ getCOPName ( )

Gets the name of the COP

Returns

const char pointer name

4.51.1.2 void idle ( )

The idle process

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

Set a region of memory

Parameters

s	Destination
c	Byte to write
n	Count

Returns

s

4.51.1.4 void mpx_init (int *cur_mod*)

Initialize MPX support software

Parameters

cur_mod	(symbolic constants MODULE_R1, MODULE_R2, etc)

4.51.1.5 void∗ sys_alloc_mem (u32int *size*)

Allocates a block of memory (similar to malloc)

Parameters

size	Number of bytes to allocate

Returns

The allocated memory

4.51.1.6 u32int∗ sys_call (context ∗ *registers*)

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

Parameters

registers	- copy of register values

Returns

u32int position of stackTop

4.51.1.7 int sys_free_mem (void ∗ *ptr*)

Frees memory

Parameters

ptr	Pointer to the block of memory to free

Returns

4.51.1.8 int sys_req (int *op_code*)

Generates interrupt 60H

Parameters

op_code	(IDLE)

Returns

0

4.51.1.9 void sys_set_free (boolean(func)(void ∗))

Sets the memory free function for sys_free_mem

Parameters

func	Function pointer to the memory free-er

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

Sets the memory allocation function for sys_alloc_mem

Parameters

func	Function pointer to the memory allocator

4.51.2 Variable Documentation

4.51.2.1 context∗ callerContext

4.51.2.2 pcb∗ cop = NULL

4.51.2.3 int current_module = -1

4.51.2.4 param params

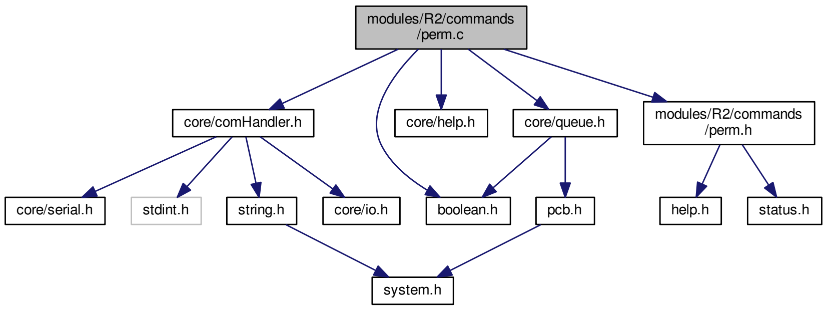
4.51.2.5 boolean(∗ student_free) (void ∗)

4.51.2.6 void∗(∗ student_malloc) (int)

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

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

Include dependency graph for perm.c:



Functions

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

4.52.1 Function Documentation

4.52.1.1 void printPcbInfo (pcb ∗ *p*)

4.52.1.2 void printQueueInfo (node ∗ *queue*)

4.52.1.3 void registerR2PermCommands ( )

Registers the permanent commands in the command handler

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

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

Usage: rpcb name

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

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

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

Usage: ppcb name priority

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

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

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

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

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

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

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

Usage: spcb name

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

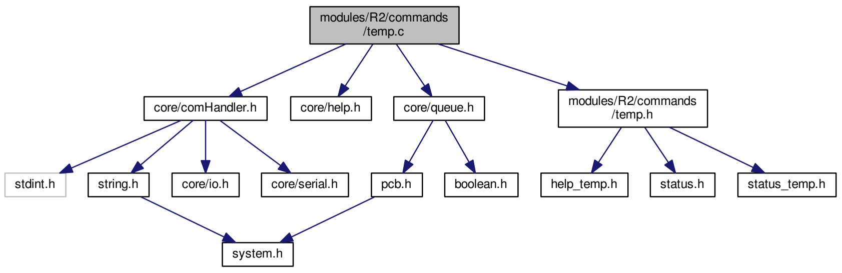
Returns

A status message indicating success/failure

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

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

Include dependency graph for temp.c:



Functions

- void registerR2TempCommands ()
- const char ∗ createPcb (char ∗∗args, int numArgs)
- const char ∗ deletePcb (char ∗∗args, int numArgs)
- const char ∗ blockPcb (char ∗∗args, int numArgs)
- const char ∗ unblockPcb (char ∗∗args, int numArgs)

4.53.1 Function Documentation

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

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

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

Usage: bpcb name

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

4.53.1.2 const char∗ createPcb (char ∗∗ *args*, int *numArgs*)

Creates a PCB and inserts it into the appropriate queue.

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

Usage: cpcb name class priority

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

4.53.1.3 const char∗ deletePcb (char ∗∗ *args*, int *numArgs*)

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

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

Usage: dpcb name

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

Returns

A status message indicating success/failure

4.53.1.4 void registerR2TempCommands ( )

Registers the temporary commands in the command handler

4.53.1.5 const char∗ unblockPcb (char ∗∗ *args*, int *numArgs*)

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

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

Usage: upcb name

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

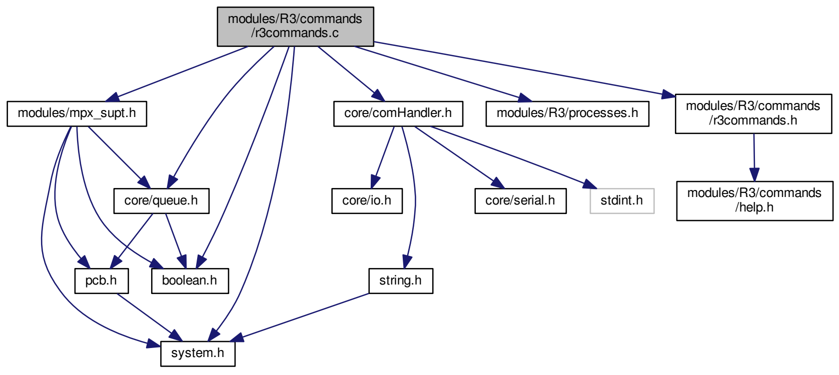
Returns

A status message indicating success/failure

4.54 modules/R3/commands/r3commands.c File Reference

```
#include <system.h>
#include <boolean.h>
#include <core/queue.h>
#include <core/comHandler.h>
#include <modules/R3/processes.h>
#include <modules/R3/commands/r3commands.h>
#include <modules/mpx_supt.h>
```

Include dependency graph for r3commands.c:



Macros

- #define P1_NAME ((const char∗) "r3p1")
- #define P2_NAME ((const char∗) "r3p2")
- #define P3_NAME ((const char∗) "r3p3")
- #define P4_NAME ((const char∗) "r3p4")
- #define P5_NAME ((const char∗) "r3p5")

Functions

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

4.54.1 Macro Definition Documentation

4.54.1.1 #define P1_NAME ((const char∗) "r3p1")

4.54.1.2 #define P2_NAME ((const char∗) "r3p2")

4.54.1.3 #define P3_NAME ((const char∗) "r3p3")

4.54.1.4 #define P4_NAME ((const char∗) "r3p4")

4.54.1.5 #define P5_NAME ((const char∗) "r3p5")

4.54.2 Function Documentation

4.54.2.1 const char∗ loadr3 (char ∗∗ *args,* int *numArgs*)

Loads the r3 processes to the queue.
")

Usage: loadr3

Args: [no args] - loads processes

Parameters

args	The arguments to pass to the function

Returns

""

4.54.2.2 void registerR3Commands ( )

Registers commands in command handler

4.54.2.3 const char∗ yield (char ∗∗ *args,* int *numArgs*)

Yields command handler to allow other processes to run.

Usage: yield

Args: [no args] - yields command handler

Parameters

args	The arguments to pass to the function

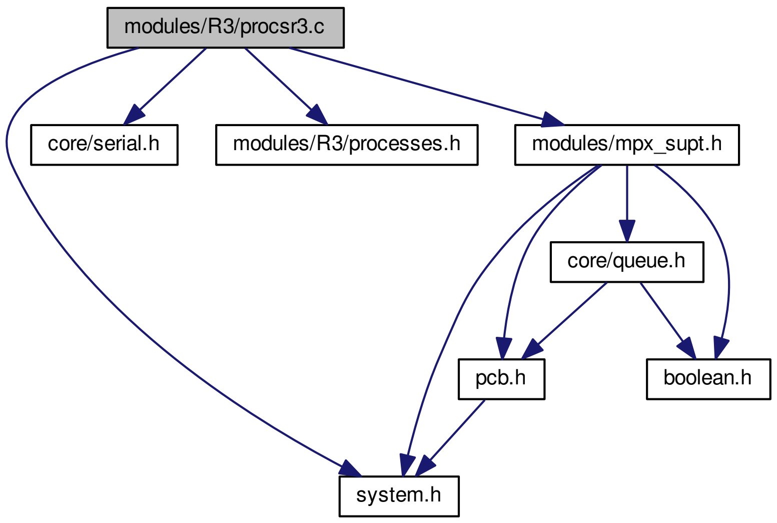
Returns

""

4.55 modules/R3/procsr3.c File Reference

```
#include <system.h>
#include <core/serial.h>
#include <modules/R3/processes.h>
#include <modules/mpx_supt.h>
```

Include dependency graph for procsr3.c:



Macros

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

Functions

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

4.55.1 Macro Definition Documentation

4.55.1.1 #define RC_1 1

4.55.1.2 #define RC_2 2

4.55.1.3 #define RC_3 3

4.55.1.4 #define RC_4 4

4.55.1.5 #define RC_5 5

4.55.2 Function Documentation

4.55.2.1 void proc1 ( )

4.55.2.2 void proc2 ( )

4.55.2.3 void proc3 ( )

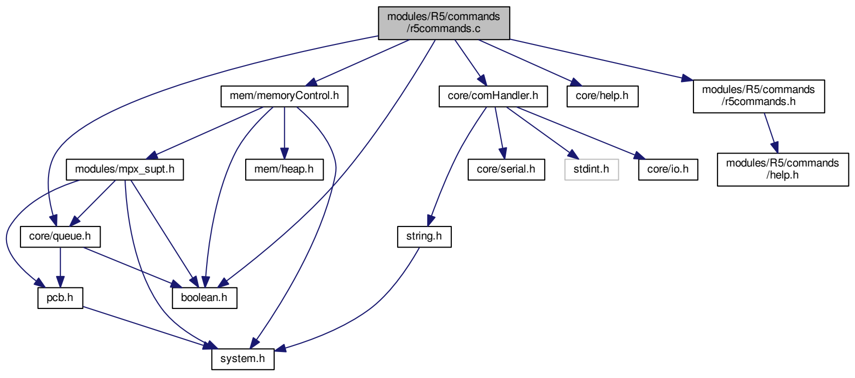
4.55.2.4 void proc4 ( )

4.55.2.5 void proc5 ( )

4.56 modules/R5/commands/r5commands.c File Reference

```
#include <boolean.h>
#include <core/comHandler.h>
#include <core/help.h>
#include <core/queue.h>
#include <modules/R5/commands/r5commands.h>
#include <mem/memoryControl.h>
```

Include dependency graph for r5commands.c:



Functions

- void printBlockInfo (cmcb ∗blockList)
- void printCmcbInfo (cmcb ∗block)
- void registerR5PermCommands ()
- const char ∗ showMemory (char ∗∗args, int numArgs)

4.56.1 Function Documentation

4.56.1.1 void printBlockInfo (cmcb ∗ *blockList*)

4.56.1.2 void printCmcbInfo (cmcb ∗ *block*)

4.56.1.3 void registerR5PermCommands ( )

Registers the permanent commands in the command handler

4.56.1.4 const char∗ showMemory (char ∗∗ *args,* int *numArgs*)

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

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

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

Parameters

args	The arguments to pass to the function
numArgs	The number of arguments

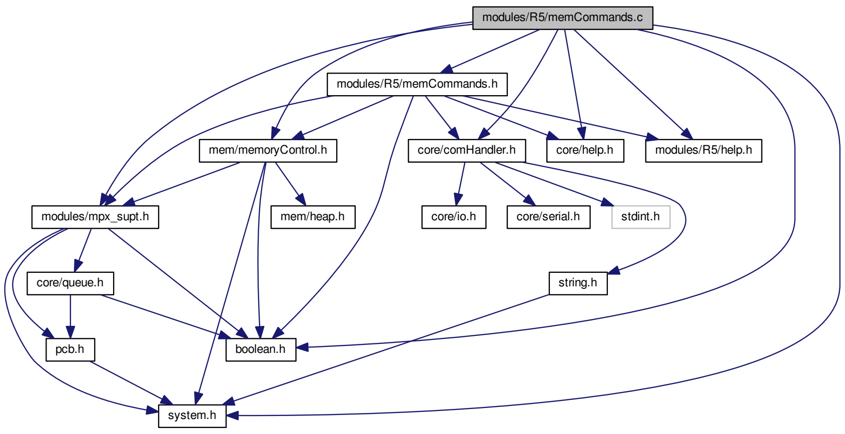
Returns

A status message indicating success/failure

4.57 modules/R5/memCommands.c File Reference

```
#include <mem/memoryControl.h>
#include <boolean.h>
#include <core/comHandler.h>
#include <core/help.h>
#include <modules/mpx_supt.h>
#include <modules/R5/memCommands.h>
#include <modules/R5/help.h>
#include <system.h>
```

Include dependency graph for memCommands.c:



Functions

- void registerR5TempCommands ()
- const char ∗ initHeap (char ∗∗args, int numArgs)
- const char ∗ allocateMem (char ∗∗args, int numArgs)
- const char ∗ freeMemory (char ∗∗args, int numArgs)
- const char ∗ isEmptyCom (char ∗∗args, int numArgs)

4.57.1 Function Documentation

4.57.1.1 const char∗ allocateMem (char ∗∗ *args*, int *numArgs*)

Allocates a memory block if enough memory is available

Parameters

size	- size of memory to allocate in bytes

Returns

pointer to the me

4.57.1.2 const char∗ freeMemory (char ∗∗ *args*, int *numArgs*)

Deallocates the block of memory at the mempointer

Parameters

memPointer	- pointer to the mem block

4.57.1.3 const char∗ initHeap (char ∗∗ *args,* int *numArgs*)

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

Returns

True or false

4.57.1.4 const char∗ isEmptyCom (char ∗∗ *args,* int *numArgs*)

Check if memory is empty

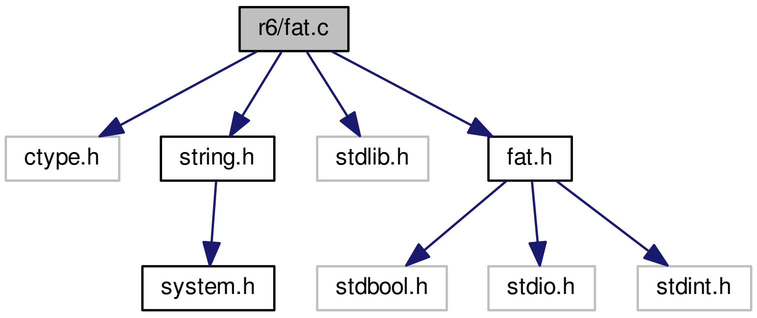
4.57.1.5 void registerR5TempCommands ( )

Registers the permanent commands in the command handler

4.58 r6/fat.c File Reference

```
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#include "fat.h"
```

Include dependency graph for fat.c:



Functions

- void _loadBootSectorInfo ()
- void _loadFATTables ()
- void _loadRootDirectroy ()
- dir_entry ∗ _loadSectorAsDirectoryEntries (uint16_t sector)
- void _readDirectoryEntry (dir_entry ∗dir)
- long _getDiskOffsetForDirEntry (int idx)
- void _saveFATTables ()
- void _saveDirEntry (dir_entry ∗dir)
- void _refreshDirectory ()
- void _prepNewDirSector (uint16_t sector)
- int _getFirstFreeIndexInSector (dir_entry ∗dirs)
- int _getFirstFreeIndexInDirs (dir_entry ∗dirs, int maxSize)
- uint16_t _getFirstOpenSector ()
- void initialize (FILE ∗diskImage)
- void destroy ()
- boot_sector ∗ getBootSector ()
- fat_tables ∗ getFATTables ()
- dir_entry ∗ getCurrentDirectory ()
- int getCurrentDirectoryMaxSize ()
- unsigned char ∗ getFileFromSector (uint16_t sector, int size)
- int changeToDirectory (uint16_t sector)
- int changeToParentDirectory ()
- void setFilename (int idx, const char ∗filename, const char ∗fileExt)
- bool moveFile (int idx, uint16_t destSector)

Variables

- FILE ∗ _DiskImage
- boot_sector _BootSector
- fat_tables _FATTables
- dir_entry ∗ _CurrentDirectory
- int _CurrDirSize = 0
- bool _isCurrentRoot = false

4.58.1 Function Documentation

4.58.1.1 long _getDiskOffsetForDirEntry (int *idx*)

Gets the disk offset (in bytes) of the file at the given index of the current directory.

Parameters

idx	The index of the current directory

Returns

The disk offset in byte

4.58.1.2 int _getFirstFreeIndexInDirs (dir_entry ∗ *dirs*, int *maxSize*)

Gets the first free directory entry index in the given array of entries. Supports a variable max size.

Parameters

dirs	The array on entries to search through
maxSize	The size of the array

Returns

The index of the first free directory entry

4.58.1.3 int _getFirstFreeIndexInSector (dir_entry ∗ *dirs*)

Gets the first free directory entry index in the given array of entries.

Parameters

dirs	The array on entries to search through

Returns

The index of the first free directory entry

4.58.1.4 uint16_t _getFirstOpenSector ( )

Gets the first open sector.

Returns

The ID of the first empty sector

4.58.1.5 void _loadBootSectorInfo ( )

Loads boot sector information.

4.58.1.6 void _loadFATTables ( )

Loads the FAT table information

4.58.1.7 void _loadRootDirectroy ( )

Loads the root directory as the current directory.

4.58.1.8 dir_entry ∗ _loadSectorAsDirectoryEntries (uint16_t *sector*)

Loads a sector as an array of (bytesPerSector / DIR_ENTRY_SIZE) directory entries.

Parameters

sector	The sector to load from.

Returns

An array of directory entries.

4.58.1.9 void _prepNewDirSector (uint16_t *sector*)

Preps the given sector to be a new directory sector.

Parameters

sector	The sector ID to set

4.58.1.10 void _readDirectoryEntry (dir_entry ∗ *dir*)

Reads a single directory entry into the passed entry.

Parameters

The	directory entry to store the information in

4.58.1.11 void _refreshDirectory ( )

Refreshes (reloads) the current directory.

4.58.1.12 void _saveDirEntry (dir_entry ∗ *dir*)

Saves the passed directory entry to the disc.

4.58.1.13 void _saveFATTables ( )

Saves the FAT Tables to the disc.

4.58.1.14 int changeToDirectory (uint16_t *sector*)

Changes to the directory specified by the given logical sector.

Parameters

sector	The starting logical sector of the directory.

Returns

The maximum number of entries that can appear in this directory.

4.58.1.15 int changeToParentDirectory ( )

Changes the current directory to the parent of the current.

Returns

The maximum size of the new current directory, or -1 if the directory didn't switch.

4.58.1.16 void destroy ( )

Destroys the FAT abstraction, freeing any memory used internally.

4.58.1.17 boot_sector∗ getBootSector ( )

Gets the boot sector of the FAT File System.

Returns

The boot sector of the FAT File System

4.58.1.18 dir_entry∗ getCurrentDirectory ( )

Gets the current directory of the FAT File System.

Returns

The current directory as an array of directory entries.

4.58.1.19 int getCurrentDirectoryMaxSize ( )

Gets the maximum size of the current directory.

Returns

The maximum size of the current directory.

4.58.1.20 fat_tables∗ getFATTables ( )

Gets the FAT Tables for the FAT File System.

Returns

The FAT Tables for the FAT File System

4.58.1.21 unsigned char∗ getFileFromSector (uint16_t *sector,* int *size*)

Gets the file specified by the given sector.

Parameters

sector	The starting logical sector of the file
size	The size of the file (in bytes)

Returns

An array of bytes representing the file.

4.58.1.22 void initialize (FILE ∗ *diskImage*)

Initializes the FAT abstraction with the given disk image.

Parameters

diskImage	The pointer to an opened file that is a FAT12 disk image.

4.58.1.23 bool moveFile (int *idx,* uint16_t *destSector*)

Moves a file at the specified index of the current directory to the directory at the destination sector.

Parameters

idx	The index of the listing in the current direcory
destSector	The beginning sector of the target directory

4.58.1.24 void setFilename (int *idx,* const char ∗ *filename,* const char ∗ *fileExt*)

Sets the name of the file at the specified index of the current list of directory entries.

Parameters

idx	The index of the listing in the current directory
filename	The name of the file (max 8 characters)
fileExt	The extension of the file (max 3 characters)

4.58.2 Variable Documentation

4.58.2.1 boot_sector _BootSector

4.58.2.2 int _CurrDirSize = 0

4.58.2.3 dir_entry∗ _CurrentDirectory

4.58.2.4 FILE∗ _DiskImage

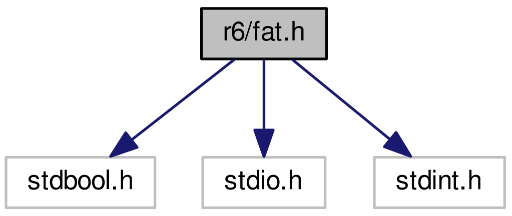
4.58.2.5 fat_tables _FATTables

4.58.2.6 bool _isCurrentRoot = false

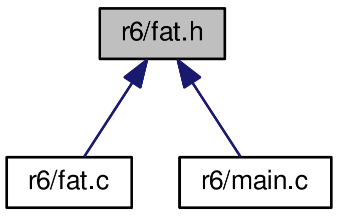
4.59 r6/fat.h File Reference

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

Include dependency graph for fat.h:



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



Data Structures

- struct boot_sector
- struct dir_entry
- struct fat_tables

Macros

- #define BOOT_SECTOR_OFFSET 0
- #define FAT1_OFFSET 1
- #define FAT2_OFFSET 10
- #define ROOT_DIRECTORY_OFFSET 19
- #define DATA_AREA_OFFSET 33
- #define UNUSED 0x00
- #define RESERVED_CLUSTER_BEGIN 0xFF0
- #define RESERVED_CLUSTER_END 0xFF6
- #define BAD_CLUSTER 0xFF7
- #define LAST_CLUSTER_BEGIN 0xFF8
- #define LAST_CLUSTER_END 0xFFF
- #define READ_ONLY 0x01
- #define HIDDEN 0x02
- #define SYSTEM 0x04
- #define VOLUME_LABEL 0x08
- #define SUBDIRECTORY 0x10
- #define ARCHIVE 0x20
- #define DIR_ENTRY_SIZE 32
- #define MAX_FILENAME_LENGTH 8
- #define MAX_EXT_LENGTH 3
- #define DELETED 0xE5
- #define REMAINING_FREE 0x00

Functions

- void initialize (FILE ∗diskImage)
- void destroy ()
- boot_sector ∗ getBootSector ()
- fat_tables ∗ getFATTables ()
- dir_entry ∗ getCurrentDirectory ()
- int getCurrentDirectoryMaxSize ()
- unsigned char ∗ getFileFromSector (uint16_t cluster, int size)
- int changeToDirectory (uint16_t cluster)
- int changeToParentDirectory ()
- void setFilename (int idx, const char ∗filename, const char ∗fileExt)
- bool moveFile (int idx, uint16_t destSector)

4.59.1 Macro Definition Documentation

4.59.1.1 #define ARCHIVE 0x20

4.59.1.2 #define BAD_CLUSTER 0xFF7

4.59.1.3 #define BOOT_SECTOR_OFFSET 0

4.59.1.4 #define DATA_AREA_OFFSET 33

4.59.1.5 #define DELETED 0xE5

4.59.1.6 #define DIR_ENTRY_SIZE 32

4.59.1.7 #define FAT1_OFFSET 1

4.59.1.8 #define FAT2_OFFSET 10

4.59.1.9 #define HIDDEN 0x02

4.59.1.10 #define LAST_CLUSTER_BEGIN 0xFF8

4.59.1.11 #define LAST_CLUSTER_END 0xFFF

4.59.1.12 #define MAX_EXT_LENGTH 3

4.59.1.13 #define MAX_FILENAME_LENGTH 8

4.59.1.14 #define READ_ONLY 0x01

4.59.1.15 #define REMAINING_FREE 0x00

4.59.1.16 #define RESERVED_CLUSTER_BEGIN 0xFF0

4.59.1.17 #define RESERVED_CLUSTER_END 0xFF6

4.59.1.18 #define ROOT_DIRECTORY_OFFSET 19

4.59.1.19 #define SUBDIRECTORY 0x10

4.59.1.20 #define SYSTEM 0x04

4.59.1.21 #define UNUSED 0x00

4.59.1.22 #define VOLUME_LABEL 0x08

4.59.2 Function Documentation

4.59.2.1 int changeToDirectory (uint16_t *sector*)

Changes to the directory specified by the given logical cluster.

Parameters

cluster	The starting logical cluster of the directory.

Returns

The maximum number of entries that can appear in this directory.

Changes to the directory specified by the given logical sector.

Parameters

sector	The starting logical sector of the directory.

Returns

The maximum number of entries that can appear in this directory.

4.59.2.2 int changeToParentDirectory ( )

Changes the current directory to the parent of the current.

Returns

The maximum size of the new current directory.

Changes the current directory to the parent of the current.

Returns

The maximum size of the new current directory, or -1 if the directory didn't switch.

4.59.2.3 void destroy ( )

Destroys the FAT abstraction, freeing any memory used internally.

4.59.2.4 boot_sector∗ getBootSector ( )

Gets the boot sector of the FAT File System.

Returns

The boot sector of the FAT File System

4.59.2.5 dir_entry∗ getCurrentDirectory ( )

Gets the current directory of the FAT File System.

Returns

The current directory as an array of directory entries.

4.59.2.6 int getCurrentDirectoryMaxSize ( )

Gets the maximum size of the current directory.

Returns

The maximum size of the current directory.

4.59.2.7 fat_tables∗ getFATTables ( )

Gets the FAT Tables for the FAT File System.

Returns

The FAT Tables for the FAT File System

4.59.2.8 unsigned char∗ getFileFromSector (uint16_t *sector,* int *size*)

Gets the file specified by the given cluster.

Parameters

cluster	The starting logical cluster of the file
size	The size of the file (in bytes)

Returns

An array of bytes representing the file.

Gets the file specified by the given sector.

Parameters

sector	The starting logical sector of the file
size	The size of the file (in bytes)

Returns

An array of bytes representing the file.

4.59.2.9 void initialize (FILE ∗ *diskImage*)

Initializes the FAT abstraction with the given disk image.

Parameters

diskImage	The pointer to an opened file that is a FAT12 disk image.

4.59.2.10 bool moveFile (int *idx*, uint16_t *destSector*)

Moves a file at the specified index of the current directory to the directory at the destination sector.

Parameters

idx	The index of the listing in the current direcory
destSector	The beginning sector of the target directory

4.59.2.11 void setFilename (int *idx*, const char ∗ *filename*, const char ∗ *fileExt*)

Sets the name of the file at the specified index of the current list of directory entries.

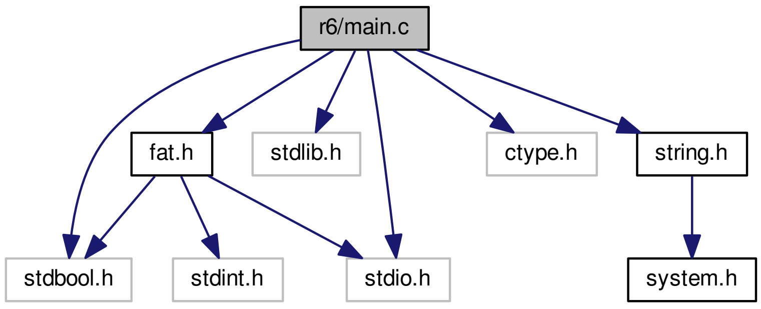
Parameters

idx	The index of the listing in the current directory
filename	The name of the file (max 8 characters)
fileExt	The extension of the file (max 3 characters)

4.60 r6/main.c File Reference

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "fat.h"
```

Include dependency graph for main.c:



Functions

- void _launchCommandInterface ()
- void _printBootSectorInfo ()
- void _printFATTableInfo ()
- void _printDirectoryEntries (dir_entry ∗entries, int maxEntries)
- void _printDirectoryEntriesByType (dir_entry ∗entries, int maxEntries, char ∗ext)
- void _printDirectoryEntriesByFileName (dir_entry ∗entries, int maxEntries, char ∗name, char ∗fileExt)
- int _getClusterOfFileWithName (const char ∗name, const char ∗ext)
- int _getSizeOfFileWithName (const char ∗name, const char ∗ext)
- void _printFile (uint16_t sector, int fileSize, bool pag)
- int _getIndexOfFileWithName (const char ∗name, const char ∗ext)
- bool _fncmp (const char ∗n1, const char ∗n2)
- bool _extcmp (const char ∗n1, const char ∗n2)
- bool _nameCmpHelper (const char ∗n1, const char ∗n2, int maxElements)
- void _callCommand (char ∗command)
- int main (int numArgs, char ∗args[ ])

Variables

- char ∗ paths [256]
- int depth = 0
- const char ∗ imageName = ""
- bool printFileFlag = false
- char ∗ filename = ""
- FILE ∗ diskImage

4.60.1 Function Documentation

4.60.1.1 void _callCommand (char ∗ *command*)

Handles parsing of command and calling of correct operations

4.60.1.2 bool _extcmp (const char ∗ *n1,* const char ∗ *n2*)

Compares the extension with the given extension to determine equality. // TODO: Words are hard

Parameters

n1	The first name of the comparison (no null terminator)
n2	The second name of the comparison (null terminated)

Returns

True, if the names are equal

4.60.1.3 bool _fncmp (const char ∗ *n1,* const char ∗ *n2*)

Compares the filename with the given filename to determine equality. // TODO: Words are hard

Parameters

n1	The first name of the comparison (no null terminator)
n2	The second name of the comparison (null terminated)

Returns

True, if the names are equal

4.60.1.4 int _getClusterOfFileWithName (const char ∗ *name*, const char ∗ *ext*)

Gets the cluster of the file with the given name

4.60.1.5 int _getIndexOfFileWithName (const char ∗ *name*, const char ∗ *ext*)

Gets the index of the file with the specified name and extension

4.60.1.6 int _getSizeOfFileWithName (const char ∗ *name*, const char ∗ *ext*)

Gets the size of the file with the given name

4.60.1.7 void _launchCommandInterface ( )

Starts the interactive shell session.

4.60.1.8 bool _nameCmpHelper (const char ∗ *n1*, const char ∗ *n2*, int *maxElements*)

Compares two string name with up to 'maxElements' number of characters. // TODO: Words are hard

Parameters

n1	The first name of the comparison (no null terminator)
n2	The second name of the comparison (null terminated)
maxElements	The number of elements to check

Returns

True, if the names are equal

4.60.1.9 void _printBootSectorInfo ( )

Prints information for the boot sector

4.60.1.10 void _printDirectoryEntries (dir_entry ∗ *entries,* int *maxEntries*)

Prints all directory entries in current dir

4.60.1.11 void _printDirectoryEntriesByFileName (dir_entry ∗ *entries,* int *maxEntries,* char ∗ *name,* char ∗ *fileExt*)

Prints any files with given name

4.60.1.12 void _printDirectoryEntriesByType (dir_entry ∗ *entries,* int *maxEntries,* char ∗ *ext*)

Prints all directory entries in the current dir with the provided extension

4.60.1.13 void _printFATTableInfo ( )

Prints fat table information

4.60.1.14 void _printFile (uint16_t *sector,* int *fileSize,* bool *pag*)

Prints contents of given file

4.60.1.15 int main (int *numArgs,* char ∗ *args[ ]*)

4.60.2 Variable Documentation

4.60.2.1 int depth = 0

4.60.2.2 FILE∗ diskImage

4.60.2.3 char∗ filename = ""

4.60.2.4 const char∗ imageName = ""

4.60.2.5 char∗ paths[256]

4.60.2.6 bool printFileFlag = false

Index