MPX_CORE GROUP 9

Version R2
02/23/2021

Table of Contents

MPX_Core Project

This project is about building a primitive operating system that includes a command line interface, process management and memory management

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

File List

Here is a list of all files with brief descriptions:

Data Structure Documentation

date_time Struct Reference

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

Data Fields

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

Field Documentation

int day_m

int day_w

int day_y

int hour

int min

int mon

int sec

int year

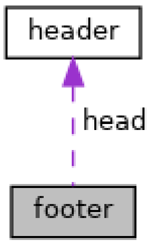
The documentation for this struct was generated from the following file:

include/**system.h**

footer Struct Reference

`#include <heap.h>`

Collaboration diagram for footer:

IMAGE 

Data Fields

header head

Field Documentation

header head

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

include/mem/**heap.h**

gdt_descriptor_struct Struct Reference

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

Data Fields

u16int limit
u32int base

Field Documentation

u32int base

u16int limit

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

include/core/**tables.h**

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

Field Documentation

u8int access

u8int base_high

u16int base_low

u8int base_mid

u8int flags

u16int limit_low

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

include/core/**tables.h**

header Struct Reference

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

Data Fields

int **size**
int **index_id**

Field Documentation

int index_id

int size

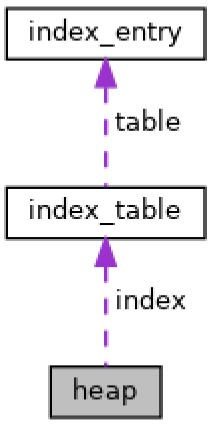
The documentation for this struct was generated from the following file:

include/mem/**heap.h**

heap Struct Reference

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

Collaboration diagram for heap:



IMAGE

Data Fields

index_table index
u32int base
u32int max_size
u32int min_size

Field Documentation

u32int base

index_table index

u32int max_size

u32int min_size

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

include/mem/**heap.h**

idt_entry_struct Struct Reference

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

Data Fields

u16int base_low
u16int sselect
u8int zero
u8int flags
u16int base_high

Field Documentation

u16int base_high

u16int base_low

u8int flags

u16int sselect

u8int zero

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

include/core/**tables.h**

idt_struct Struct Reference

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

Data Fields

u16int limit
u32int base

Field Documentation

u32int base

u16int limit

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

include/core/**tables.h**

index_entry Struct Reference

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

Data Fields

int **size**
int **empty**
u32int block

Field Documentation

u32int block

int empty

int size

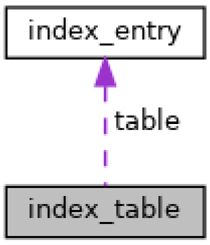
The documentation for this struct was generated from the following file:

include/mem/**heap.h**

index_table Struct Reference

`#include <heap.h>`

Collaboration diagram for index_table:



IMAGE

Data Fields

index_entry table [**TABLE_SIZE**]
int **id**

Field Documentation

int id

index_entry table[TABLE_SIZE]

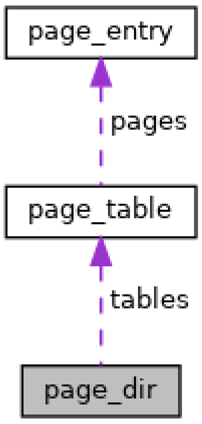
The documentation for this struct was generated from the following file:

include/mem/**heap.h**

page_dir Struct Reference

`#include <paging.h>`

Collaboration diagram for page_dir:



IMAGE

Data Fields

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

Field Documentation

page_table* tables[1024]

u32int tables_phys[1024]

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

include/mem/**paging.h**

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

Field Documentation

u32int accessed

u32int dirty

u32int frameaddr

u32int present

u32int reserved

u32int usermode

u32int writeable

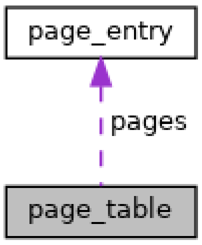
The documentation for this struct was generated from the following file:

include/mem/**paging.h**

page_table Struct Reference

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

Collaboration diagram for page_table:

IMAGE 

Data Fields

page_entry pages [1024]

Field Documentation

page_entry pages[1024]

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

include/mem/**paging.h**

param Struct Reference

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

Data Fields

int **op_code**
int **device_id**
char * **buffer_ptr**
int * **count_ptr**

Field Documentation

char* buffer_ptr

int* count_ptr

int device_id

int op_code

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

modules/**mpx_supt.h**

pcb Struct Reference

`#include <pcb.h>`

Collaboration diagram for pcb:

IMAGE

Data Fields

char **name** [20]
int **class**
int **priority**
int **state**
int **suspended**
unsigned char **stack** [**STACK_SIZE**]
unsigned char * **topStack**
unsigned char * **baseStack**
struct **pcb** * **next**
struct **pcb** * **previous**

Field Documentation

unsigned char* baseStack

int class

char name[20]

struct pcb* next

struct pcb* previous

int priority

unsigned char stack[STACK_SIZE]

int state

int suspended

unsigned char* topStack

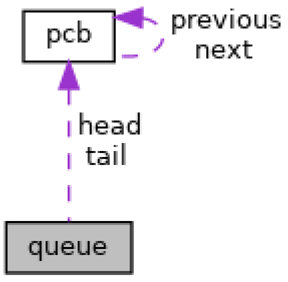
The documentation for this struct was generated from the following file:

modules/**pcb.h**

queue Struct Reference

`#include <queue.h>`

Collaboration diagram for queue:



IMAGE

Data Fields

int **size**
pcb * **head**
pcb * **tail**

Field Documentation

pcb* head

int size

pcb* tail

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

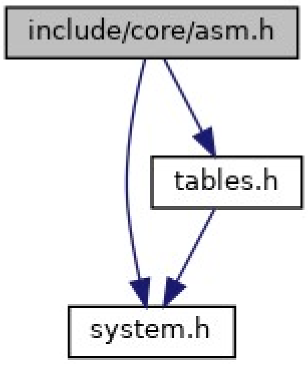
modules/**queue.h**

File Documentation

include/core/asm.h File Reference

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

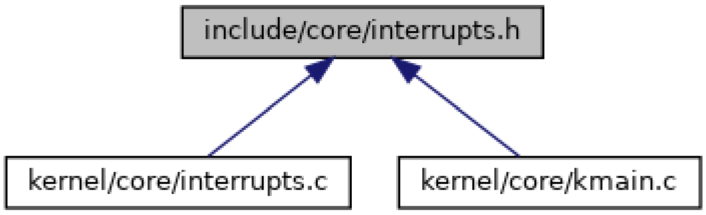
Include dependency graph for asm.h:



IMAGE

include/core/interrupts.h File Reference

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

IMAGE 

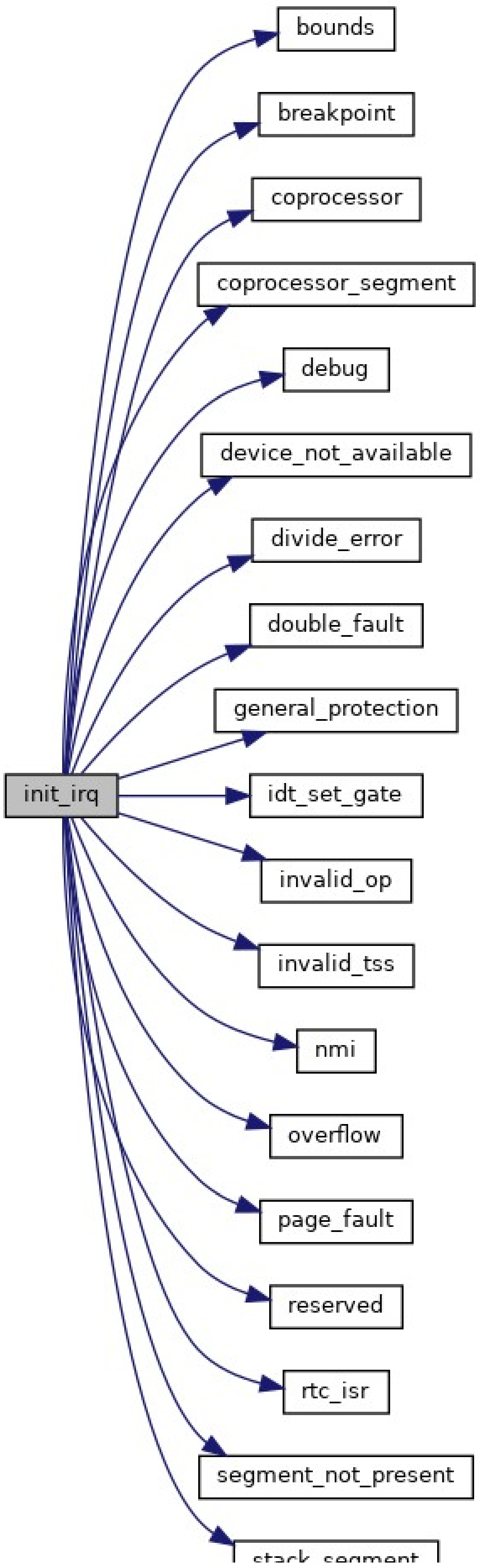
Functions

void **init_irq** (void)
void **init_pic** (void)

Function Documentation

void init_irq (void)

Here is the call graph for this function:

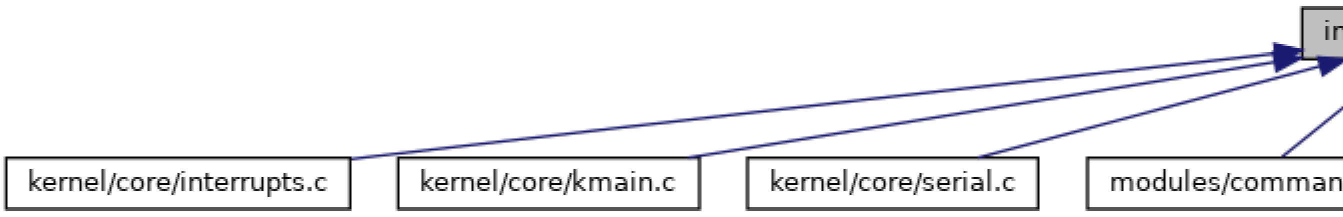

bounds
breakpoint
coprocessor
coprocessor_segment
debug
device_not_available
divide_error
double_fault
general_protection
init_irq
idt_set_gate
invalid_op
invalid_tss
nmi
overflow
page_fault
reserved
rtc_isr
segment_not_present
stack_segment

void init_pic (void)

include/core/io.h File Reference

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

IMAGE



Macros

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

Macro Definition Documentation

#define inb(port)

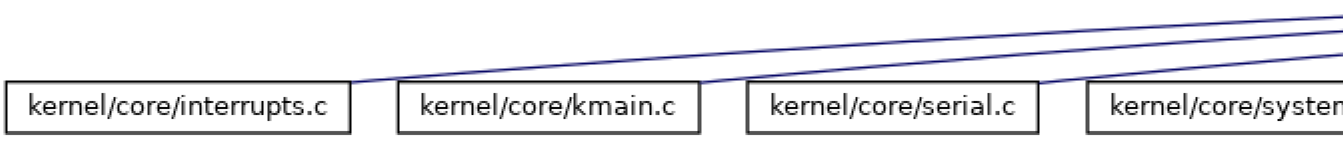
Value:
```
        ({                                 \
     unsigned char r;                       \
     asm volatile ("inb %%dx,%%al": "=a" (r): "d" (port)); \
     r;                                 \
   })
```

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

include/core/serial.h File Reference

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

IMAGE



Macros

#define **COM1** 0x3f8
#define **COM2** 0x2f8
#define **COM3** 0x3e8
#define **COM4** 0x2e8
#define **DEFAULT** "\x1b[0m"
#define **RED** "\x1b[31m"
#define **GREEN** "\x1b[32m"
#define **YELLOW** "\x1b[33m"

Functions

int **init_serial** (int device)

Initializes serial device.

int **serial_println** (const char *msg)
int **serial_print** (const char *msg)
int **set_serial_out** (int device)
int **set_serial_in** (int device)
int * **polling** (char *buffer, int *count)
void **println_error** (char *msg)
void **println_warning** (char *msg)
void **println_confirmation** (char *msg)
void **print_confirmation** (char *msg)
void **println_message** (char *msg)
void **simple_print** (char *msg)

Macro Definition Documentation

#define COM1 0x3f8

#define COM2 0x2f8

#define COM3 0x3e8

#define COM4 0x2e8

#define DEFAULT "\x1b[0m"

#define GREEN "\x1b[32m"

#define RED "\x1b[31m"

#define YELLOW "\x1b[33m"

Function Documentation

int init_serial (int *device*)

Initializes serial device.

Parameters

int	device

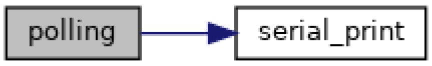
int* polling (char * *buffer*, int * *count*)

Repeatedly checks status register to see if a bit has been entered, stores and prints, or does another action to the input.

Parameters

	char *buffer, int *count

Here is the call graph for this function:

IMAGE


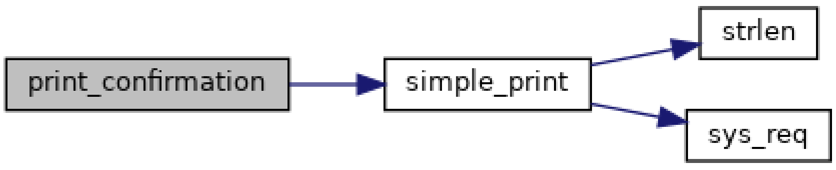
void print_confirmation (char * *msg*)

Prints the message in confirmation color green

Parameters

char	*msg

Here is the call graph for this function:



IMAGE

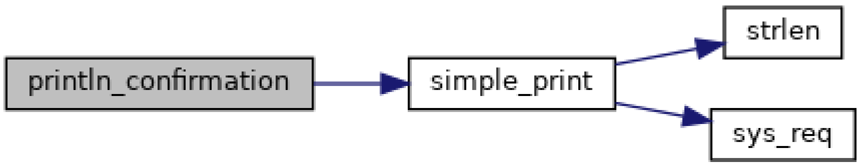
void println_confirmation (char * *msg*)

Prints the message in confirmation color green with newline

Parameters

char	*msg

Here is the call graph for this function:



IMAGE

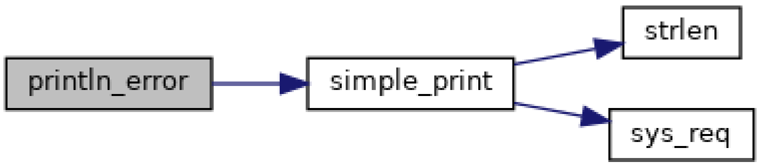
void println_error (char * *msg*)

Prints the message in error color red

Parameters

char	*msg

Here is the call graph for this function:



IMAGE

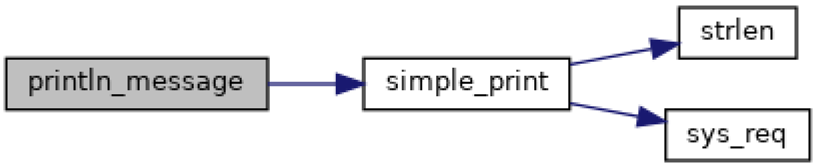
void println_message (char * *msg*)

Prints the message in default color and newline

Parameters

char	*msg

Here is the call graph for this function:



IMAGE

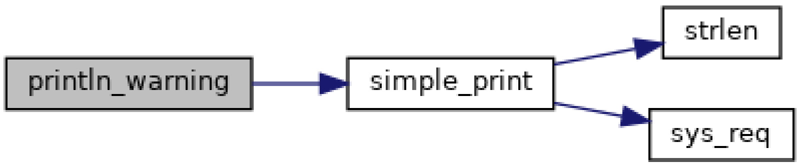
void println_warning (char * *msg*)

Prints the message in warning color yellow

Parameters

char	*msg

Here is the call graph for this function:



IMAGE

int serial_print (const char * *msg*)

Writes a message to the active serial output device.

Parameters

const	char *msg

int serial_println (const char * *msg*)

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

Parameters

const	char *msg

int set_serial_in (int *device*)

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

Parameters

int	device

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

int	device

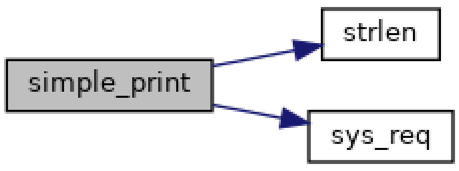
void simple_print (char * *msg*)

Prints the message out to the screen

Parameters

char	*msg

Here is the call graph for this function:

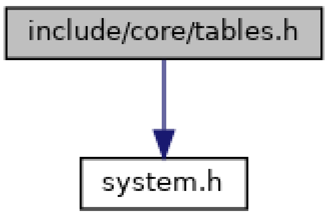


IMAGE

include/core/tables.h File Reference

`#include "system.h"`

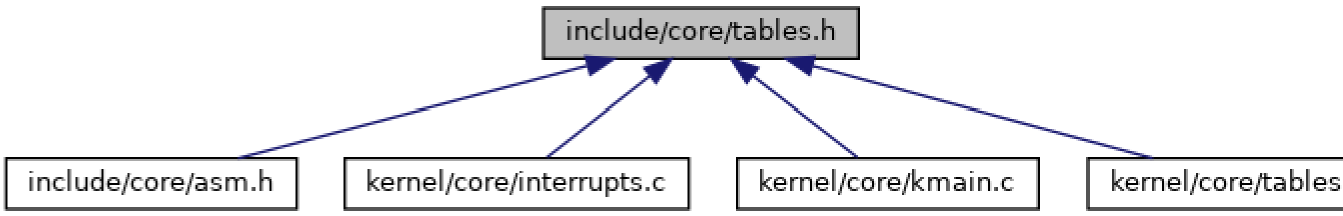
Include dependency graph for tables.h:



IMAGE

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

IMAGE



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

Function Documentation

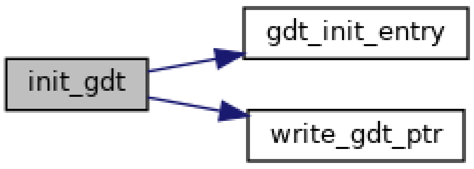
struct idt_entry_struct __attribute__ ((packed))

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

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

void init_gdt ()

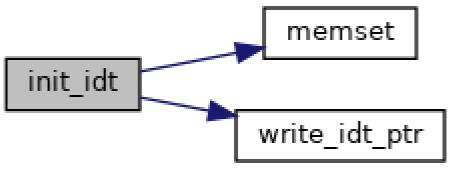
Here is the call graph for this function:



IMAGE

void init_idt ()

Here is the call graph for this function:



IMAGE

Variable Documentation

u8int access

u32int base

u8int base_high

u16int base_low

u8int base_mid

u8int flags

u16int limit

u16int limit_low

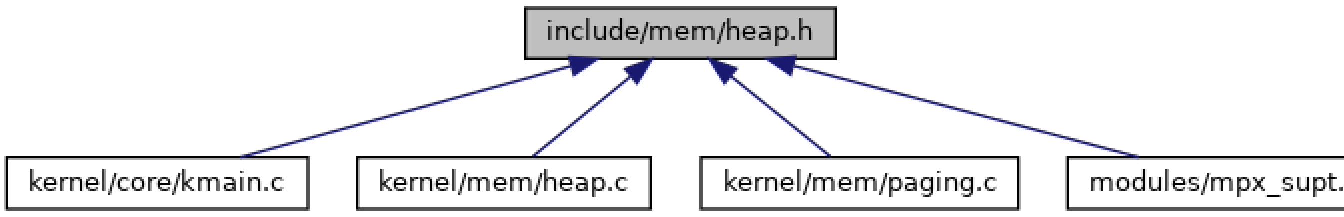
u16int sselect

u8int zero

include/mem/heap.h File Reference

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

IMAGE



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__**

Macro Definition Documentation

#define KHEAP_BASE 0xD000000

#define KHEAP_MIN 0x10000

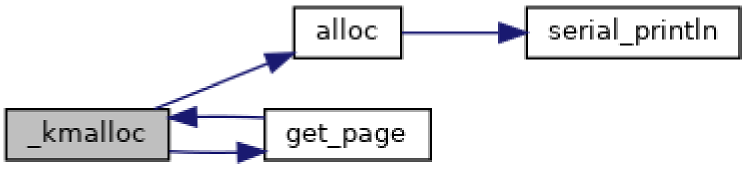
#define KHEAP_SIZE 0x1000000

#define TABLE_SIZE 0x1000

Function Documentation

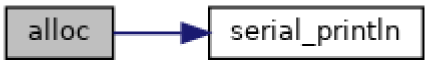
u32int _kmalloc (u32int *size*, int *align*, u32int * *phys_addr*)

Here is the call graph for this function:

IMAGE 

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

Here is the call graph for this function:

IMAGE 

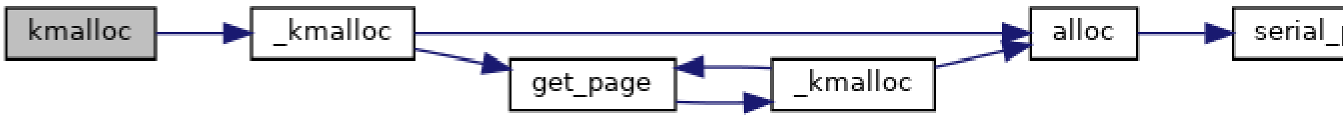
void init_kheap ()

u32int kfree ()

u32int kmalloc (u32int ***size*****)**

Here is the call graph for this function:

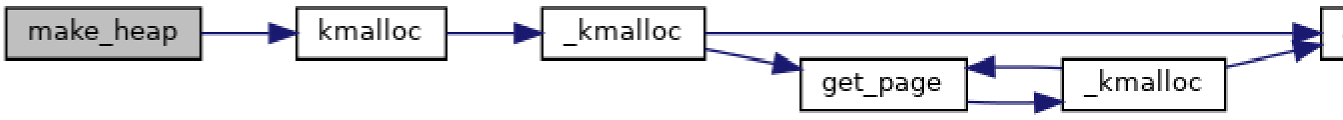
IMAGE



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

Here is the call graph for this function:

IMAGE



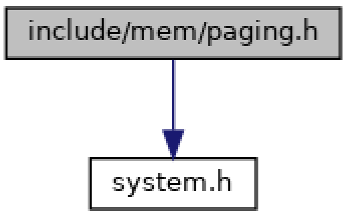
Variable Documentation

struct gdt_entry_struct __attribute__

include/mem/paging.h File Reference

`#include <system.h>`

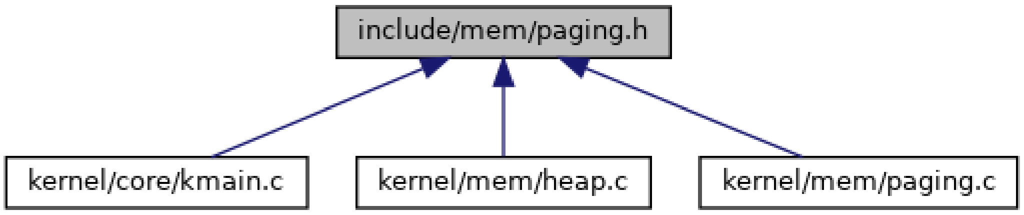
Include dependency graph for paging.h:



IMAGE

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

IMAGE



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)

Macro Definition Documentation

#define PAGE_SIZE 0x1000

Function Documentation

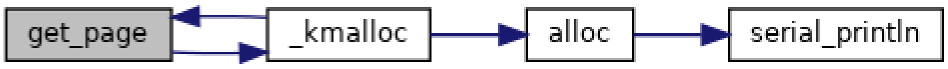
void clear_bit (u32int ***addr*****)**

u32int first_free ()

u32int get_bit (u32int ***addr*****)**

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

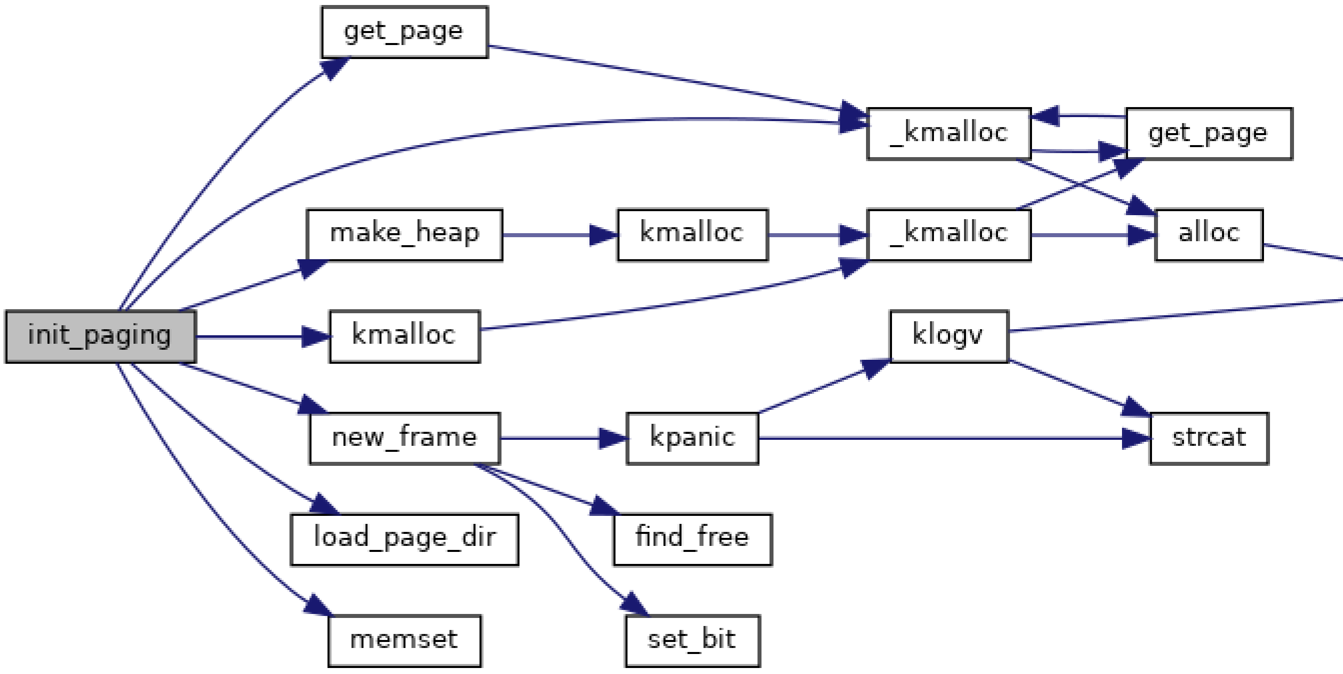
Here is the call graph for this function:

IMAGE 

void init_paging ()

Here is the call graph for this function:

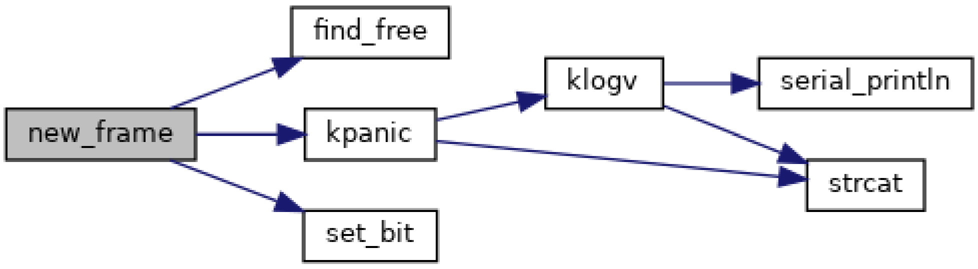
IMAGE



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

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

Here is the call graph for this function:



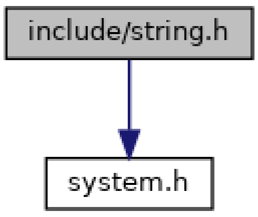
IMAGE

void set_bit (u32int ***addr*****)**

include/string.h File Reference

`#include <system.h>`

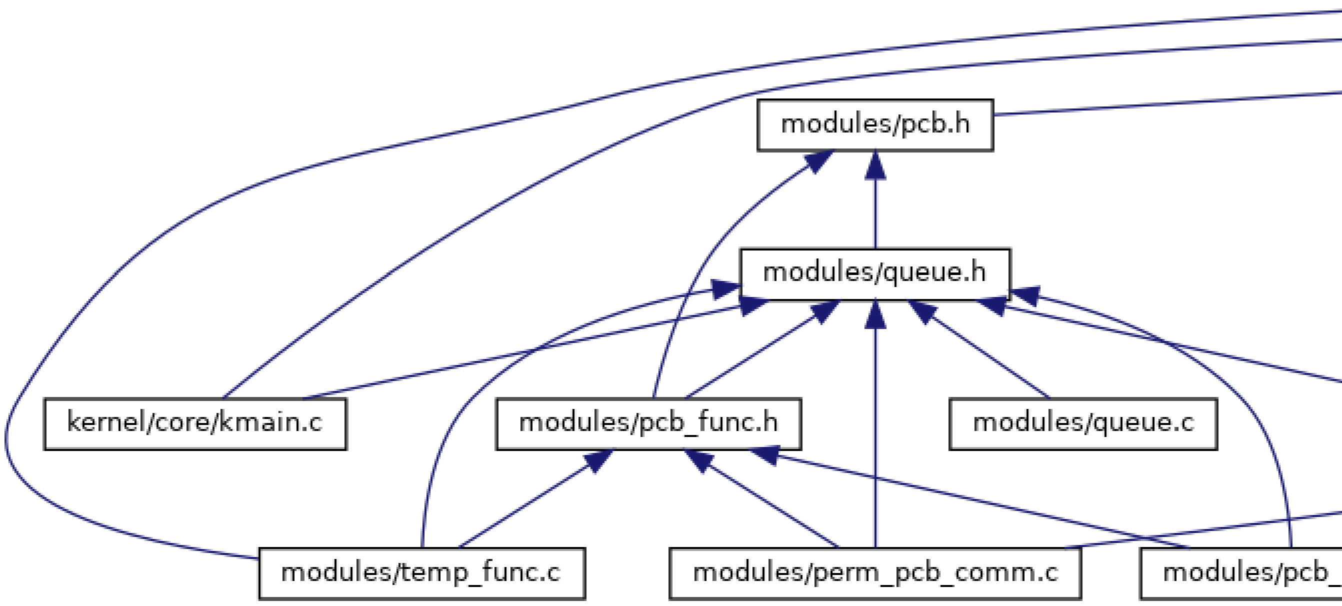
Include dependency graph for string.h:



IMAGE

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

IMAGE



Functions

int **isspace** (const char *c)
void * **memset** (void *s, int c, **size_t** n)
char * **strcpy** (char *s1, const char *s2)
char * **strcat** (char *s1, const char *s2)
int **strlen** (const char *s)
int **strcmp** (const char *s1, const char *s2)
int **strncmp** (const char *s1, const char *s2, **size_t** n)
char * **strtok** (char *s1, const char *s2)
int **atoi** (const char *s)
char * **itoa** (int n, char *str, int **base**)
char * **reverse** (char str[], int **i**, int j)
void **swap** (char *x, char *y)

Function Documentation

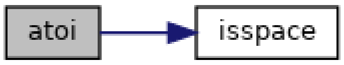
int atoi (const char * *s*)

Convert an ASCII string to an integer

Parameters

const	char *s

Here is the call graph for this function:

IMAGE 

int isspace (const char * *c*)

Determine if a character is whitespace.

Parameters

const	char *c-character to check

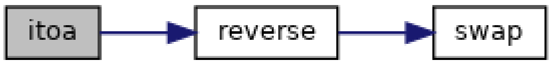
char* itoa (int *num*, char * *buffer*, int *base*)

Convert an integer to ASCII string

Parameters

int	num, char *buffer, int base

Here is the call graph for this function:

IMAGE 

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

Set a region of memory.

Parameters

void	*s-destination, int c-byte to write, size_t n-count

char* reverse (char *str*[], int *i*, int *j*)

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

Concatenate the contents of one string onto another.

Parameters

char	*s1-destination, const char *s2-source

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

String comparison

Parameters

const	char *s1-string, const char *s2-string

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

Copy one string to another.

Parameters

char	*s1-destination, char *s2-source

int strlen (const char * *s*)

Returns the length of a string.

Parameters

const	char *s

int strncmp (const char * *s1*, const char * *s2*, size_t *n*)

String comparison for a given number of characters

Parameters

const	char *s1-string 1, const char *s2-string 2, n-size_t

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

Split string into tokens

Parameters

char	*s1-string, s2-delimiter

void swap (char * *x*, char * *y*)

swaps two char values

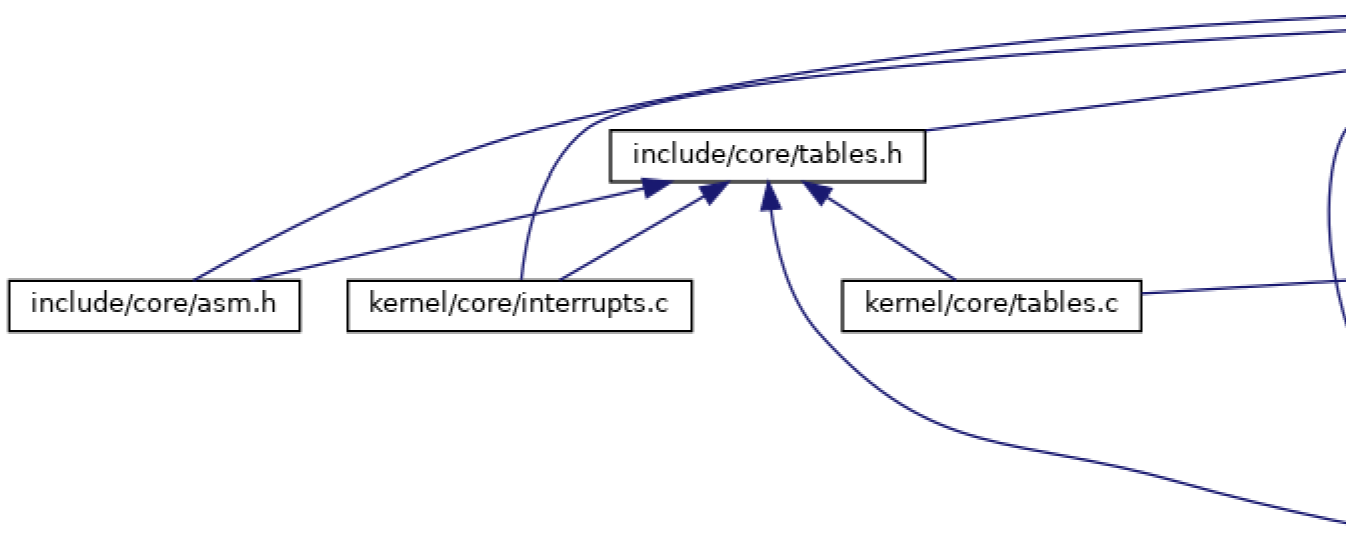
Parameters

char	*x, char *y

include/system.h File Reference

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

IMAGE



Data Structures

struct **date_time**

Macros

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

Typedefs

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

Functions

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

Macro Definition Documentation

#define asm __asm__

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

#define GDT_CS_ID 0x01

#define GDT_DS_ID 0x02

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

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

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

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

#define NULL 0

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

#define volatile __volatile__

Typedef Documentation

typedef unsigned int size_t

typedef unsigned short u16int

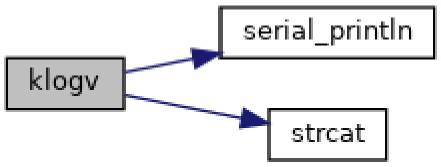
typedef unsigned long u32int

typedef unsigned char u8int

Function Documentation

void klogv (const char * *msg*)

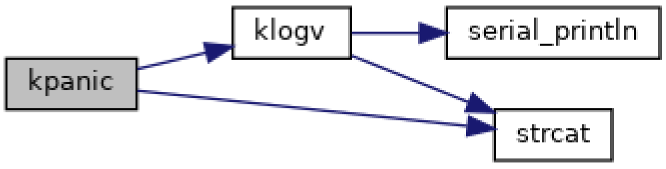
Here is the call graph for this function:



IMAGE

void kpanic (const char * *msg*)

Here is the call graph for this function:



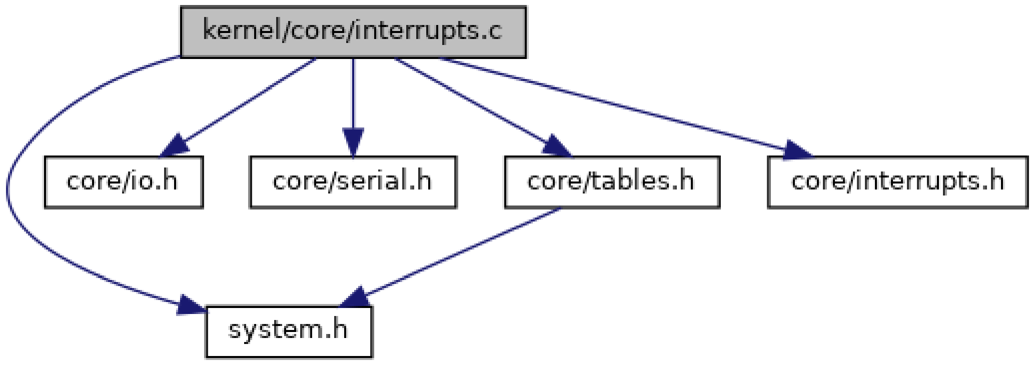
IMAGE

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:

IMAGE



Macros

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

Functions

void **divide_error** ()
void **debug** ()
void **nmi** ()
void **breakpoint** ()
void **overflow** ()
void **bounds** ()
void **invalid_op** ()
void **device_not_available** ()
void **double_fault** ()
void **coprocessor_segment** ()
void **invalid_tss** ()
void **segment_not_present** ()
void **stack_segment** ()
void **general_protection** ()
void **page_fault** ()
void **reserved** ()
void **coprocessor** ()
void **rtc_isr** ()
void **isr0** ()
void **do_isr** ()
void **init_irq** (void)
void **init_pic** (void)
void **do_divide_error** ()
void **do_debug** ()
void **do_nmi** ()
void **do_breakpoint** ()
void **do_overflow** ()

void **do_bounds** ()
void **do_invalid_op** ()
void **do_device_not_available** ()
void **do_double_fault** ()
void **do_coprocessor_segment** ()
void **do_invalid_tss** ()
void **do_segment_not_present** ()
void **do_stack_segment** ()
void **do_general_protection** ()
void **do_page_fault** ()
void **do_reserved** ()
void **do_coprocessor** ()

Variables

idt_entry **idt_entries** [256]

Macro Definition Documentation

#define ICW1 0x11

#define ICW4 0x01

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

#define PIC1 0x20

#define PIC2 0xA0

Function Documentation

void bounds ()

void breakpoint ()

void coprocessor ()

void coprocessor_segment ()

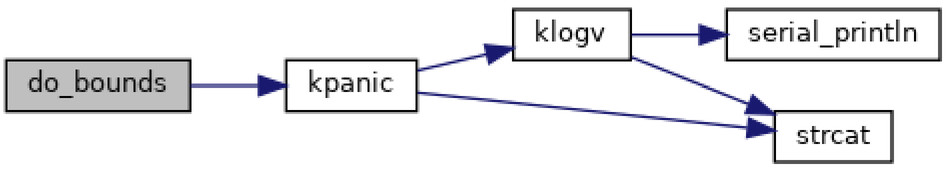
void debug ()

void device_not_available ()

void divide_error ()

void do_bounds ()

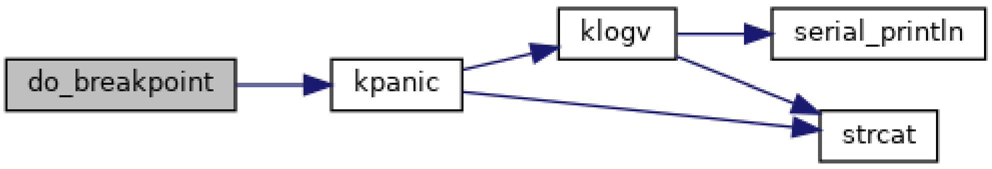
Here is the call graph for this function:



IMAGE

void do_breakpoint ()

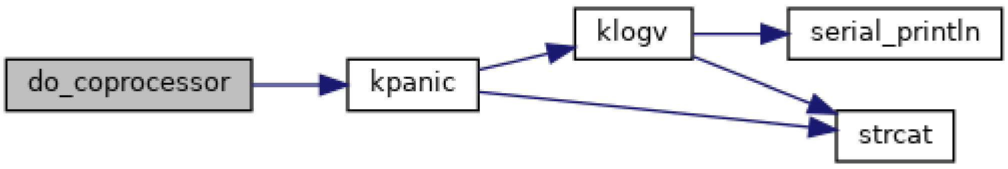
Here is the call graph for this function:



IMAGE

void do_coprocessor ()

Here is the call graph for this function:

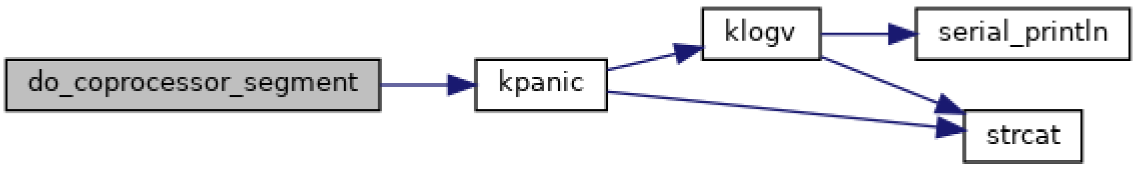


IMAGE

void do_coprocessor_segment ()

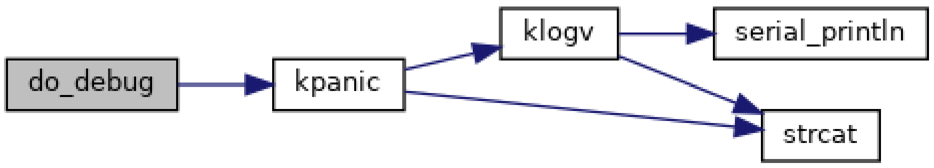
Here is the call graph for this function:

IMAGE



void do_debug ()

Here is the call graph for this function:

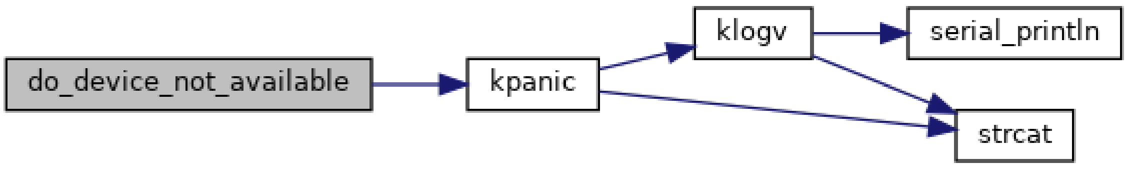


IMAGE

void do_device_not_available ()

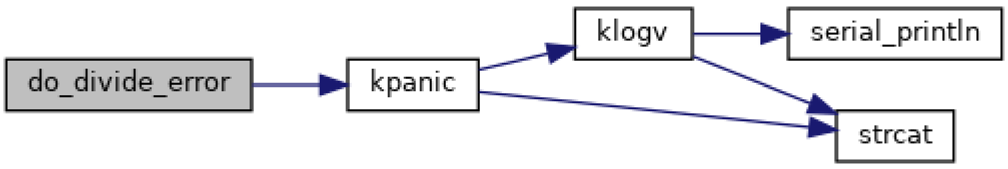
Here is the call graph for this function:

IMAGE



void do_divide_error ()

Here is the call graph for this function:

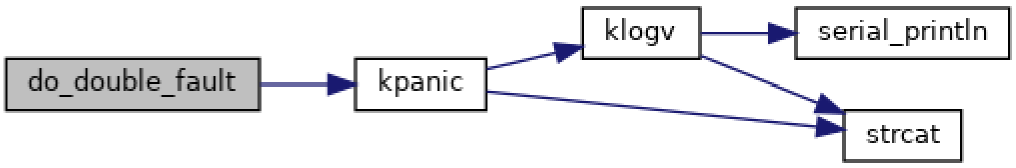


IMAGE

void do_double_fault ()

Here is the call graph for this function:

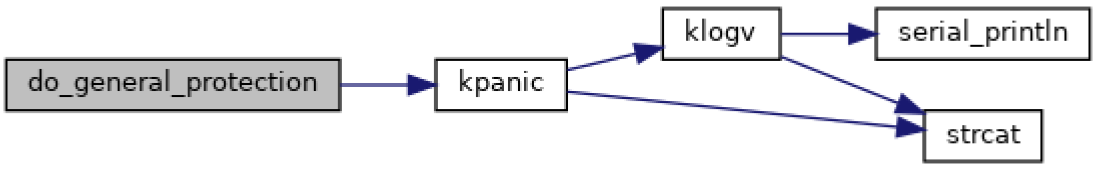
IMAGE



void do_general_protection ()

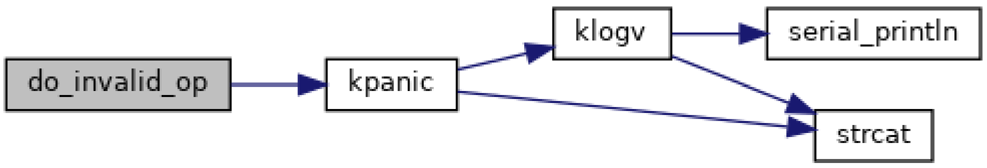
Here is the call graph for this function:

IMAGE



void do_invalid_op ()

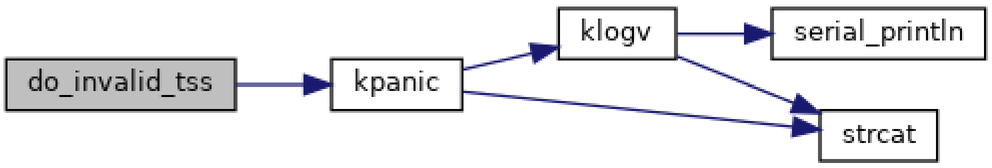
Here is the call graph for this function:



IMAGE

void do_invalid_tss ()

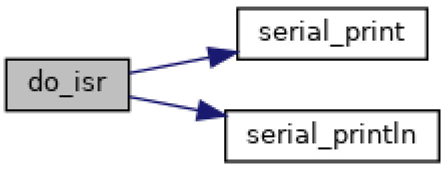
Here is the call graph for this function:



IMAGE

void do_isr ()

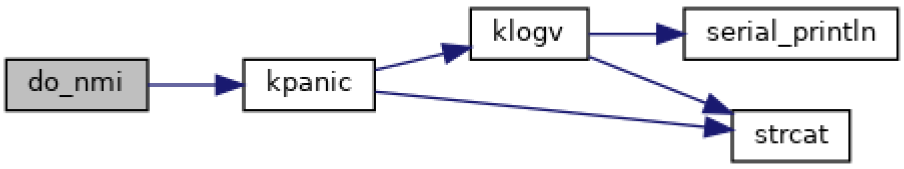
Here is the call graph for this function:



IMAGE

void do_nmi ()

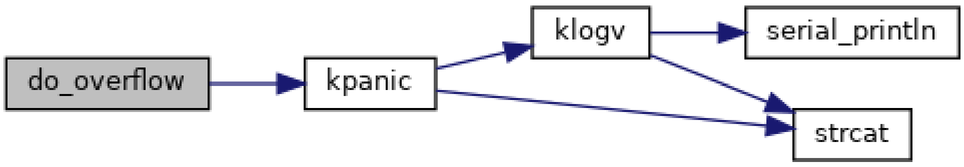
Here is the call graph for this function:



IMAGE

void do_overflow ()

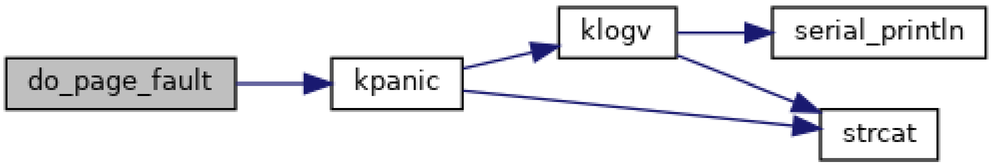
Here is the call graph for this function:



IMAGE

void do_page_fault ()

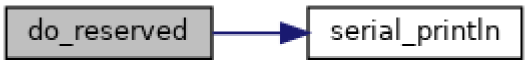
Here is the call graph for this function:



IMAGE

void do_reserved ()

Here is the call graph for this function:

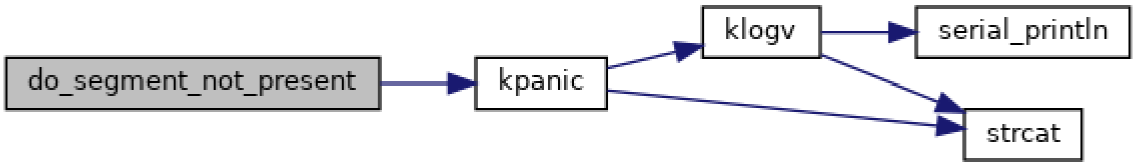


IMAGE

void do_segment_not_present ()

Here is the call graph for this function:

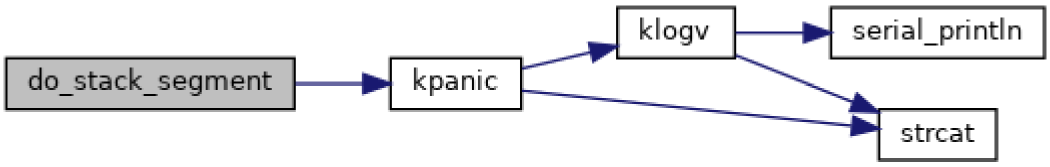
IMAGE



void do_stack_segment ()

Here is the call graph for this function:

IMAGE

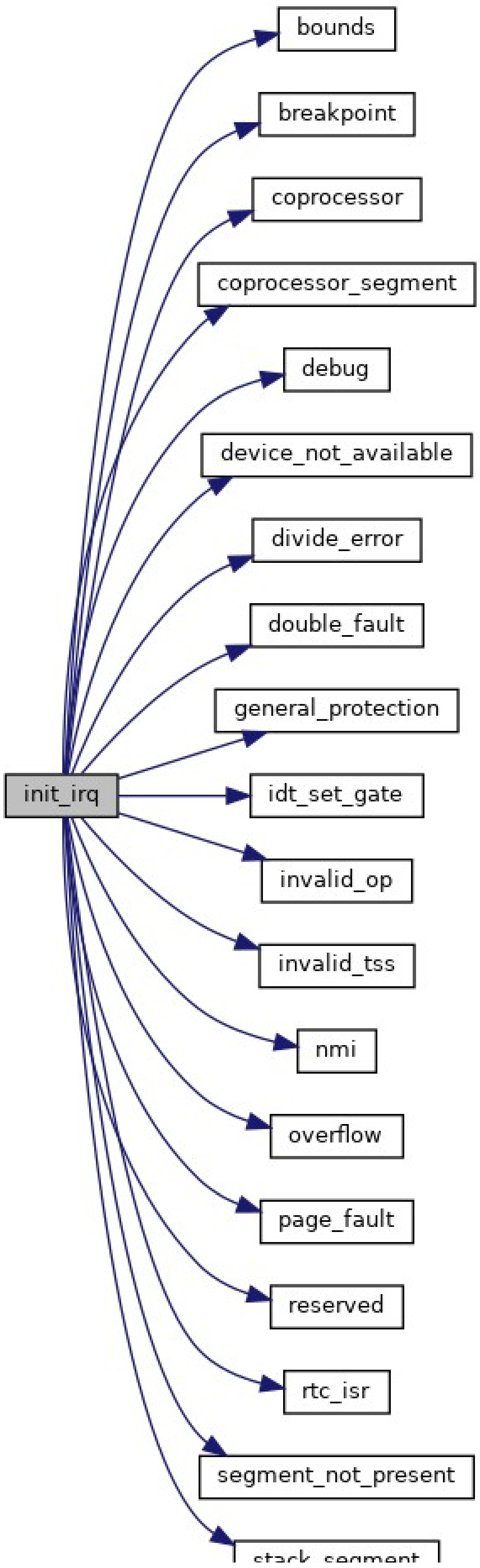


void double_fault ()

void general_protection ()

void init_irq (void)

Here is the call graph for this function:


bounds
breakpoint
coprocessor
coprocessor_segment
debug
device_not_available
divide_error
double_fault
general_protection
init_irq
idt_set_gate
invalid_op
invalid_tss
nmi
overflow
page_fault
reserved
rtc_isr
segment_not_present

void init_pic (void)

void invalid_op ()

void invalid_tss ()

void isr0 ()

void nmi ()

void overflow ()

void page_fault ()

void reserved ()

void rtc_isr ()

void segment_not_present ()

void stack_segment ()

Variable Documentation

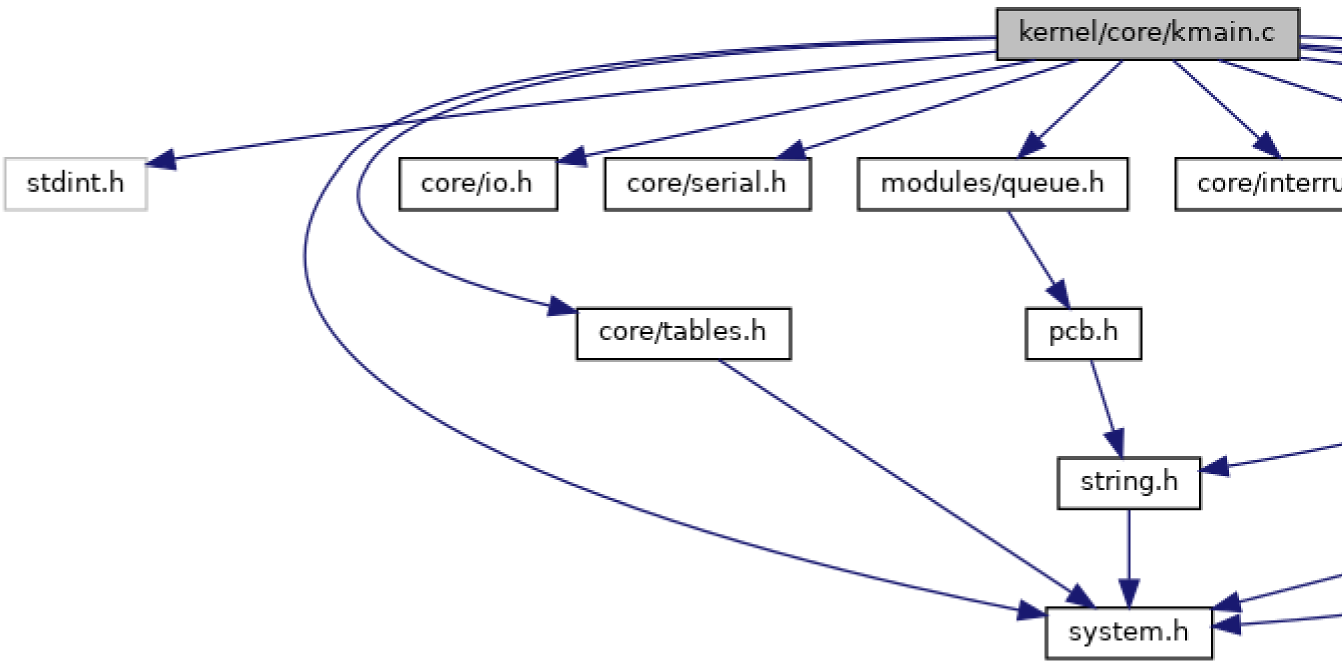
idt_entry idt_entries[256]

kernel/core/kmain.c File Reference

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

Include dependency graph for kmain.c:

IMAGE



Functions

void **kmain** (void)

Detailed Description

Kernel main. The first function called after the bootloader. Initialization of hardware, system structures, devices, and initial processes happens here.

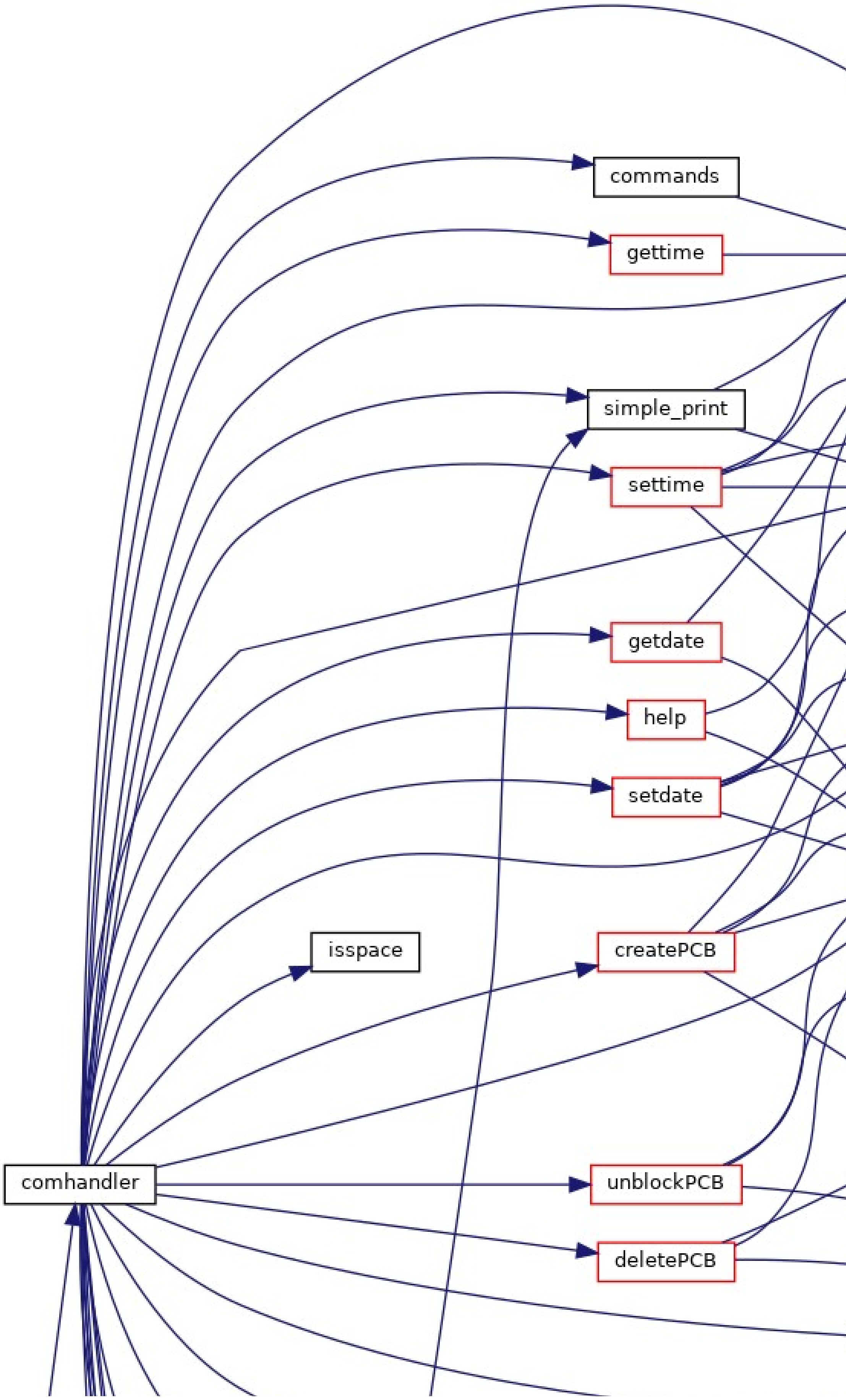
Initial Kernel – by Forrest Desjardin, 2013, Modifications by: Andrew Duncan 2014, John Jacko 2017 Ben Smith 2018, and Alex Wilson 2019

Function Documentation

void kmain (void)

Here is the call graph for this function:

IMAGE

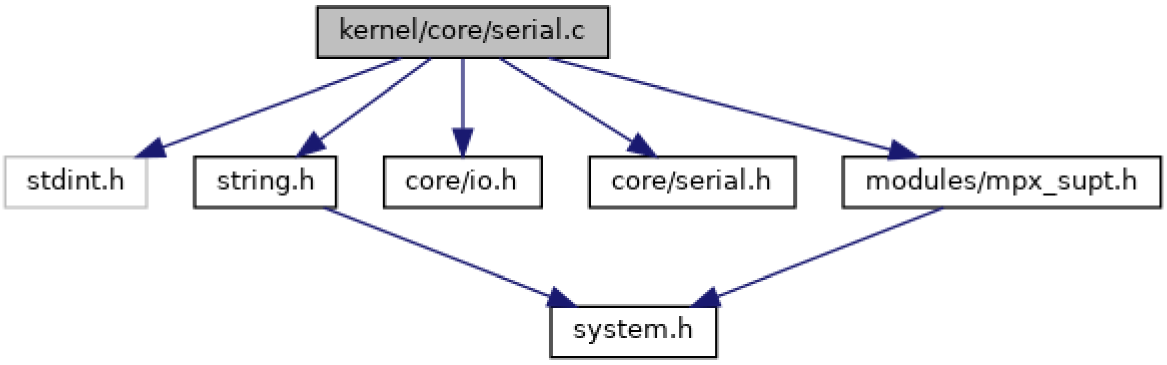

commands
gettime
simple_print
settime
getdate
help
setdate
isspace
createPCB
comhandler
unblockPCB
deletePCB

kernel/core/serial.c File Reference

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

Include dependency graph for serial.c:

IMAGE



Macros

#define **NO_ERROR** 0

Functions

int **init_serial** (int device)

Initializes serial device.

int **serial_println** (const char *msg)
int **serial_print** (const char *msg)
int **set_serial_out** (int device)
int **set_serial_in** (int device)
int * **polling** (char *buffer, int *count)
void **println_error** (char *msg)
void **println_warning** (char *msg)
void **println_confirmation** (char *msg)
void **print_confirmation** (char *msg)
void **println_message** (char *msg)
void **simple_print** (char *msg)

Variables

int **serial_port_out** = 0

Active devices used for serial output.

int **serial_port_in** = 0

Active devices used for serial output.

int **i** = 0

counter for polling

int **cursor** =0

Keepts track of the cursor position in the terminal.

Detailed Description

Contains methods and variables used for serial input and output.

Macro Definition Documentation

#define NO_ERROR 0

Function Documentation

int init_serial (int *device*)

Initializes serial device.

Parameters

int	device

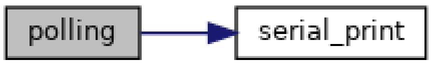
int* polling (char * *buffer*, int * *count*)

Repeatedly checks status register to see if a bit has been entered, stores and prints, or does another action to the input.

Parameters

	char *buffer, int *count

Here is the call graph for this function:

IMAGE 

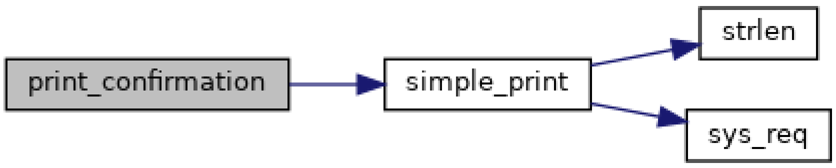
void print_confirmation (char * *msg*)

Prints the message in confirmation color green

Parameters

char	*msg

Here is the call graph for this function:



IMAGE

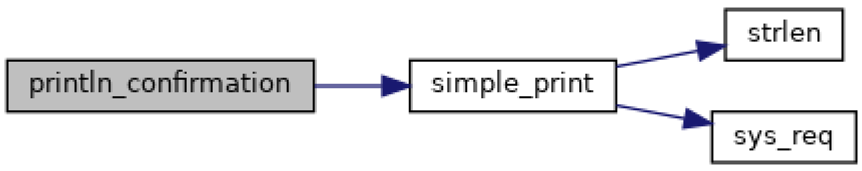
void println_confirmation (char * *msg*)

Prints the message in confirmation color green with newline

Parameters

char	*msg

Here is the call graph for this function:



IMAGE

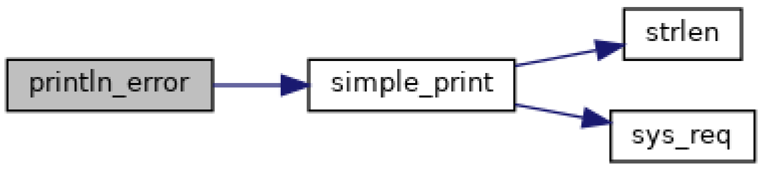
void println_error (char *   *msg*)

Prints the message in error color red

Parameters

char	*msg

Here is the call graph for this function:



IMAGE

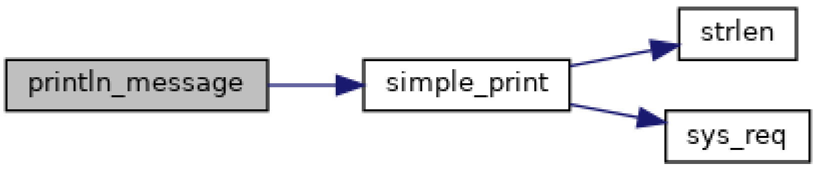
void println_message (char *   *msg*)

Prints the message in default color and newline

Parameters

char	*msg

Here is the call graph for this function:



IMAGE

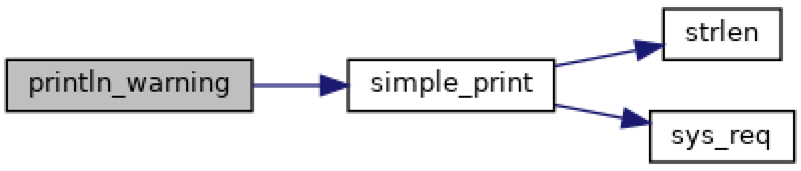
void println_warning (char *   *msg*)

Prints the message in warning color yellow

Parameters

char	*msg

Here is the call graph for this function:



IMAGE

int serial_print (const char *   *msg*)

Writes a message to the active serial output device.

Parameters

const	char *msg

int serial_println (const char *   *msg*)

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

Parameters

const	char *msg

int set_serial_in (int *device*)

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

Parameters

int	device

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

int	device

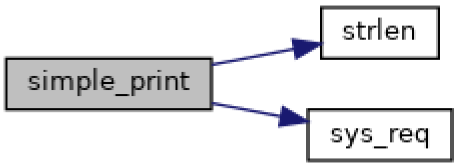
void simple_print (char * *msg*)

Prints the message out to the screen

Parameters

char	*msg

Here is the call graph for this function:



IMAGE

Variable Documentation

int cursor =0

Keepts track of the cursor position in the terminal.

int i = 0

counter for polling

int serial_port_in = 0

Active devices used for serial output.

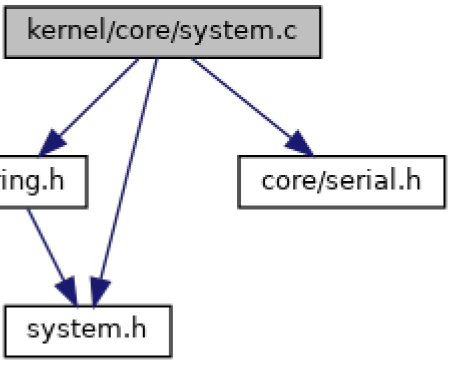
int serial_port_out = 0

Active devices used for serial output.

kernel/core/system.c File Reference

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

Include dependency graph for system.c:



IMAGE

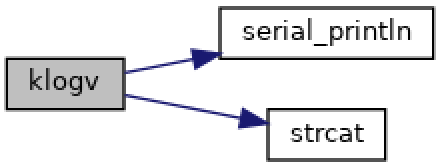
Functions

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

Function Documentation

void klogv (const char * *msg*)

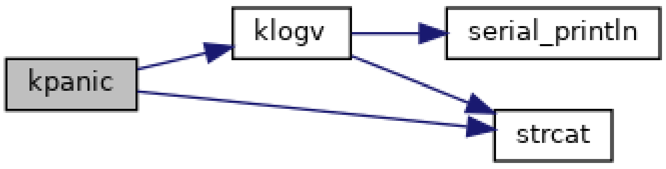
Here is the call graph for this function:



IMAGE

void kpanic (const char * *msg*)

Here is the call graph for this function:

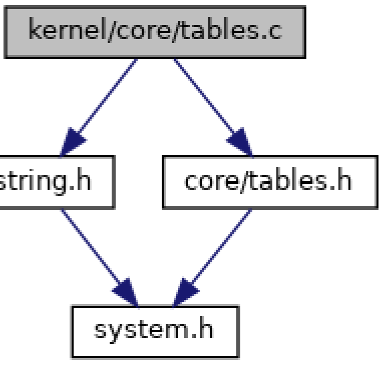


IMAGE

kernel/core/tables.c File Reference

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

Include dependency graph for tables.c:



IMAGE

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]

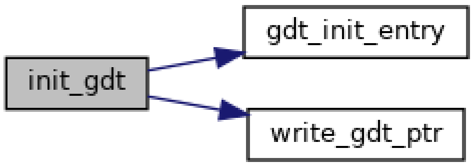
Function Documentation

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

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

void init_gdt ()

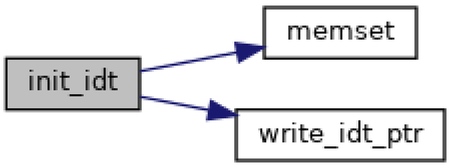
Here is the call graph for this function:



IMAGE

void init_idt ()

Here is the call graph for this function:



IMAGE

void write_gdt_ptr (u32int , size_t)

void write_idt_ptr (u32int)

Variable Documentation

gdt_entry gdt_entries[5]

gdt_descriptor gdt_ptr

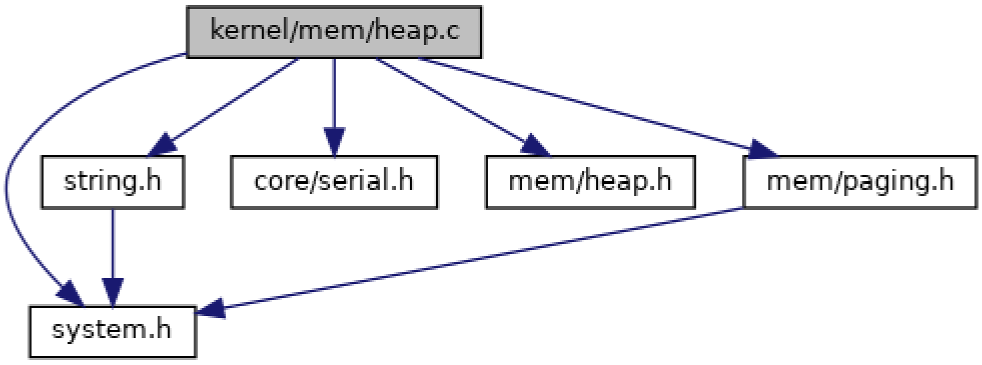
idt_entry idt_entries[256]

idt_descriptor idt_ptr

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:



IMAGE

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

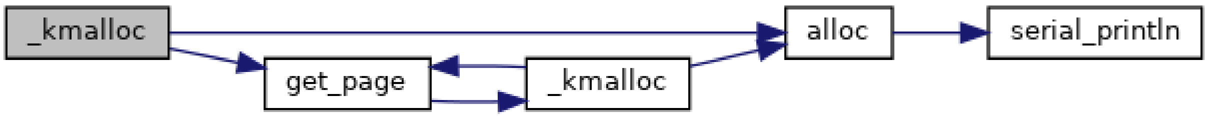
heap * **kheap** = 0
heap * **curr_heap** = 0
page_dir * **kdir**
void * **end**
void **_end**
void **__end**
u32int phys_alloc_addr = (**u32int**)&**end**

Function Documentation

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

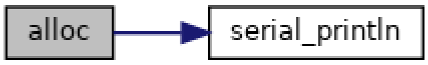
Here is the call graph for this function:

IMAGE



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

Here is the call graph for this function:

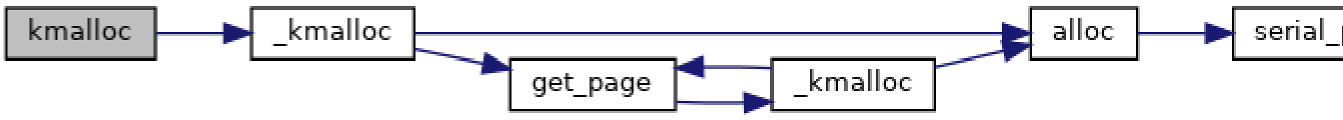


IMAGE

u32int kmalloc (u32int *size*)

Here is the call graph for this function:

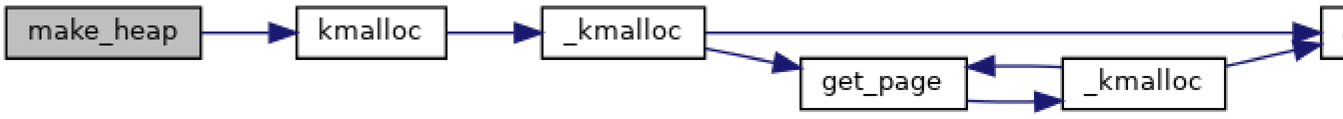
IMAGE



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

Here is the call graph for this function:

IMAGE



Variable Documentation

void __end

void _end

heap* curr_heap = 0

void* end

page_dir* kdir

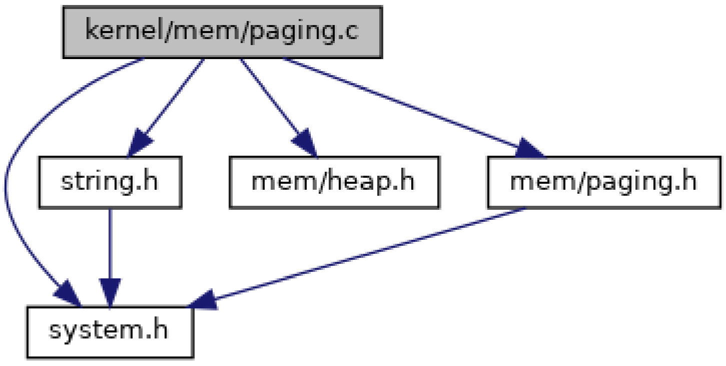
heap* kheap = 0

u32int phys_alloc_addr = (u32int)&end

kernel/mem/paging.c File Reference

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

Include dependency graph for paging.c:



IMAGE

Functions

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

Variables

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

Function Documentation

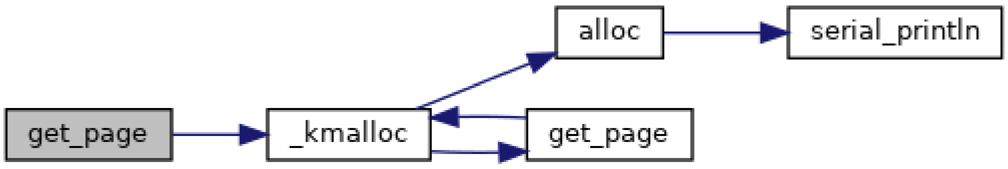
void clear_bit (u32int *addr*)

u32int find_free ()

u32int get_bit (u32int *addr*)

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

Here is the call graph for this function:

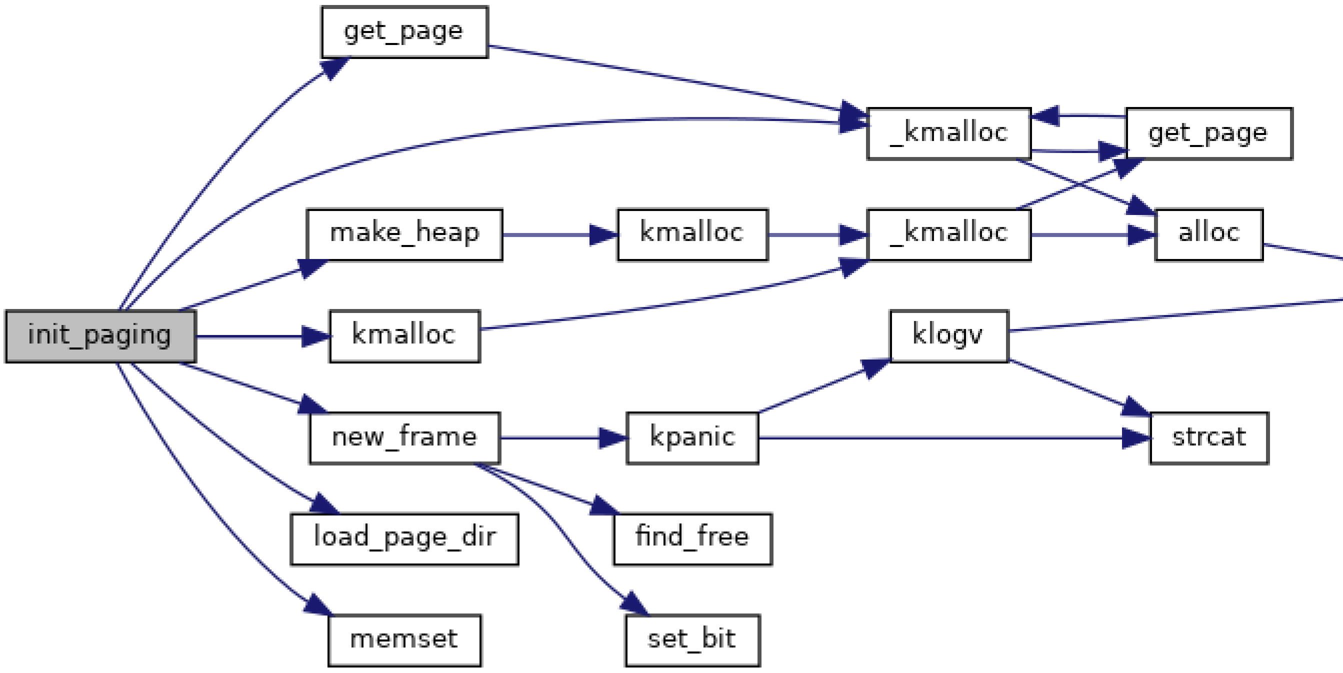


IMAGE

void init_paging ()

Here is the call graph for this function:

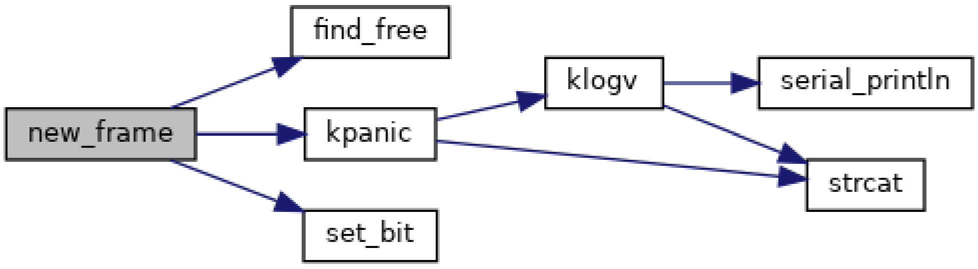
IMAGE



void load_page_dir (page_dir *   *new_dir*)

void new_frame (page_entry *   *page*)

Here is the call graph for this function:



IMAGE

void set_bit (u32int   *addr*)

Variable Documentation

page_dir* cdir = 0

u32int* frames

page_dir* kdir = 0

heap* kheap

u32int mem_size = 0x4000000

u32int nframes

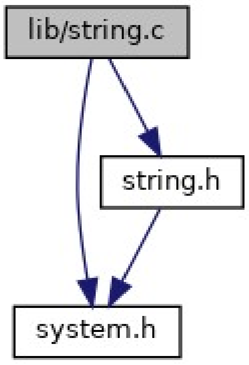
u32int page_size = 0x1000

u32int phys_alloc_addr

lib/string.c File Reference

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

Include dependency graph for string.c:



IMAGE

Functions

int **strlen** (const char *s)
char * **strcpy** (char *s1, const char *s2)
int **atoi** (const char *s)
char * **itoa** (int num, char *buffer, int **base**)
char * **reverse** (char *str, int **i**, int j)
void **swap** (char *x, char *y)
int **strcmp** (const char *s1, const char *s2)
int **strncmp** (const char *s1, const char *s2, **size_t** n)
char * **strcat** (char *s1, const char *s2)
int **isspace** (const char *c)
void * **memset** (void *s, int c, **size_t** n)
char * **strtok** (char *s1, const char *s2)

Detailed Description

Implementation of C string functions

Function Documentation

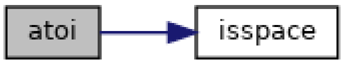
int atoi (const char * *s*)

Convert an ASCII string to an integer

Parameters

const	char *s

Here is the call graph for this function:



IMAGE

int isspace (const char * *c*)

Determine if a character is whitespace.

Parameters

const	char *c-character to check

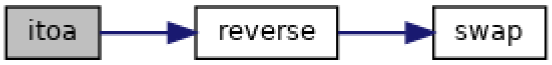
char* itoa (int *num*, char * *buffer*, int *base*)

Convert an integer to ASCII string

Parameters

int	num, char *buffer, int base

Here is the call graph for this function:

IMAGE 

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

Set a region of memory.

Parameters

void	*s-destination, int c-byte to write, size_t n-count

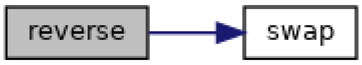
char* reverse (char * *str*, int *i*, int *j*)

reverses contents of string

Parameters

char	*str, int i, int j

Here is the call graph for this function:

IMAGE 

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

Concatenate the contents of one string onto another.

Parameters

char	*s1-destination, const char *s2-source

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

String comparison

Parameters

const	char *s1-string, const char *s2-string

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

Copy one string to another.

Parameters

char	*s1-destination, char *s2-source

int strlen (const char * *s*)

Returns the length of a string.

Parameters

const	char *s

int strncmp (const char * *s1*, const char * *s2*, size_t *n*)

String comparison for a given number of characters

Parameters

const	char *s1-string 1, const char *s2-string 2, n-size_t

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

Split string into tokens

Parameters

char	*s1-string, s2-delimiter

void swap (char * *x*, char * *y*)

swaps two char values

Parameters

char	*x, char *y

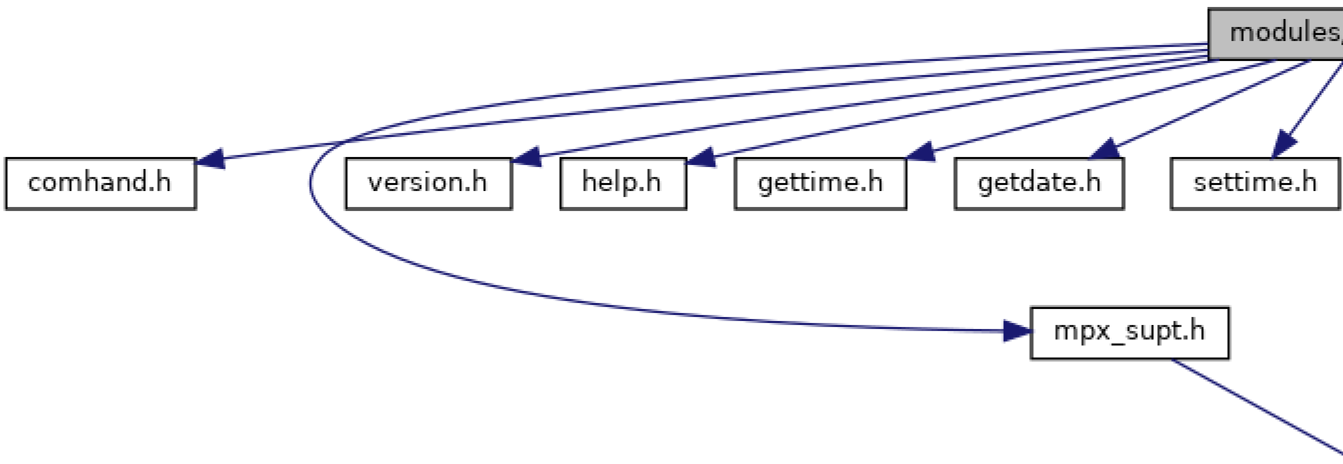
mainpage.txt File Reference

modules/comhand.c File Reference

```
#include "comhand.h"
#include "mpx_supt.h"
#include "version.h"
#include "help.h"
#include "gettime.h"
#include "getdate.h"
#include "settime.h"
#include "setdate.h"
#include "commands.h"
#include "temp_func.h"
#include "queue.h"
#include "perm_pcb_comm.h"
#include <core/serial.h>
#include <string.h>
```

Include dependency graph for comhand.c:

IMAGE



Functions

int **comhandler** ()

Detailed Description

handles the input commands from the command line

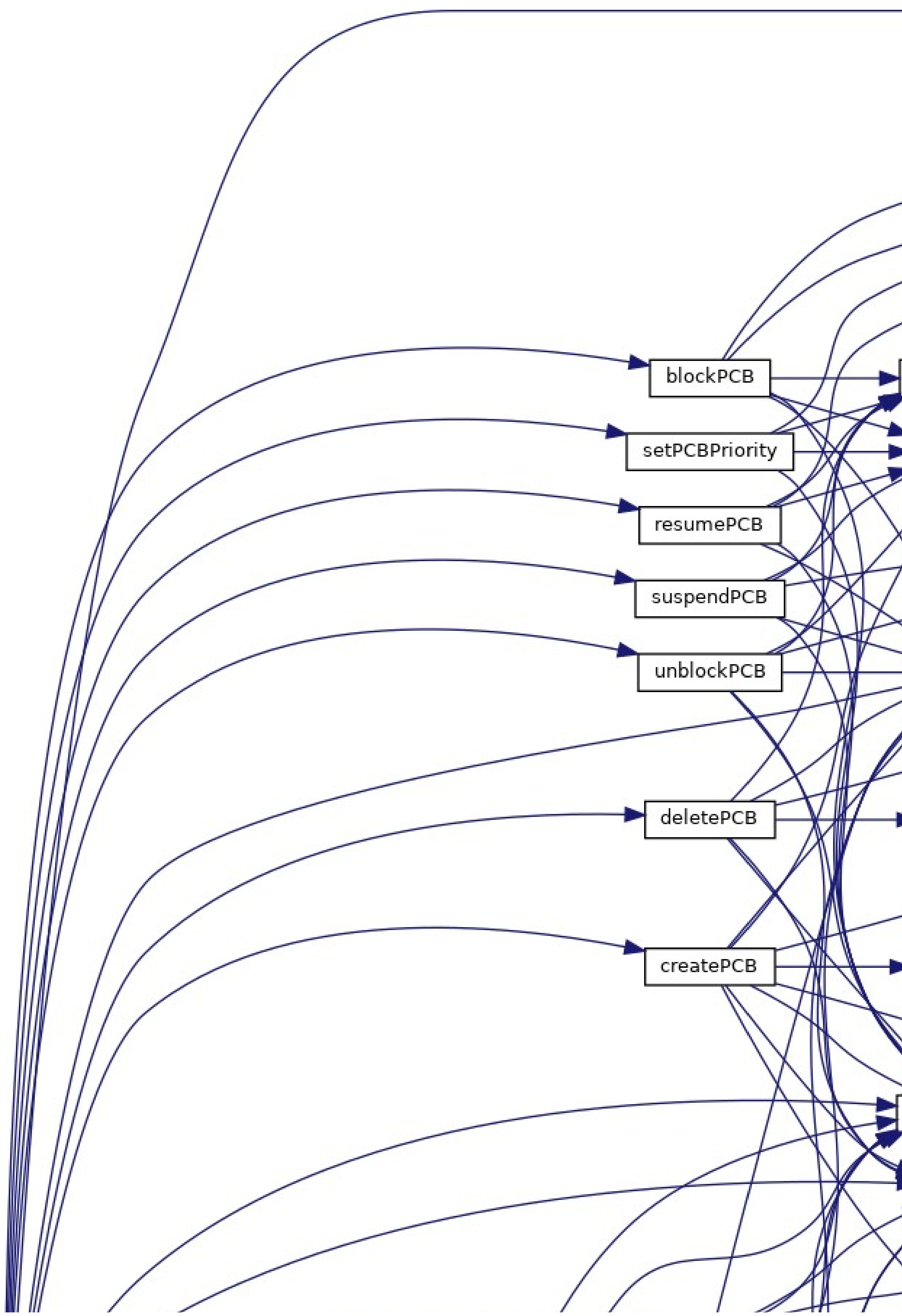
Function Documentation

int comhandler ()

Calls the polling function in **serial.c** and interprets the commands given to it

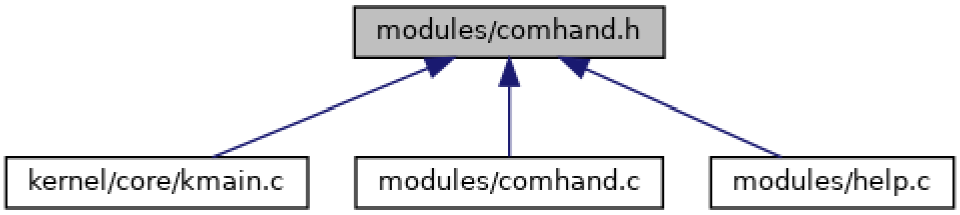
Here is the call graph for this function:

IMAGE


blockPCB
setPCBPriority
resumePCB
suspendPCB
unblockPCB
deletePCB
createPCB

modules/comhand.h File Reference

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

IMAGE 

Macros

#define **BUFFER** 100

Functions

int **comhandler** ()

Detailed Description

comhand header file

Macro Definition Documentation

#define BUFFER 100

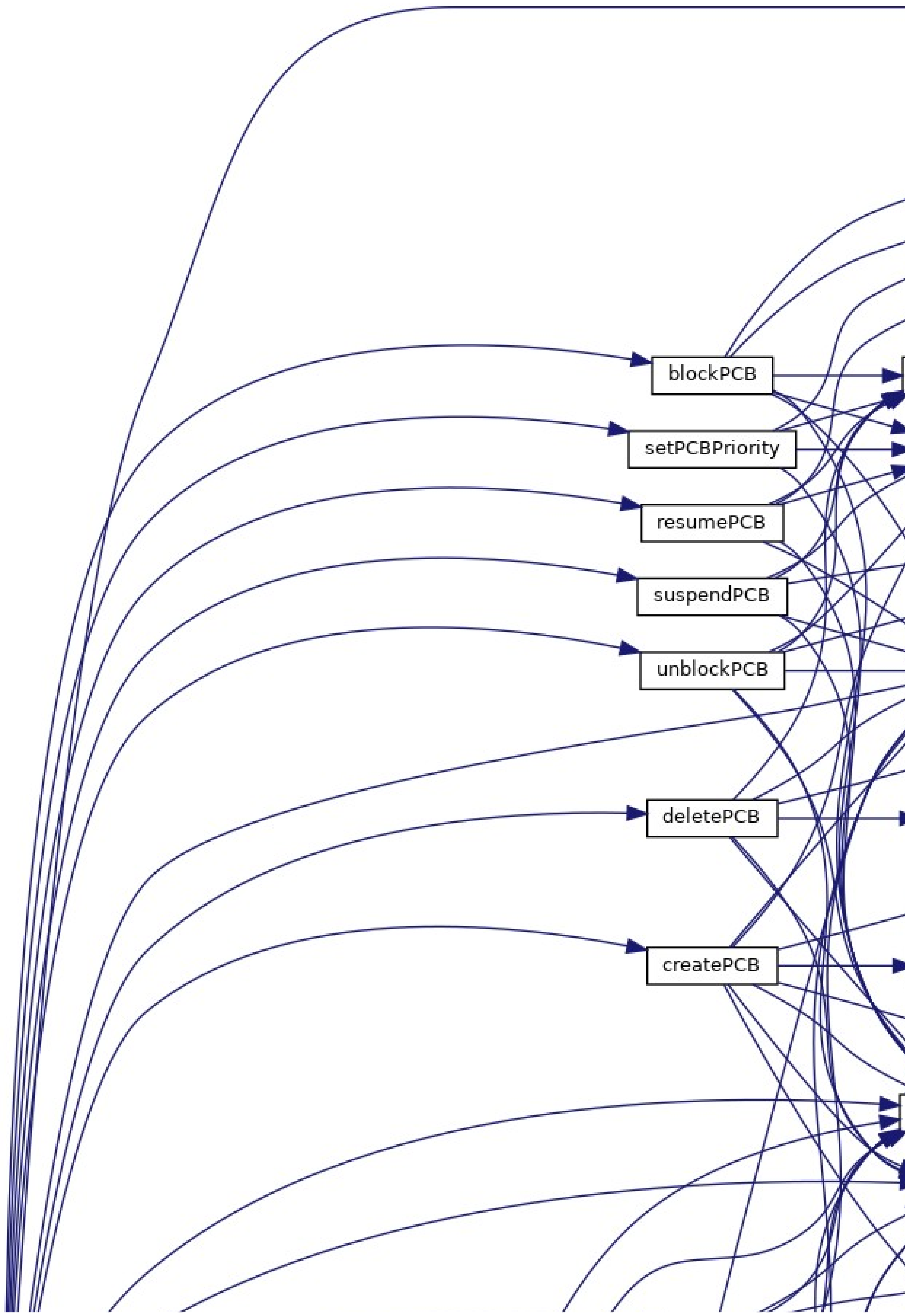
Function Documentation

int comhandler ()

Calls the polling function in **serial.c** and interprets the commands given to it

Here is the call graph for this function:

IMAGE

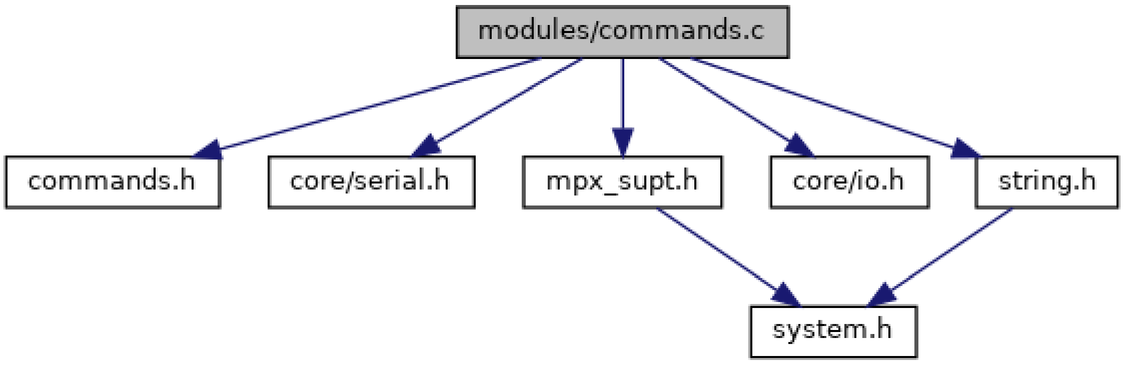

blockPCB
setPCBPriority
resumePCB
suspendPCB
unblockPCB
deletePCB
createPCB

modules/commands.c File Reference

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

Include dependency graph for commands.c:

IMAGE



Functions

void **commands** ()

Detailed Description

Contains function **commands()** to display the available user commands

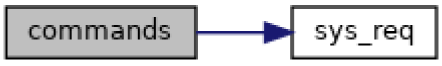
Function Documentation

void commands ()

Outputs the current available user commands

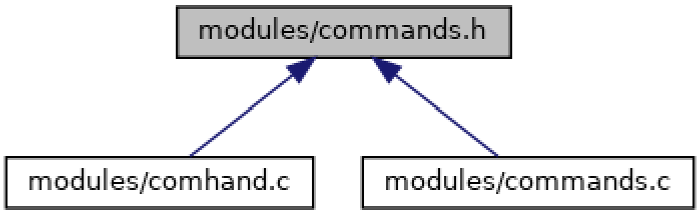
Here is the call graph for this function:

IMAGE



modules/commands.h File Reference

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

IMAGE 

Functions

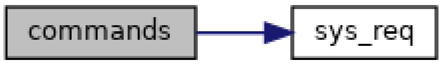
void **commands** ()

Function Documentation

void commands ()

Outputs the current available user commands

Here is the call graph for this function:

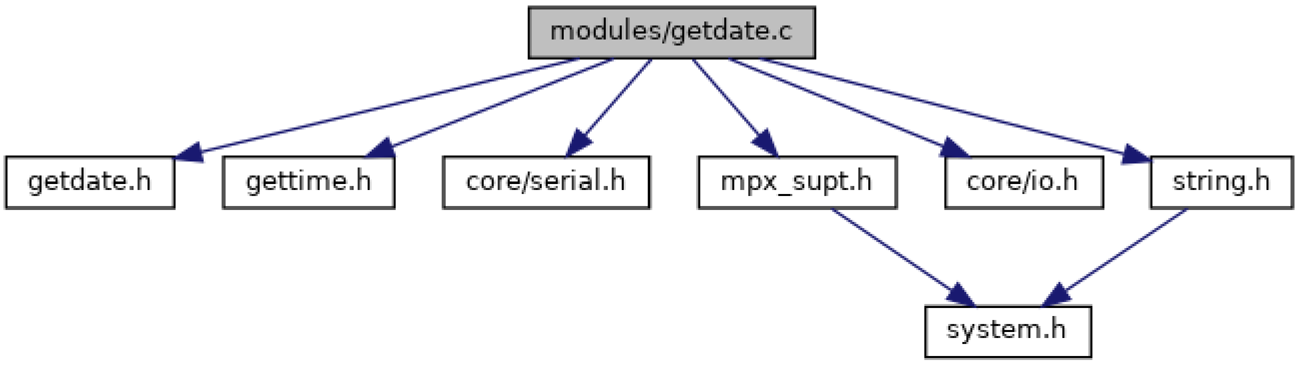
IMAGE 

modules/getdate.c File Reference

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

Include dependency graph for getdate.c:

IMAGE



Functions

void **getdate** ()

Detailed Description

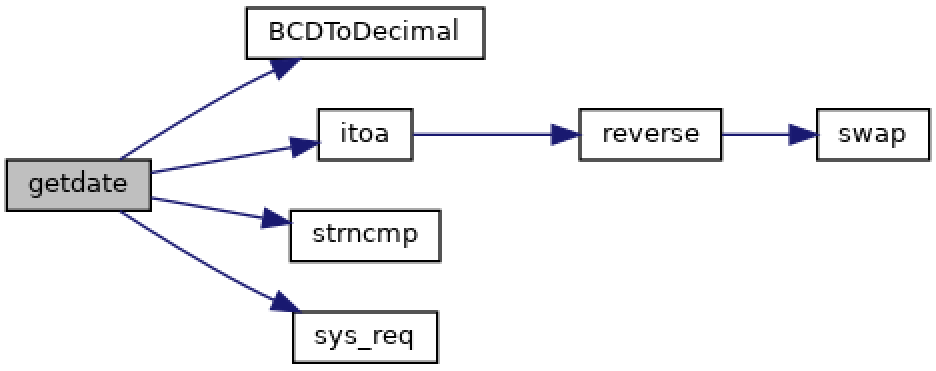
Contains function **getdate()** to display the current date

Function Documentation

void getdate ()

Displays the current date on the machine

Here is the call graph for this function:

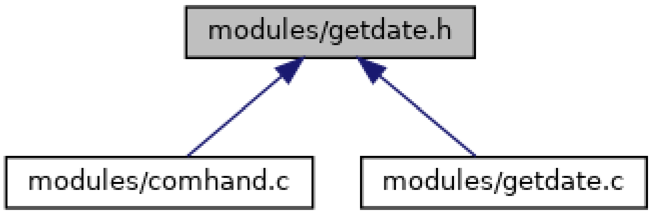


IMAGE

modules/getdate.h File Reference

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

IMAGE



Functions

void **getdate** ()

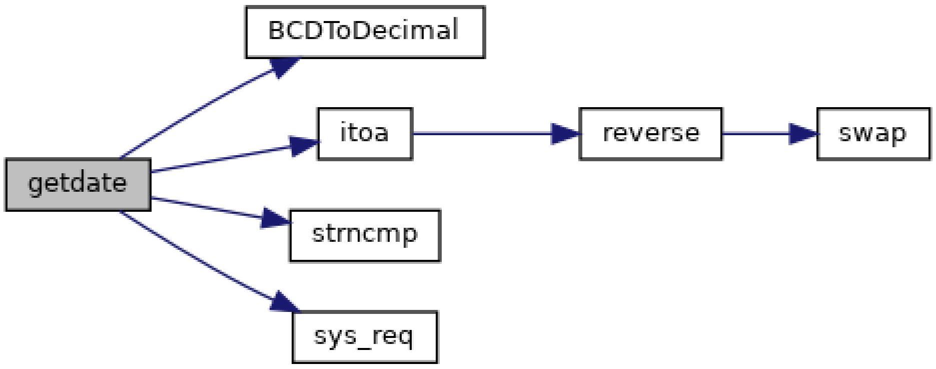
Function Documentation

void getdate ()

Displays the current date on the machine

Here is the call graph for this function:

IMAGE

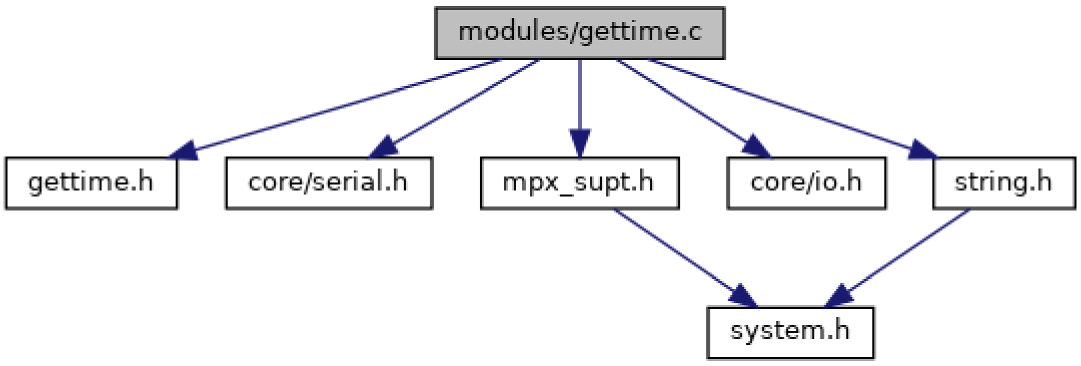


modules/gettime.c File Reference

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

Include dependency graph for gettime.c:

IMAGE



Functions

void **gettime** ()
int **BCDToDecimal** (int BCD)
int **DecimalToBCD** (int decimal)

Detailed Description

Contains function **gettime()** to display the current time

Function Documentation

int BCDToDecimal (int *BCD*)

Converts BCD (Binary Coded Decimal) to Decimal

Parameters

int	BCD

int DecimalToBCD (int *decimal*)

Converts Decimal to BCD (Binary Coded Deciaml)

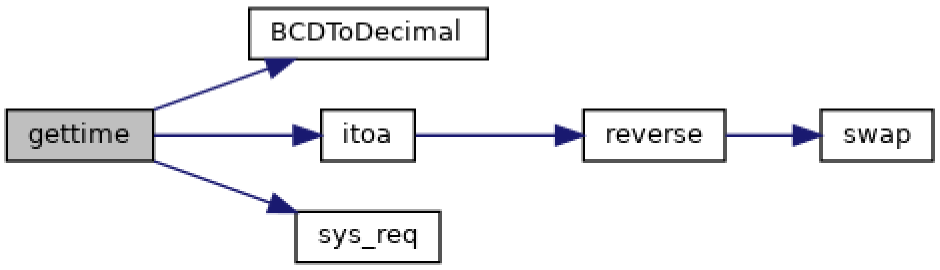
Parameters

int	decimal

void gettime ()

Gets the current time running on the system

Here is the call graph for this function:

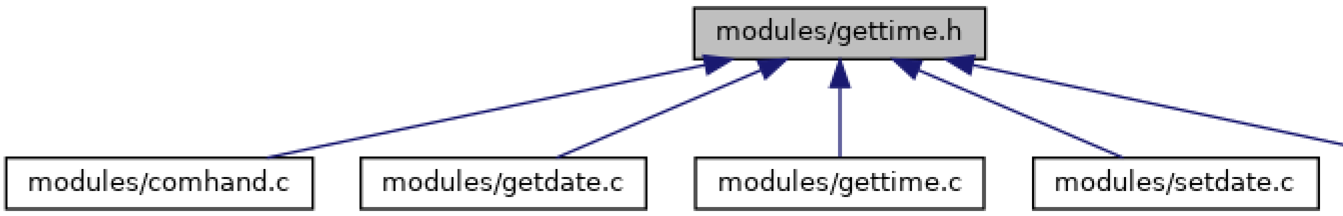


IMAGE

modules/gettime.h File Reference

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

IMAGE



Functions

void **gettime** ()
int **BCDToDecimal** (int BCD)
int **DecimalToBCD** (int decimal)

Function Documentation

int BCDToDecimal (int *BCD*)

Converts BCD (Binary Coded Decimal) to Decimal

Parameters

int	BCD

int DecimalToBCD (int *decimal*)

Converts Decimal to BCD (Binary Coded Deciaml)

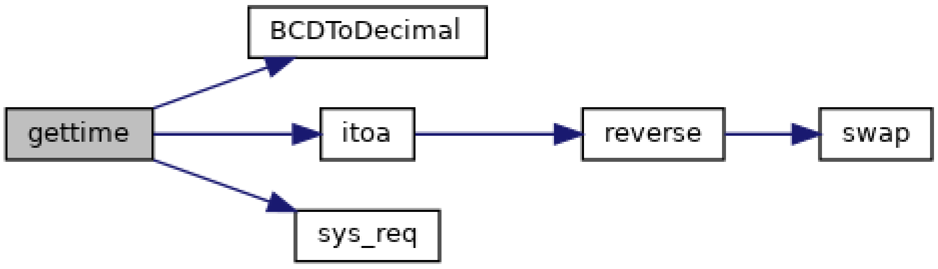
Parameters

int	decimal

void gettime ()

Gets the current time running on the system

Here is the call graph for this function:



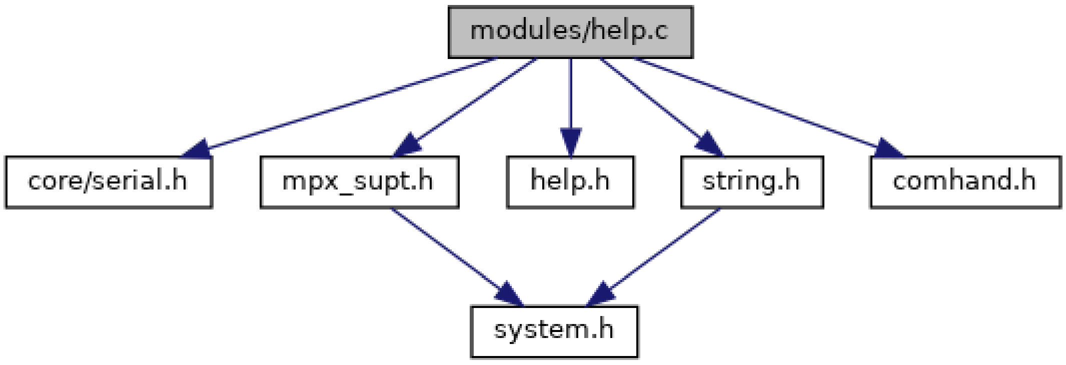
IMAGE

modules/help.c File Reference

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

Include dependency graph for help.c:

IMAGE



Functions

void **help** (char *msg)
void **display_help** (int count, char *name, char *usage, char *descript)

Detailed Description

Handles the help pages for all commands on the system

Function Documentation

void display_help (int *count*, char * *name*, char * *usage*, char * *descript*)

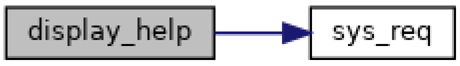
used in **help()** to print help page to terminal

Parameters

int	count, char *name, char *usage, char *descript,

Here is the call graph for this function:

IMAGE



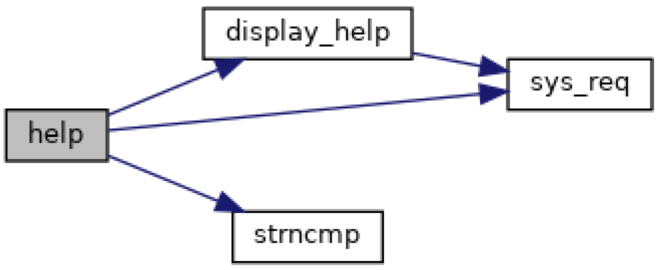
void help (char * *msg*)

Displays the correct help page for the given command

Parameters

char	*msg

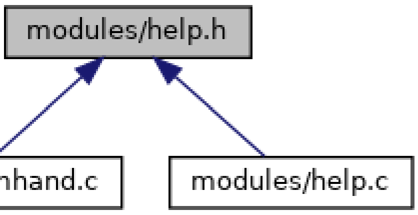
Here is the call graph for this function:



IMAGE

modules/help.h File Reference

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

IMAGE 

Functions

void **help** (char *msg)
void **display_help** (int count, char *name, char *usage, char *descript)

Function Documentation

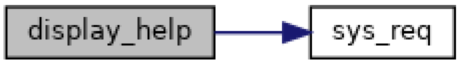
void display_help (int *count*, char * *name*, char * *usage*, char * *descript*)

used in **help()** to print help page to terminal

Parameters

int	count, char *name, char *usage, char *descript,

Here is the call graph for this function:

IMAGE 

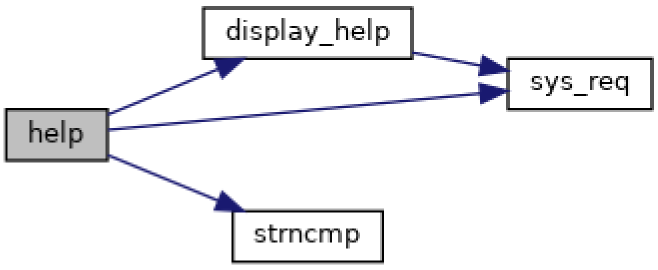
void help (char * *msg*)

Displays the correct help page for the given command

Parameters

char	*msg

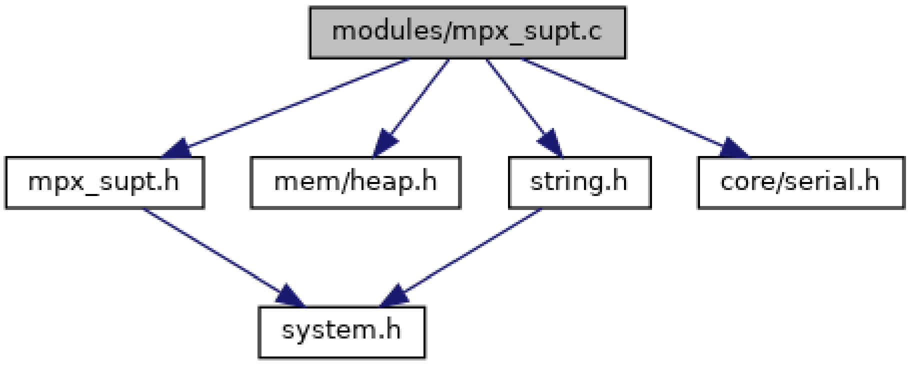
Here is the call graph for this function:

IMAGE 

modules/mpx_supt.c File Reference

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

Include dependency graph for mpx_supt.c:



IMAGE

Functions

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

Variables

param params

global variable containing parameter used when making system calls via sys_req

int **current_module** = -1

global for the current module

u32int(* **student_malloc**)(**u32int**)
int(* **student_free**)(void *)

Detailed Description

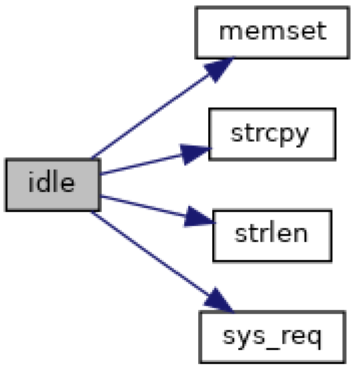
contains the MPX support functions

Function Documentation

void idle ()

The idle process, used in dispatching it will only be dispatched if NO other processes are available to execute.

Here is the call graph for this function:



IMAGE

void mpx_init (int *cur_mod*)

```
Initialize MPX support software, based
     on the current module.  The operation of
     MPX will changed based on the module selected.
     THIS must be called as the first executable
     statement inside your command handler.
```

Parameters

int	cur_mod

void* sys_alloc_mem (u32int *size*)

Allocates a block of memory (similar to malloc)

Parameters

u32int	size

int sys_free_mem (void * *ptr*)

Frees memory

Parameters

void	*ptr

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

This function is use to issue system requests for service.

Parameters

int	op_code, int device_id, char *buffer_ptr, int *count_ptr

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

Sets the memory free function for sys_free_mem

Parameters

s1-destination,s2-source	

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

Sets the memory allocation function for sys_alloc_mem

Parameters

Function	pointer

Variable Documentation

int current_module = -1

global for the current module

param params

global variable containing parameter used when making system calls via sys_req

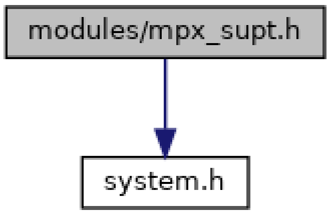
int(* student_free) (void *)

u32int(* student_malloc) (u32int)

modules/mpx_supt.h File Reference

`#include <system.h>`

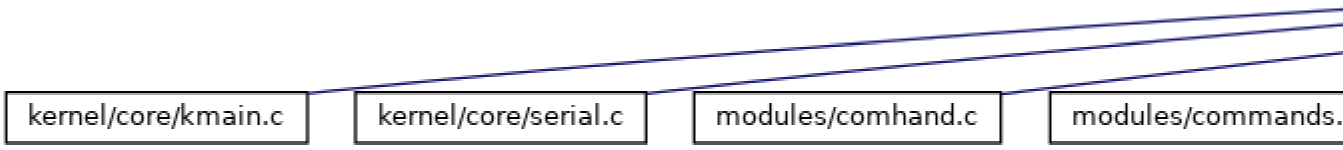
Include dependency graph for mpx_supt.h:



IMAGE

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

IMAGE



Data Structures

struct **param**

Macros

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

Functions

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

Macro Definition Documentation

#define COM_PORT 222

#define DEFAULT_DEVICE 111

#define EXIT 0

#define FALSE 0

#define IDLE 1

#define INVALID_BUFFER 1000

#define INVALID_COUNT 2000

#define INVALID_OPERATION 4

#define IO_MODULE 10

#define MEM_MODULE 11

#define MODULE_F 9

#define MODULE_R1 0

#define MODULE_R2 1

#define MODULE_R3 2

#define MODULE_R4 4

#define MODULE_R5 8

#define READ 2

#define TRUE 1

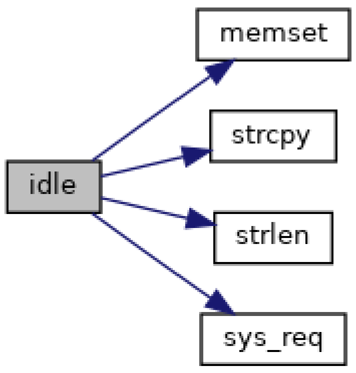
#define WRITE 3

Function Documentation

void idle ()

The idle process, used in dispatching it will only be dispatched if NO other processes are available to execute.

Here is the call graph for this function:



IMAGE

void mpx_init (int *cur_mod*)

```
Initialize MPX support software, based
     on the current module.  The operation of
     MPX will changed based on the module selected.
     THIS must be called as the first executable
     statement inside your command handler.
```

Parameters

int	cur_mod

void* sys_alloc_mem (u32int *size*)

Allocates a block of memory (similar to malloc)

Parameters

u32int	size

int sys_free_mem (void * *ptr*)

Frees memory

Parameters

void	*ptr

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

This function is use to issue system requests for service.

Parameters

int	op_code, int device_id, char *buffer_ptr, int *count_ptr

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

Sets the memory free function for sys_free_mem

Parameters

s1-destination,s2-source	

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

Sets the memory allocation function for sys_alloc_mem

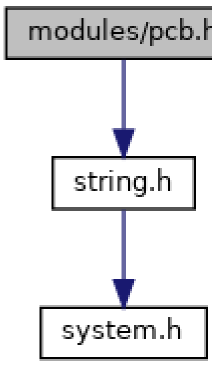
Parameters

Function	pointer

modules/pcb.h File Reference

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

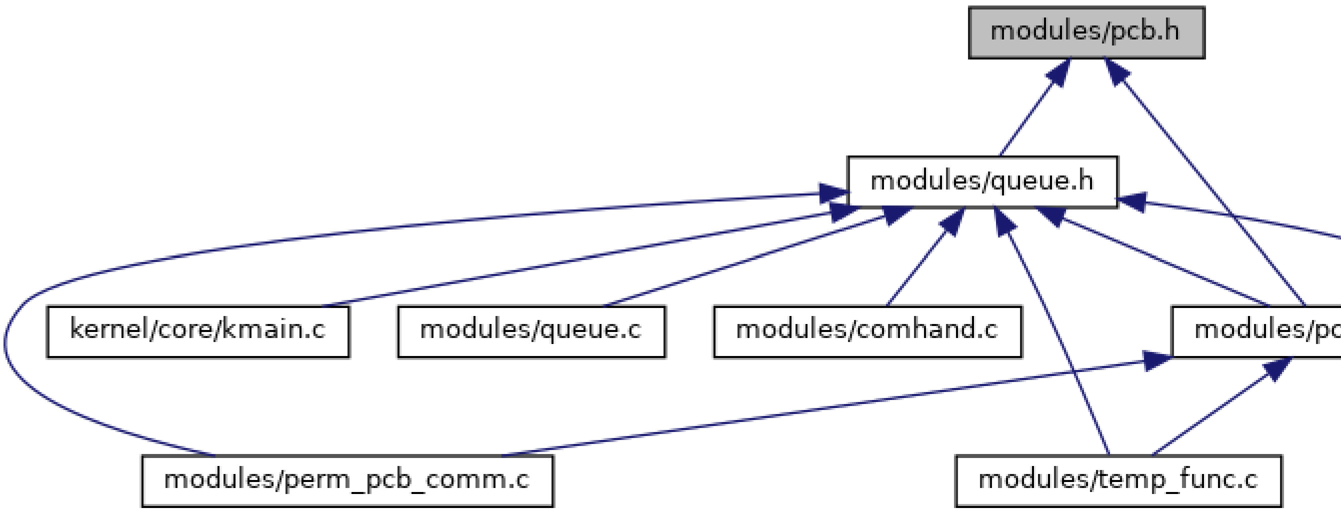
Include dependency graph for pcb.h:



IMAGE

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

IMAGE



Data Structures

struct **pcb**

Macros

#define **STACK_SIZE** 1024
#define **APPLICATION_P** 1

type of process

#define **SYSTEM_P** 0
#define **READY** 0
#define **RUNNING** 1
#define **BLOCKED** 2
#define **SUSPSEND** 1
#define **NOT_SUSP** 0

Typedefs

typedef struct **pcb pcb**

Detailed Description

Defines the PCB (Process Control Block) struct

Macro Definition Documentation

#define APPLICATION_P 1

type of process

#define BLOCKED 2

#define NOT_SUSP 0

#define READY 0

#define RUNNING 1

#define STACK_SIZE 1024

#define SUSPSEND 1

#define SYSTEM_P 0

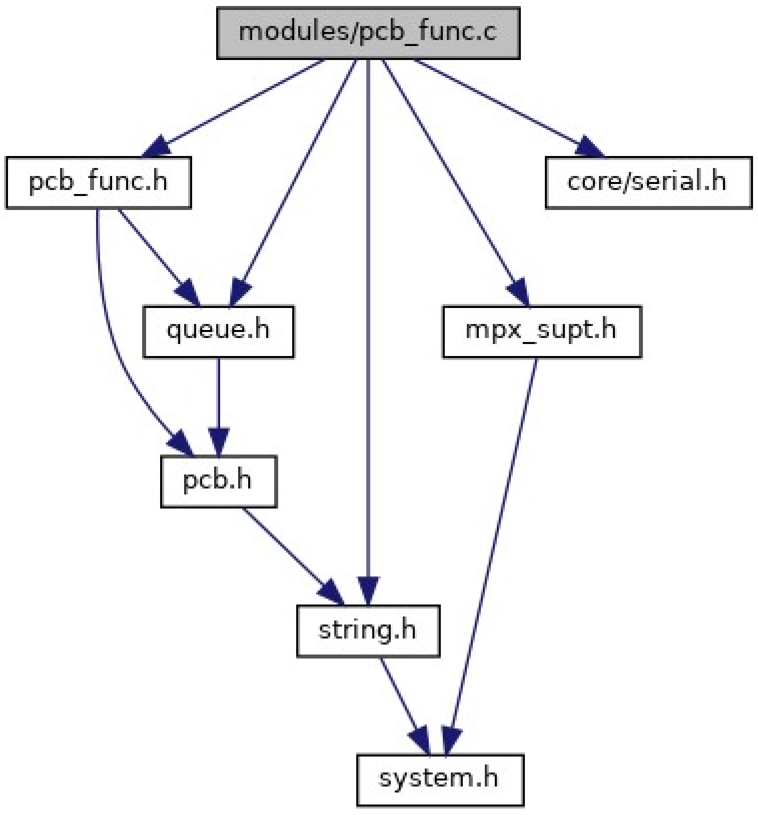
Typedef Documentation

typedef struct pcb pcb

modules/pcb_func.c File Reference

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

Include dependency graph for pcb_func.c:



IMAGE

Functions

pcb * **allocatePCB** ()
int **freePCB** (**pcb** ***pcb**)
pcb * **setupPCB** (char *name, int class, int priority)
pcb * **findPCB** (char *name)
void **insertPCB** (**pcb** ***pcb**)
int **removePCB** (**pcb** ***pcb**)

Variables

pcb * **removed**
pcb * **temp**
pcb * **parent**

Detailed Description

Implementation of pcb functions

Function Documentation

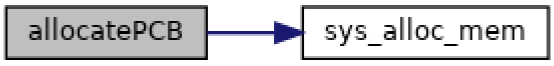
pcb* allocatePCB ()

Allocates new memory for new PCB

Returns

PCB pointer

Here is the call graph for this function:

IMAGE 

pcb* findPCB (char * *name*)

Searches all queues for a process with a given name

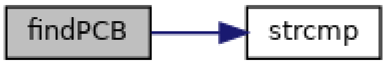
Parameters

Process	name

Returns

PCB pointer

Here is the call graph for this function:

IMAGE 

int freePCB (pcb * *pcb*)

Frees all memory associated with a given PCB

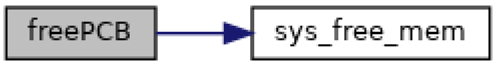
Parameters

PCB	pointer

Returns

success or error code

Here is the call graph for this function:

IMAGE 

void insertPCB (pcb * *pcb*)

Inserts a PCB into the appropriate queue

Parameters

PCB	pointer

int removePCB (pcb * *pcb*)

Removes a PCB from the queue in which it is currently stored

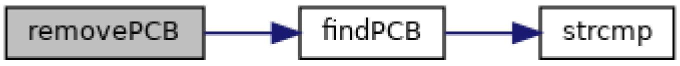
Parameters

PCB	pointer

Returns

success or error code

Here is the call graph for this function:

IMAGE 

pcb* setupPCB (char * *name*, int *class*, int *priority*)

Creates an empty PCB, intializes PCB and sets the PCB state to ready, not suspended

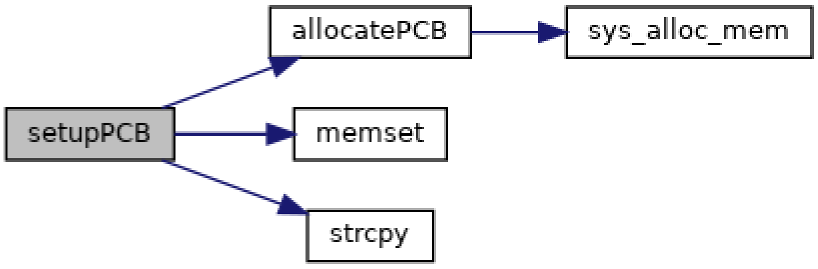
Parameters

name,class,priorit y	

Returns

PCB pointer

Here is the call graph for this function:



IMAGE

Variable Documentation

pcb* parent

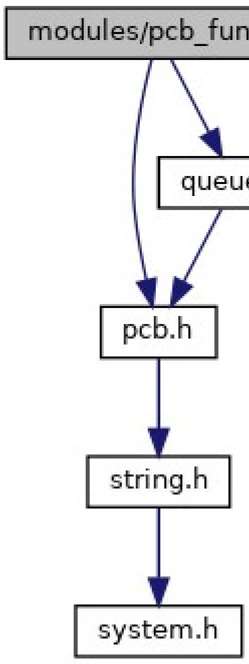
pcb* removed

pcb* temp

modules/pcb_func.h File Reference

```
#include "pcb.h"
#include "queue.h"
```

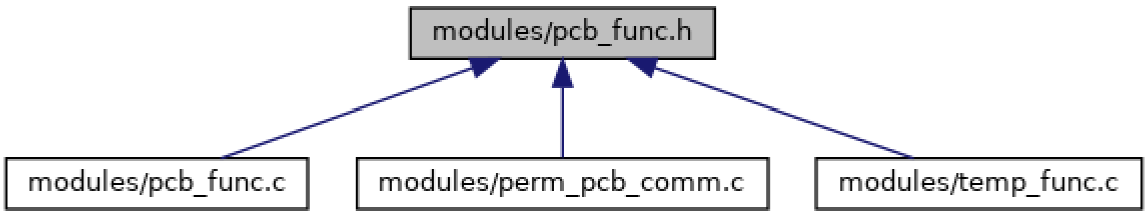
Include dependency graph for pcb_func.h:



IMAGE

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

IMAGE



Functions

pcb * **allocatePCB** ()
int **freePCB** (**pcb** ***pcb**)
pcb * **setupPCB** (char *name, int class, int priority)
pcb * **findPCB** (char *name)
void **insertPCB** (**pcb** ***pcb**)
int **removePCB** (**pcb** ***pcb**)

Detailed Description

Defines all of the pcb operation functions as internal procedures

---

Function Documentation

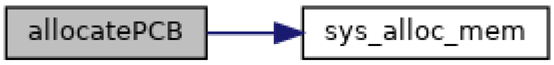
pcb* allocatePCB ()

Allocates new memory for new PCB

Returns

PCB pointer

Here is the call graph for this function:

IMAGE 

pcb* findPCB (char * *name*)

Searches all queues for a process with a given name

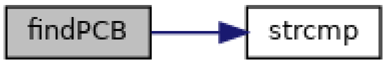
Parameters

Process	name

Returns

PCB pointer

Here is the call graph for this function:

IMAGE 

int freePCB (pcb * *pcb*)

Frees all memory associated with a given PCB

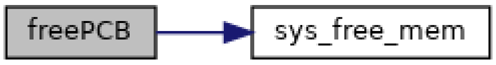
Parameters

PCB	pointer

Returns

success or error code

Here is the call graph for this function:

IMAGE 

void insertPCB (pcb * *pcb*)

Inserts a PCB into the appropriate queue

Parameters

PCB	pointer

int removePCB (pcb * *pcb*)

Removes a PCB from the queue in which it is currently stored

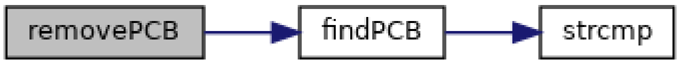
Parameters

PCB	pointer

Returns

success or error code

Here is the call graph for this function:

IMAGE 

pcb* setupPCB (char * *name*, int *class*, int *priority*)

Creates an empty PCB, intializes PCB and sets the PCB state to ready, not suspended

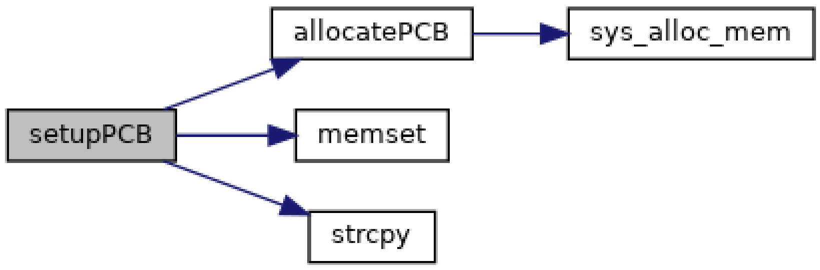
Parameters

name,class,priorit y	

Returns

PCB pointer

Here is the call graph for this function:

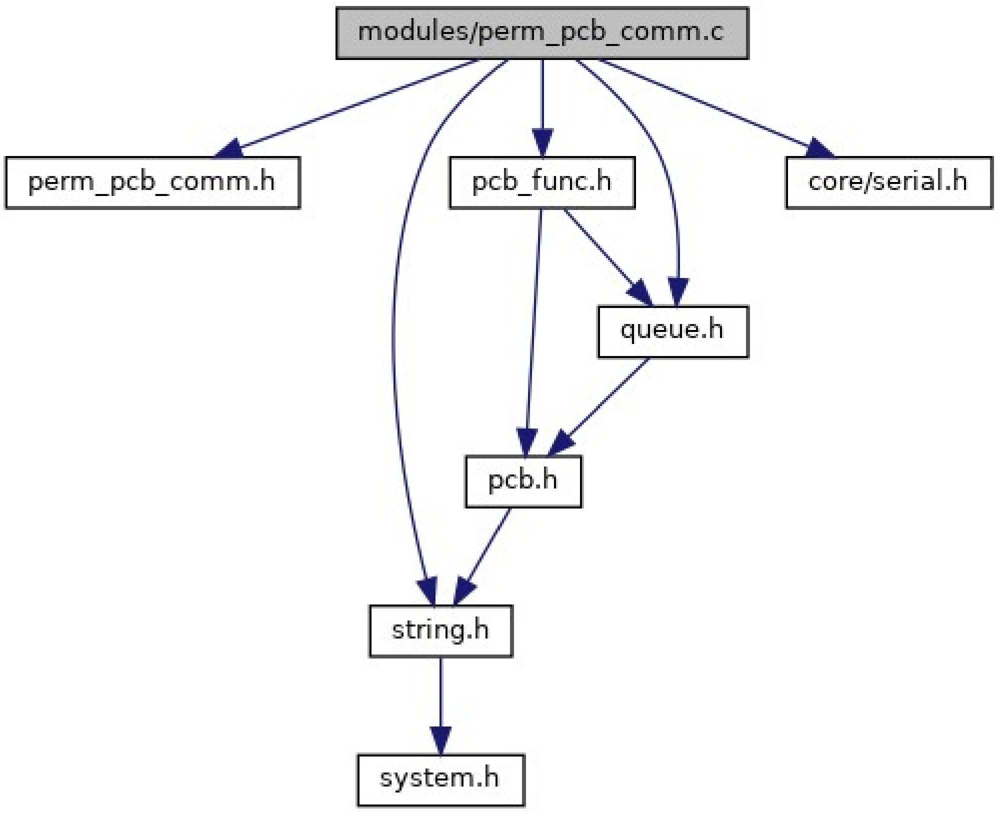


IMAGE

modules/perm_pcb_comm.c File Reference

```
#include "perm_pcb_comm.h"
#include <string.h>
#include "pcb_func.h"
#include <core/serial.h>
#include "queue.h"
```

Include dependency graph for perm_pcb_comm.c:



IMAGE

Functions

void **suspendPCB** (char *name)
void **resumePCB** (char *name)
void **setPCBPriority** (char *name, int priority)
void **showPCB** (char *name)
void **showReadyPCB** ()
void **showBlockedPCB** ()
void **showAllPCB** ()

Variables

int **flag** = 0

Detailed Description

Function implementations of permanent PCB functions for user commands

Function Documentation

void resumePCB (char * *name*)

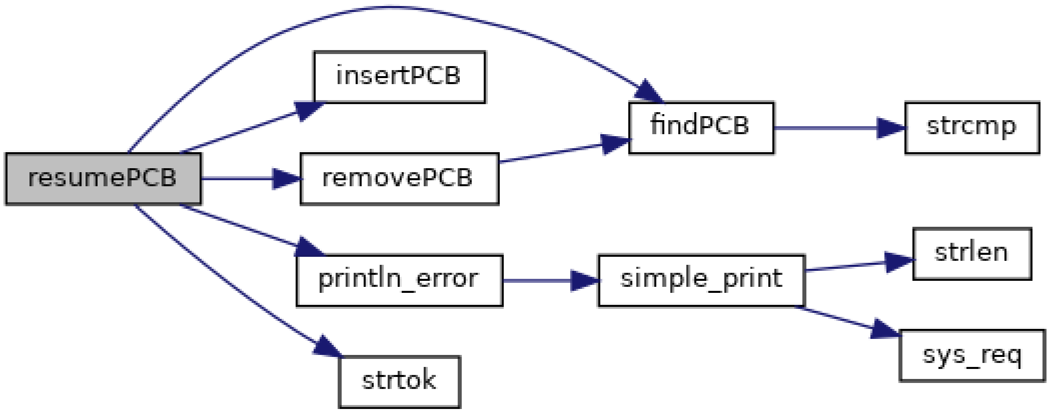
Places PCB into the not suspended state and reinserts it into the appropriate queue

Parameters

char	*name

Here is the call graph for this function:

IMAGE



void setPCBPriority (char * *name*, int *priority*)

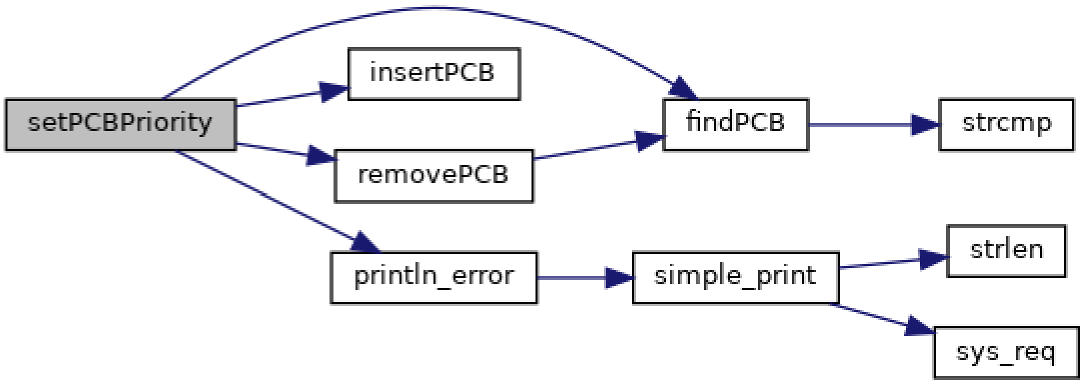
Sets a PCB's priority and reinserts the process into the correct place in the correct queue

Parameters

char	*name, int priority

Here is the call graph for this function:

IMAGE

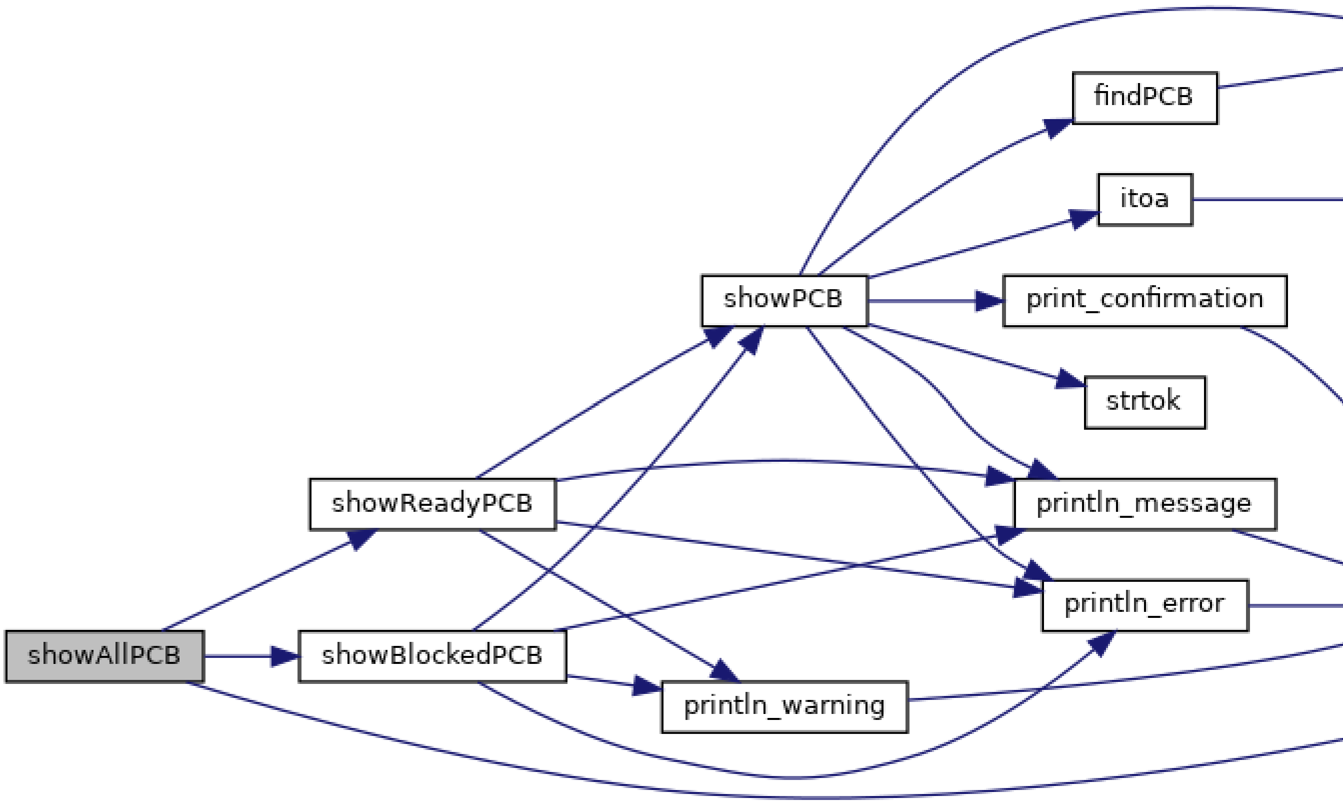


void showAllPCB ()

Shows all PCBs in all of the queues

Here is the call graph for this function:

IMAGE

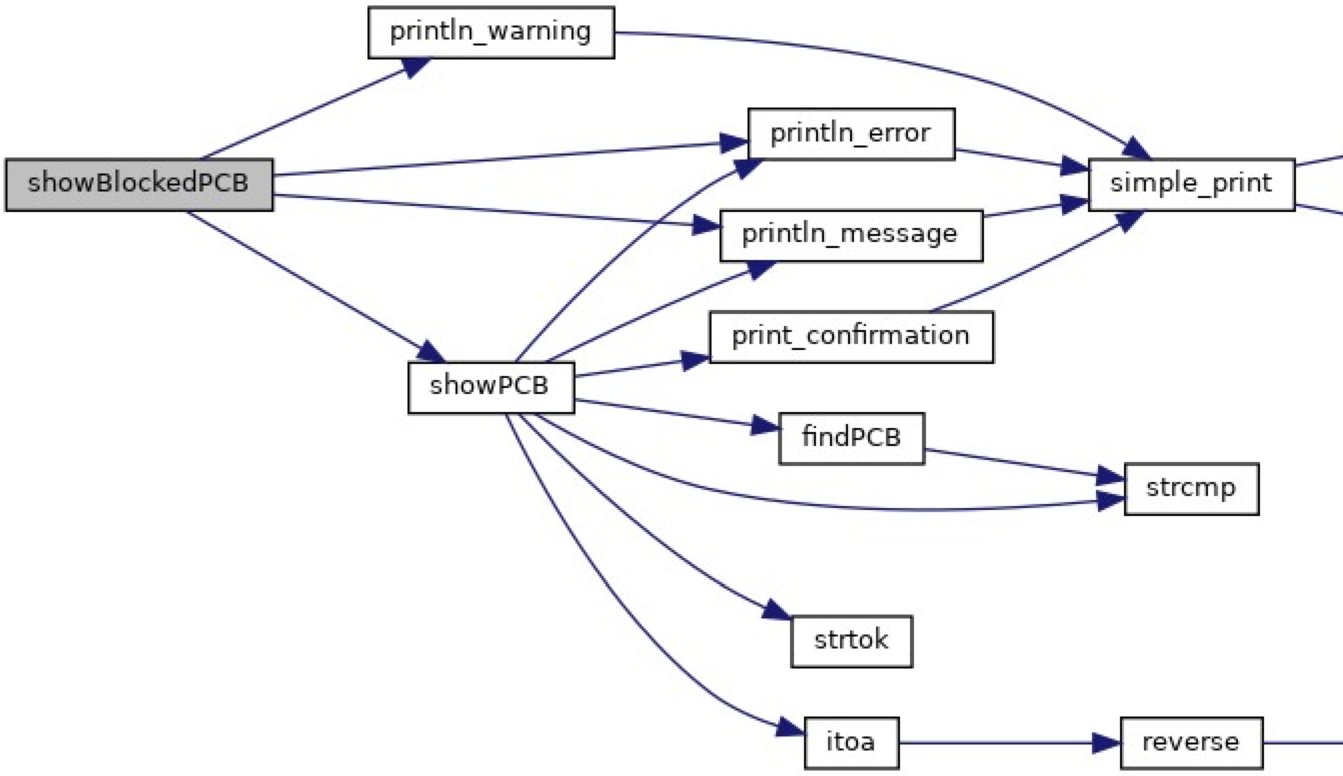


void showBlockedPCB ()

Displays all of the PCBs in the blocked queues

Here is the call graph for this function:

IMAGE



void showPCB (char * *name*)

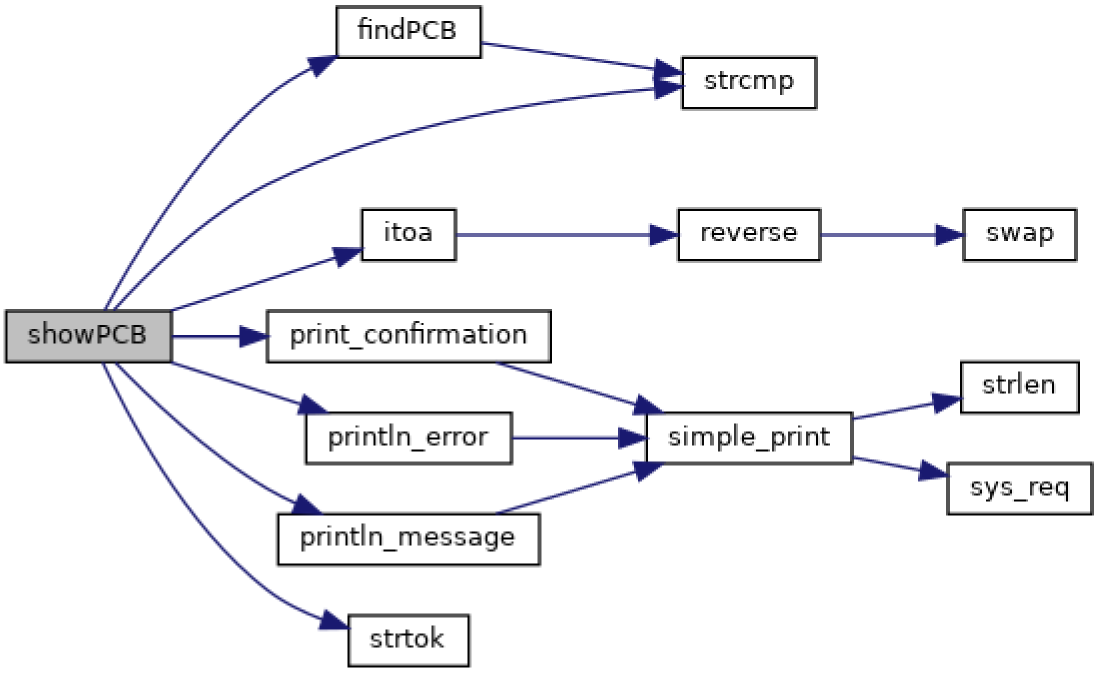
Displays the attributes for a PCB

Parameters

char	*name

Here is the call graph for this function:

IMAGE

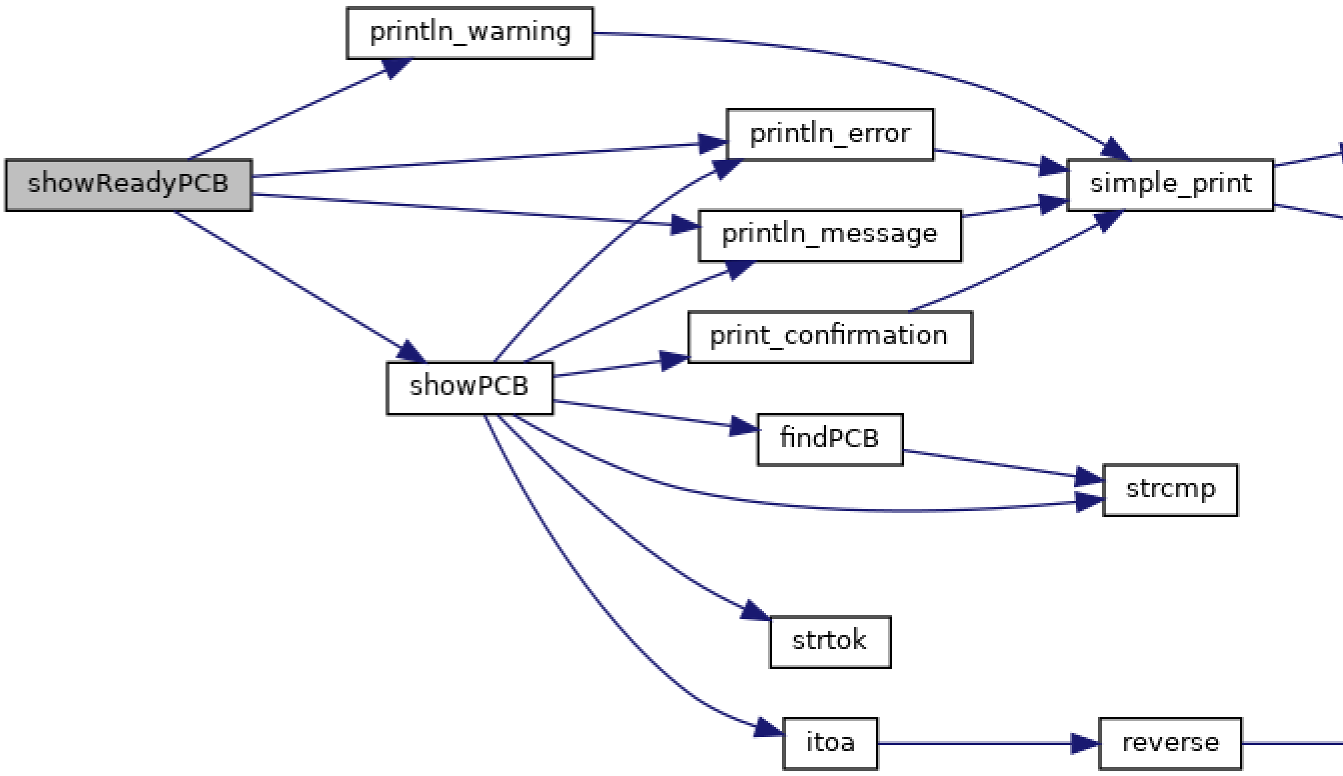


void showReadyPCB ()

Displays all of the PCBs in the ready queues

Here is the call graph for this function:

IMAGE



void suspendPCB (char * *name*)

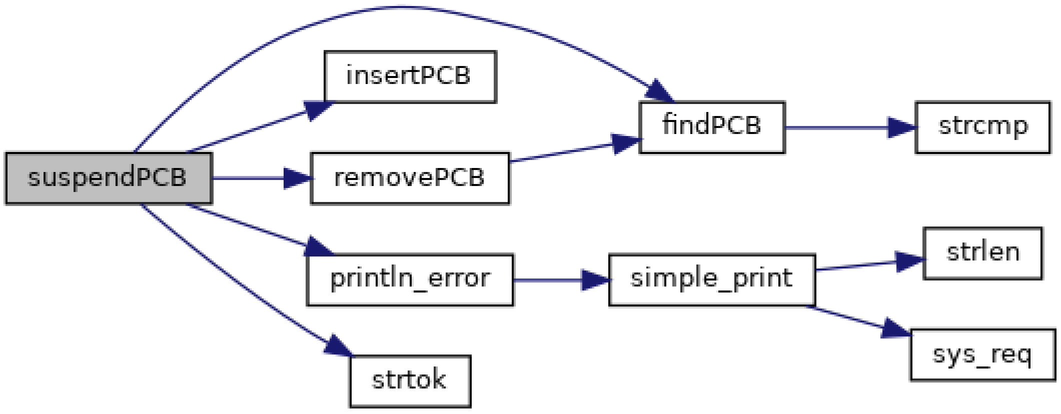
Places the PCB into the suspended state and reinserts into the appropriate queue

Parameters

char	*name

Here is the call graph for this function:

IMAGE



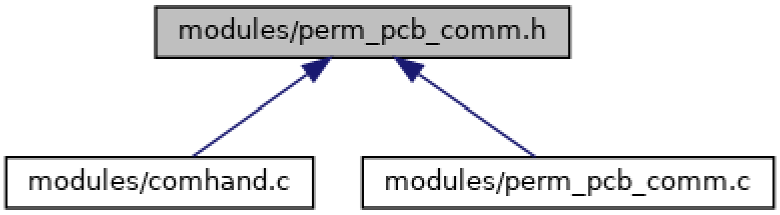
Variable Documentation

int flag = 0

modules/perm_pcb_comm.h File Reference

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

IMAGE



Functions

void **suspendPCB** (char *name)
void **resumePCB** (char *name)
void **setPCBPriority** (char *name, int priority)
void **showPCB** (char *name)
void **showReadyPCB** ()
void **showBlockedPCB** ()
void **showAllPCB** ()

Detailed Description

Function definitions for permanent pcb user commands

Function Documentation

void resumePCB (char * *name*)

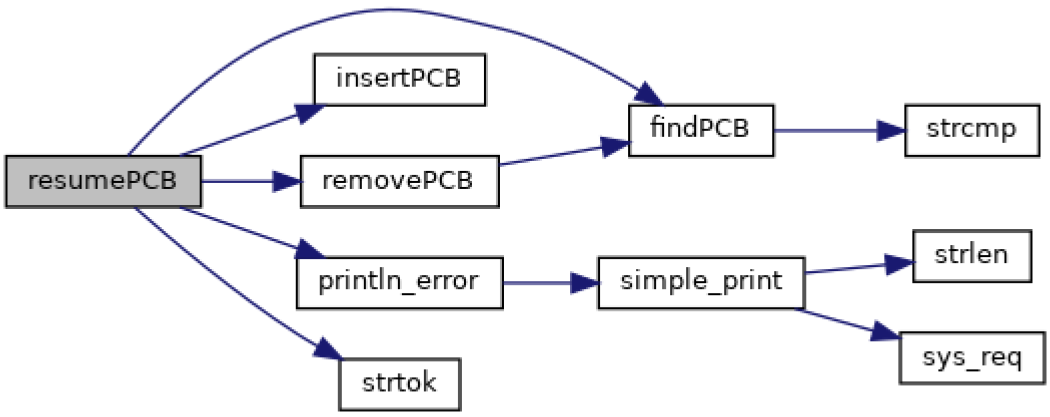
Places PCB into the not suspended state and reinserts it into the appropriate queue

Parameters

char	*name

Here is the call graph for this function:

IMAGE



void setPCBPriority (char * *name*, int *priority*)

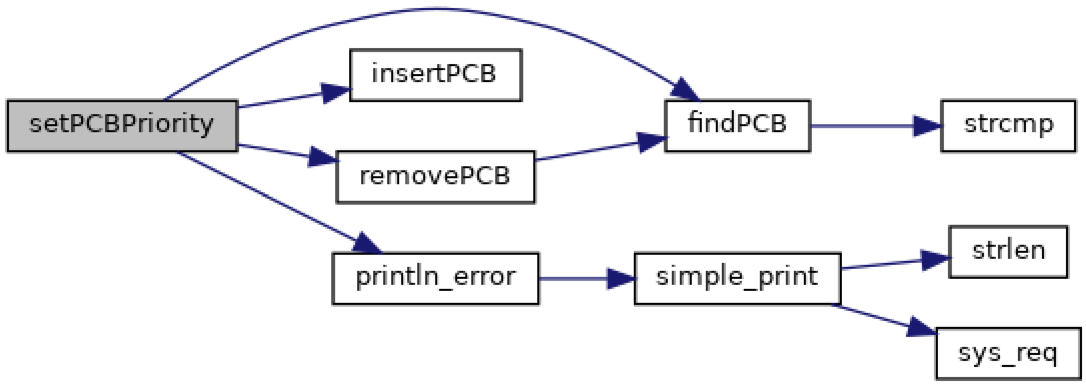
Sets a PCB's priority and reinserts the process into the correct place in the correct queue

Parameters

char	*name, int priority

Here is the call graph for this function:

IMAGE

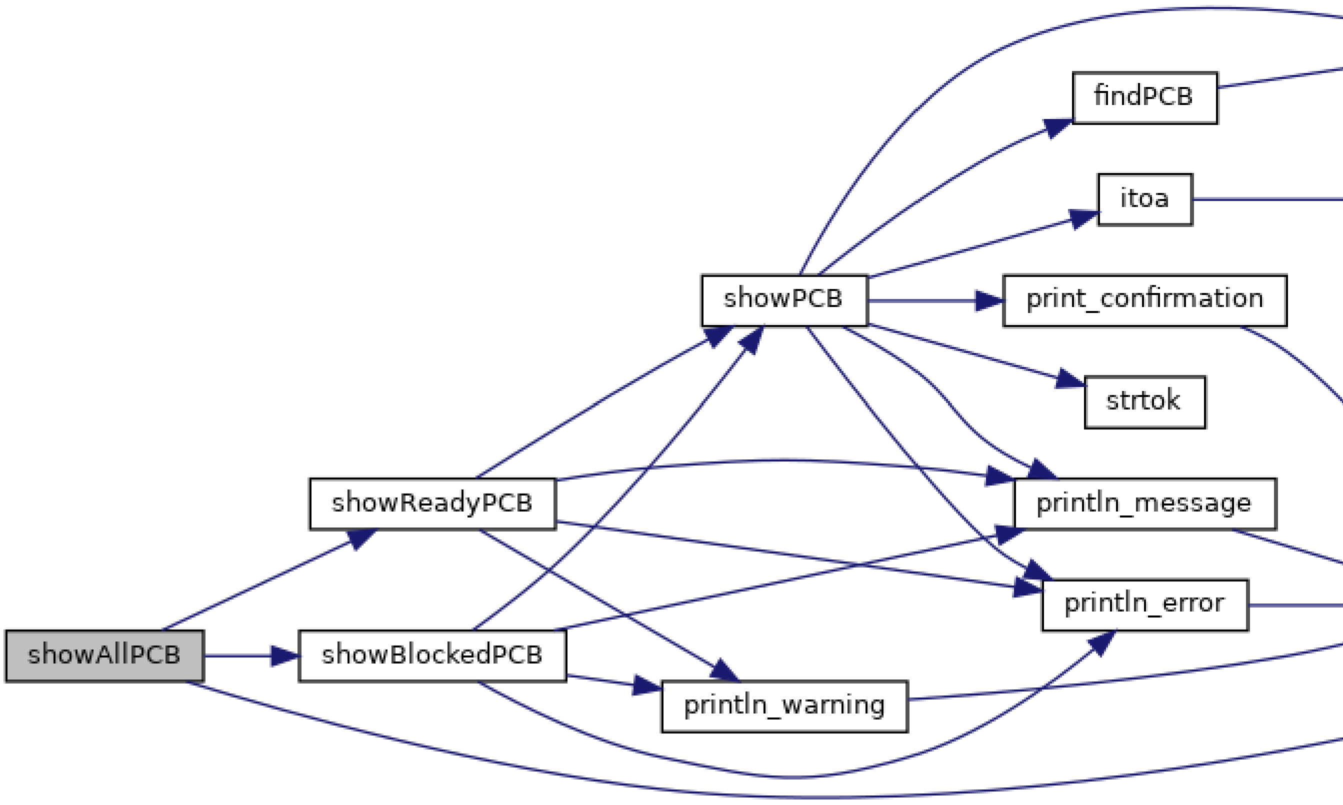


void showAllPCB ()

Shows all PCBs in all of the queues

Here is the call graph for this function:

IMAGE

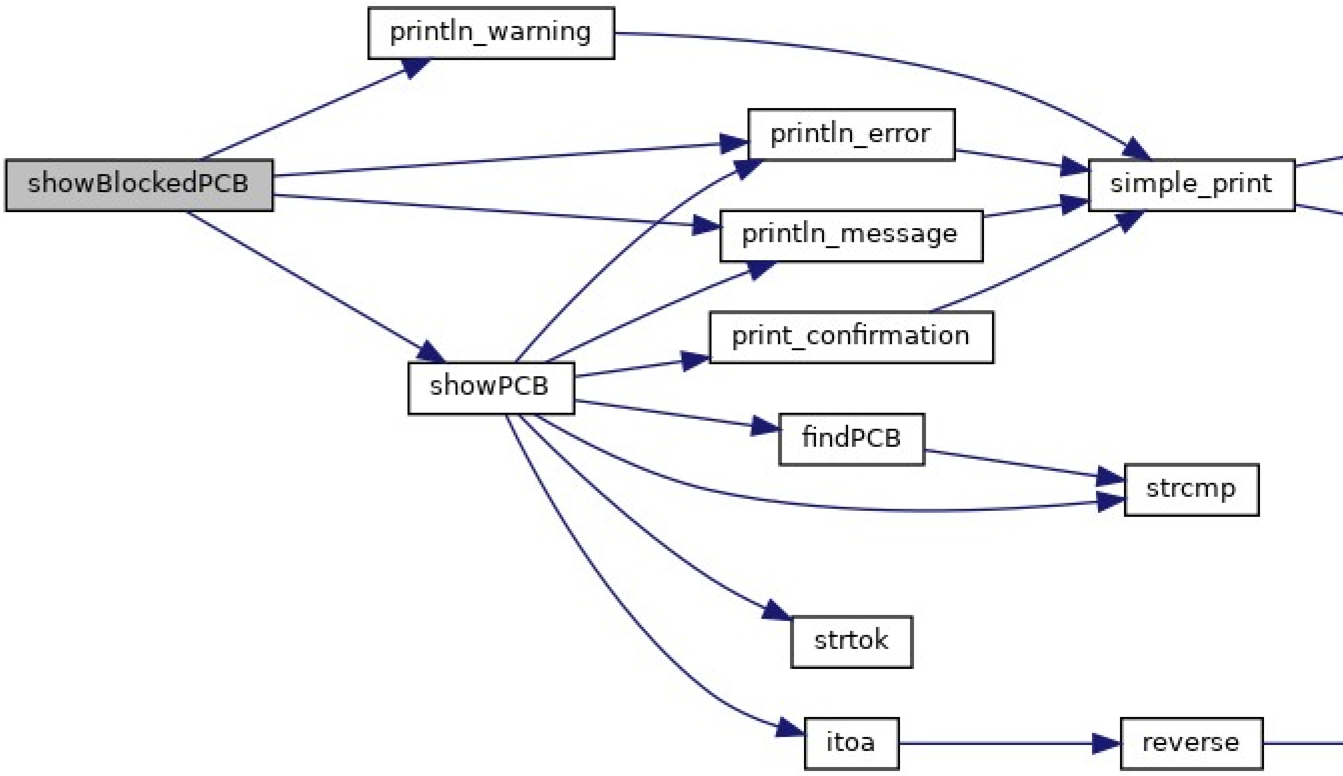


void showBlockedPCB ()

Displays all of the PCBs in the blocked queues

Here is the call graph for this function:

IMAGE



void showPCB (char * *name*)

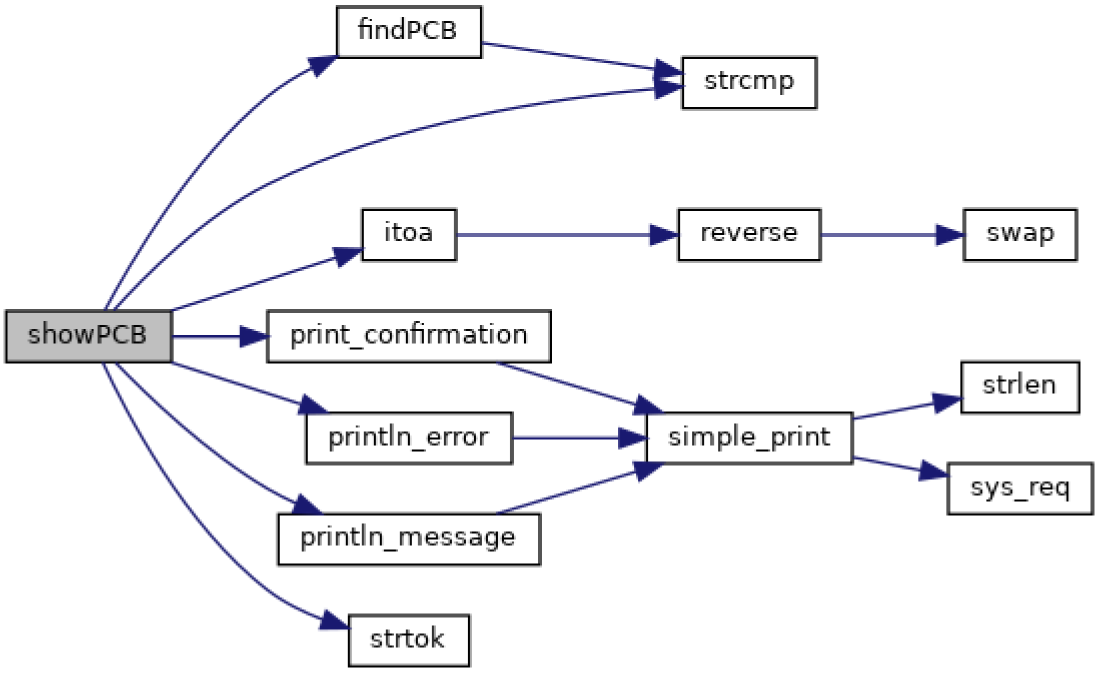
Displays the attributes for a PCB

Parameters

char	*name

Here is the call graph for this function:

IMAGE

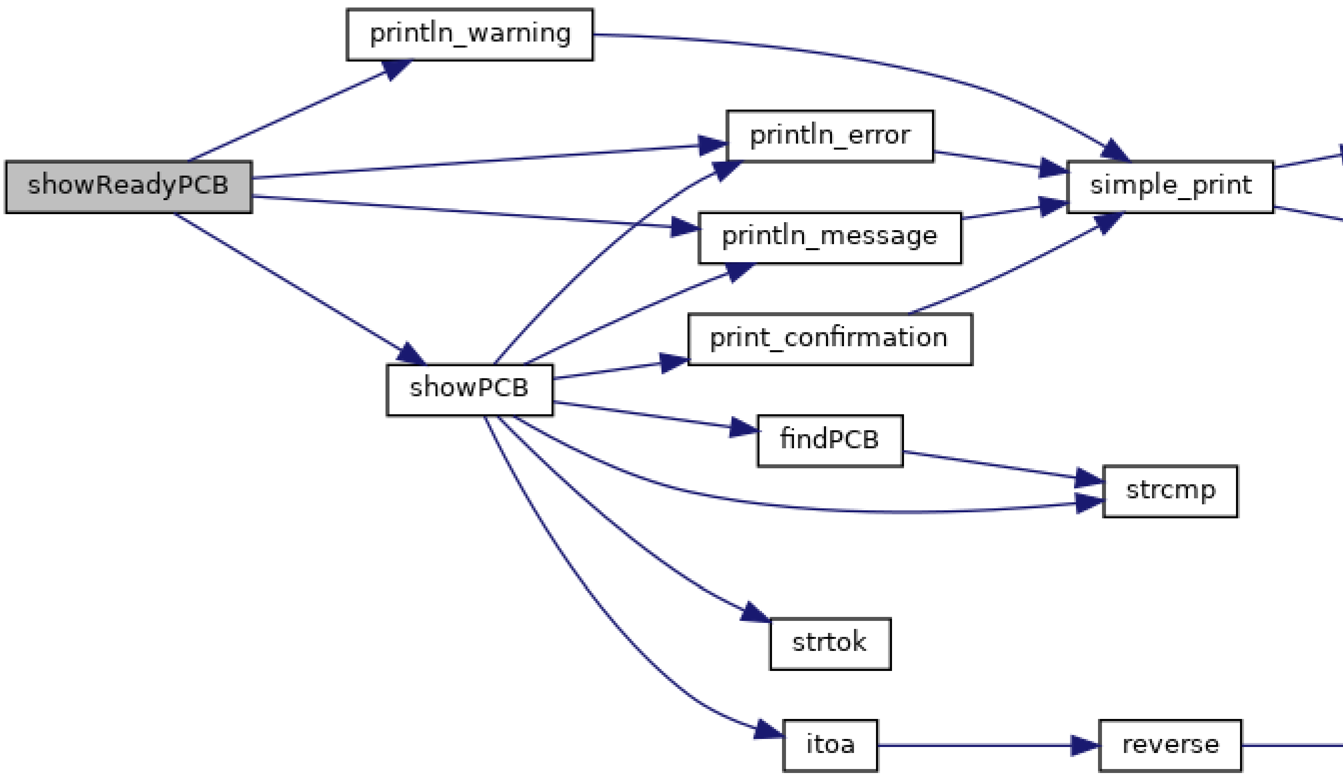


void showReadyPCB ()

Displays all of the PCBs in the ready queues

Here is the call graph for this function:

IMAGE



void suspendPCB (char *   *name*)

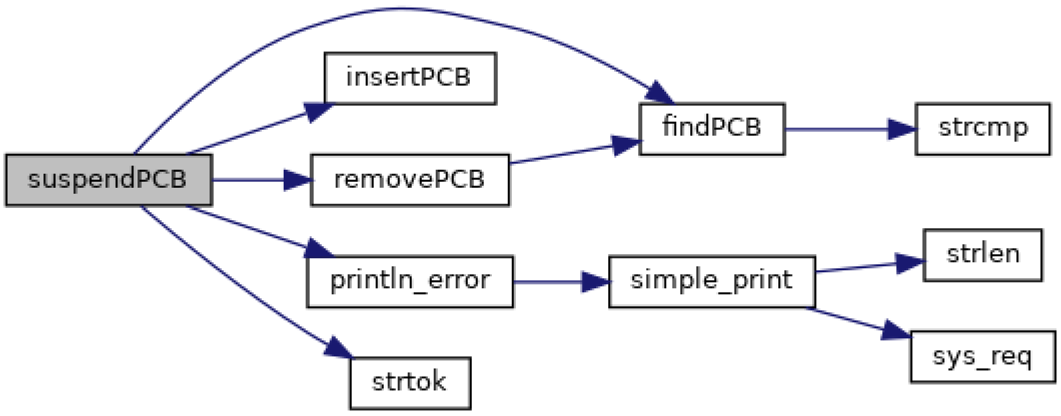
Places the PCB into the suspended state and reinserts into the appropriate queue

Parameters

char	*name

Here is the call graph for this function:

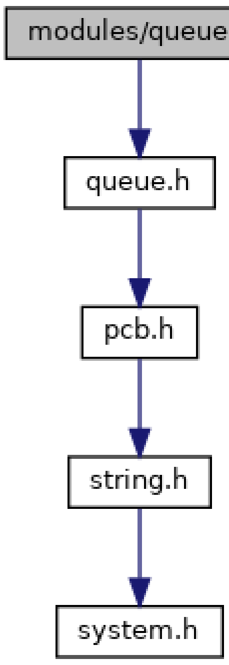
IMAGE



modules/queue.c File Reference

`#include "queue.h"`

Include dependency graph for queue.c:



IMAGE

Variables

queue readyQueue = {0, **NULL**, **NULL**}
queue readySuspendedQueue = {0, **NULL**, **NULL**}
queue blockedQueue = {0, **NULL**, **NULL**}
queue blockedSuspendedQueue = {0, **NULL**, **NULL**}

Detailed Description

Defines global queues of read, ready-suspended, blocked, and blocked-suspended

---

Variable Documentation

queue blockedQueue = {0, NULL, NULL}

queue blockedSuspendedQueue = {0, NULL, NULL}

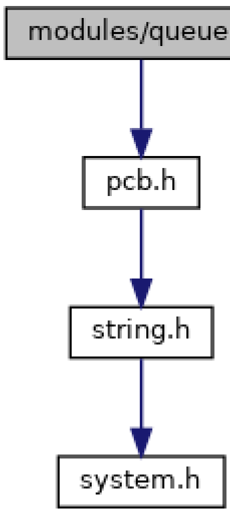
queue readyQueue = {0, NULL, NULL}

queue readySuspendedQueue = {0, NULL, NULL}

modules/queue.h File Reference

`#include "pcb.h"`

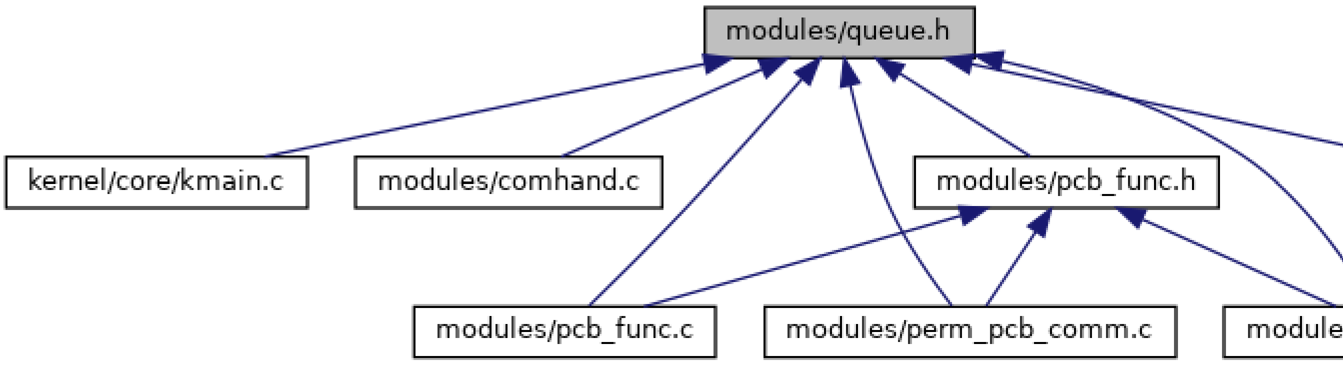
Include dependency graph for queue.h:



IMAGE

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

IMAGE



Data Structures

struct **queue**

Typedefs

typedef struct **queue queue**

Variables

queue readyQueue
queue readySuspendedQueue
queue blockedQueue
queue blockedSuspendedQueue

Detailed Description

Defines the struct of a queue to use for containing PCBs

Typedef Documentation

typedef struct queue queue

Variable Documentation

queue blockedQueue

queue blockedSuspendedQueue

queue readyQueue

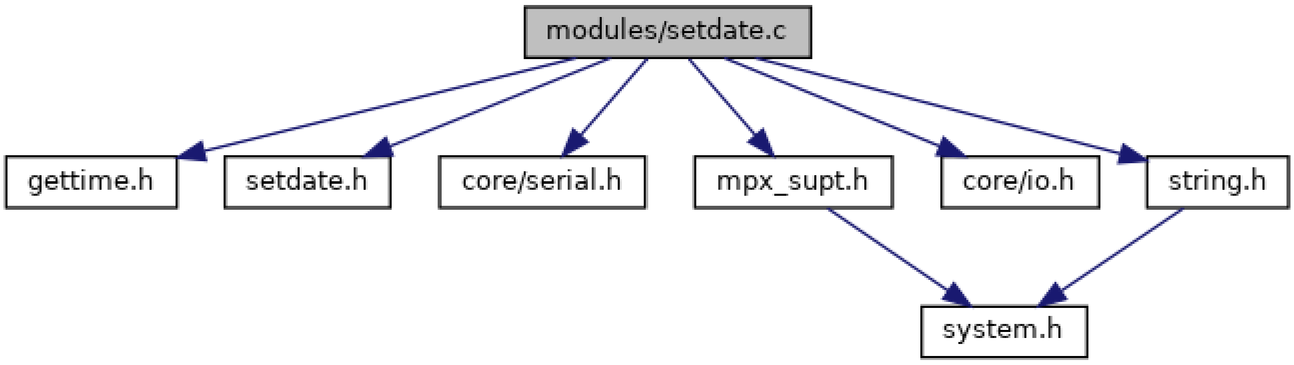
queue readySuspendedQueue

modules/setdate.c File Reference

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

Include dependency graph for setdate.c:

IMAGE



Functions

void **setdate** (char *date)

Detailed Description

contains **setdate(char *date)** function to set a new date on the system

Function Documentation

void setdate (char * *date*)

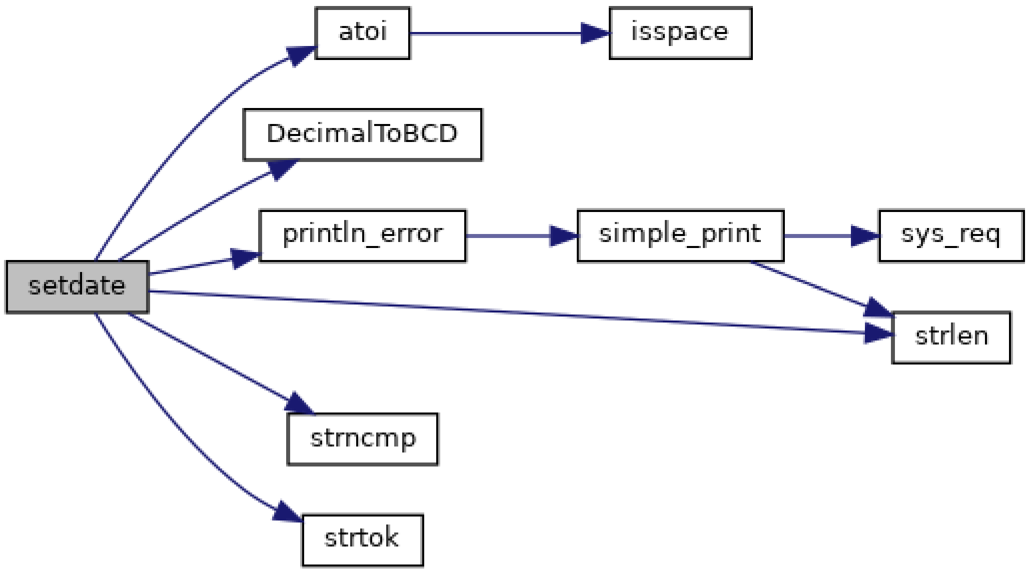
sets the date to the given input

Parameters

char	*date

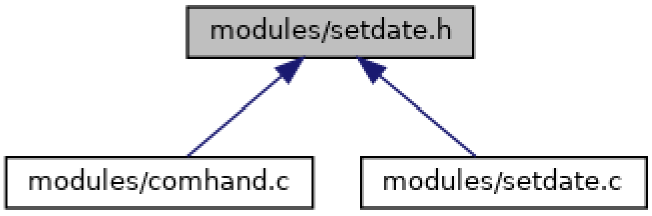
Here is the call graph for this function:

IMAGE


setdate
atoi
isspace
DecimalToBCD
println_error
simple_print
sys_req
strlen
strncmp
strtok

modules/setdate.h File Reference

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

IMAGE 

Functions

void **setdate** (char *date)

Function Documentation

void setdate (char * *date*)

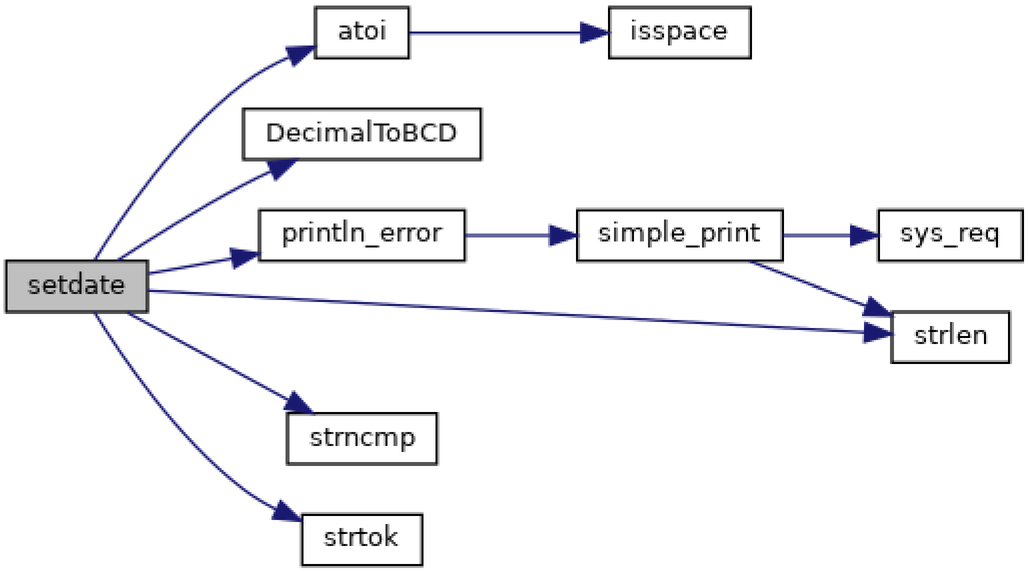
sets the date to the given input

Parameters

char	*date

Here is the call graph for this function:

IMAGE

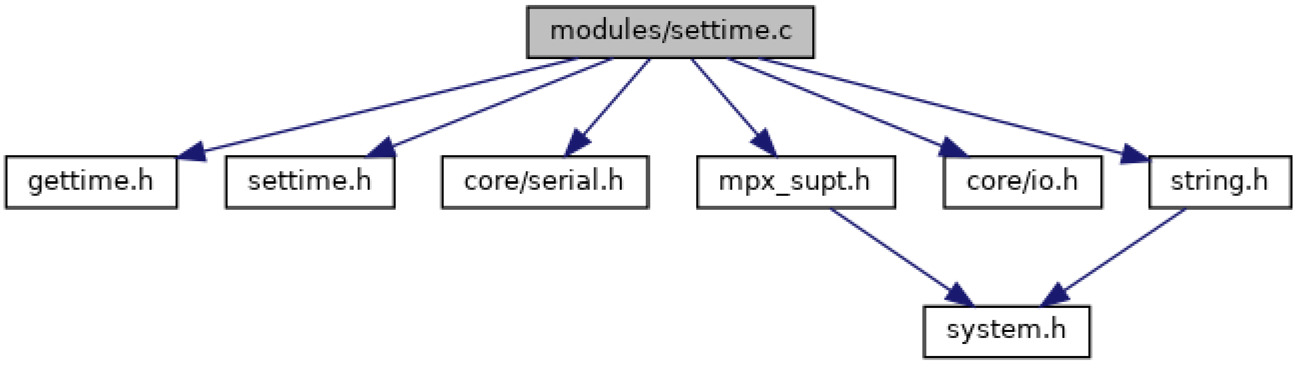


modules/settime.c File Reference

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

Include dependency graph for settime.c:

IMAGE



Functions

void **settime** (char *time)

Detailed Description

Sets a new time given by the user

Function Documentation

void settime (char * *time*)

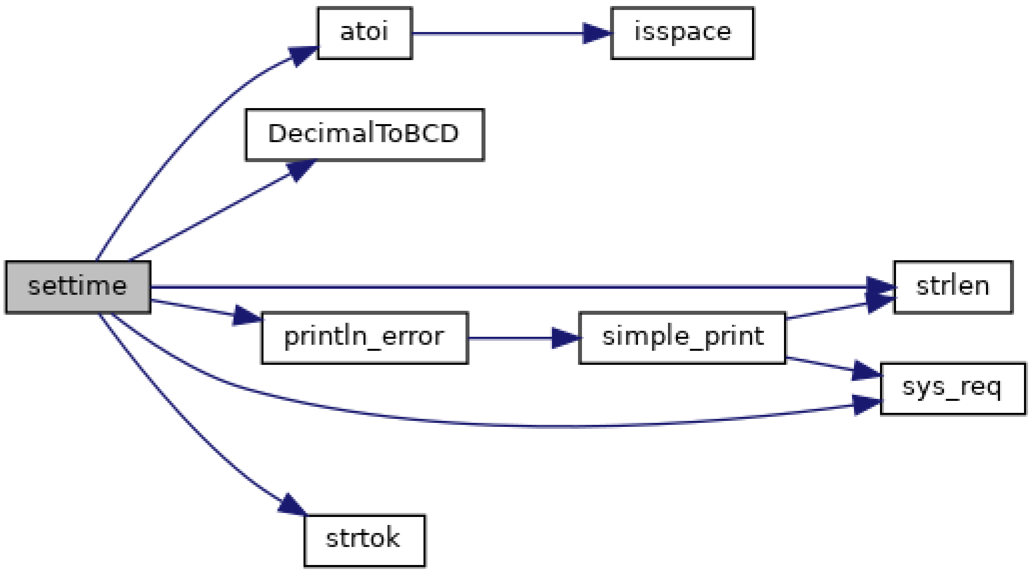
Allows user to change the time on the system

Parameters

char	*time

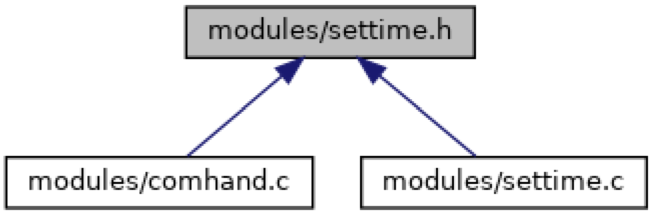
Here is the call graph for this function:

IMAGE


settime
atoi
isspace
DecimalToBCD
strlen
println_error
simple_print
sys_req
strtok

modules/settime.h File Reference

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

IMAGE 

Functions

void **settime** (char *time)

Function Documentation

void settime (char * *time*)

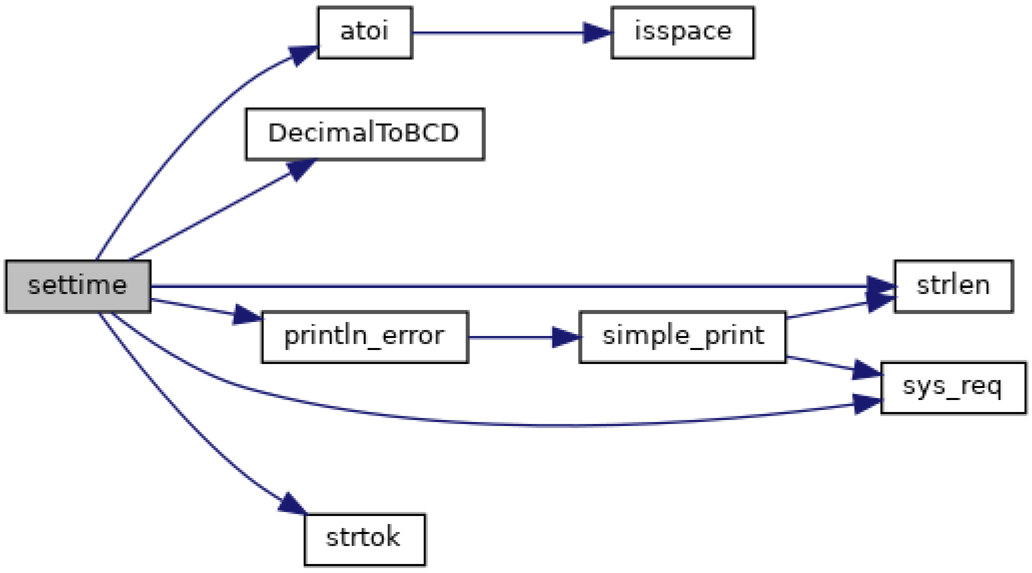
Allows user to change the time on the system

Parameters

char	*time

Here is the call graph for this function:

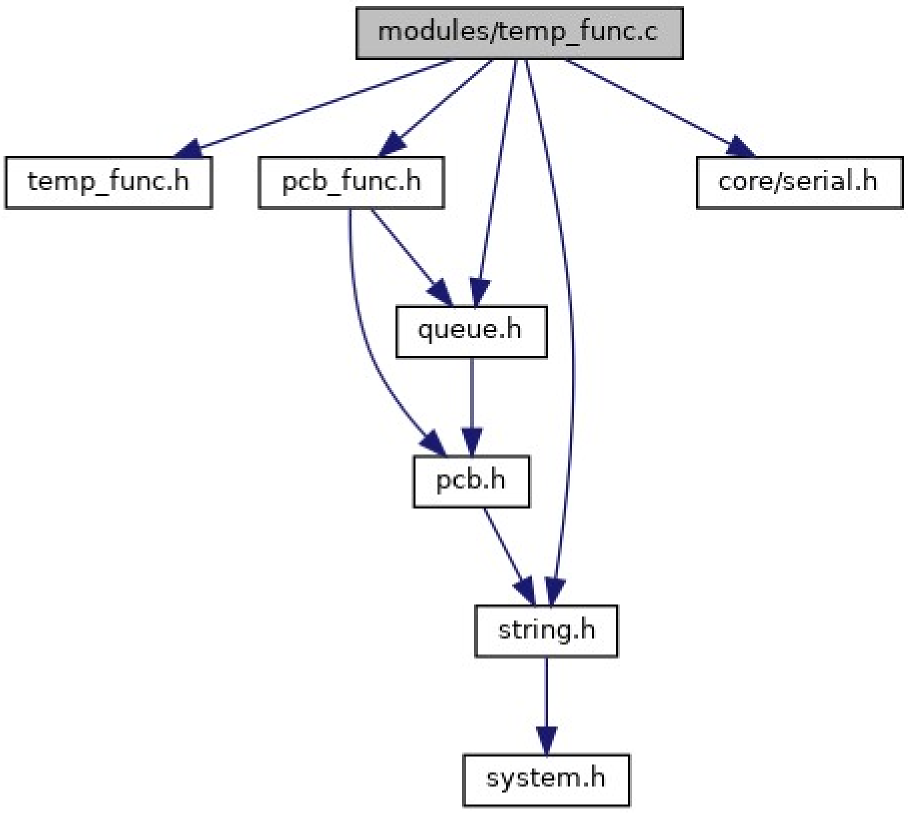
IMAGE



modules/temp_func.c File Reference

```
#include "temp_func.h"
#include "pcb_func.h"
#include <string.h>
#include <core/serial.h>
#include "queue.h"
```

Include dependency graph for temp_func.c:



IMAGE

Functions

void **createPCB** (char ***params**)
void **deletePCB** (char *name)
void **blockPCB** (char *name)
void **unblockPCB** (char *name)

Detailed Description

Implementation of temprorary pcb functions/commands

Function Documentation

void blockPCB (char * *name*)

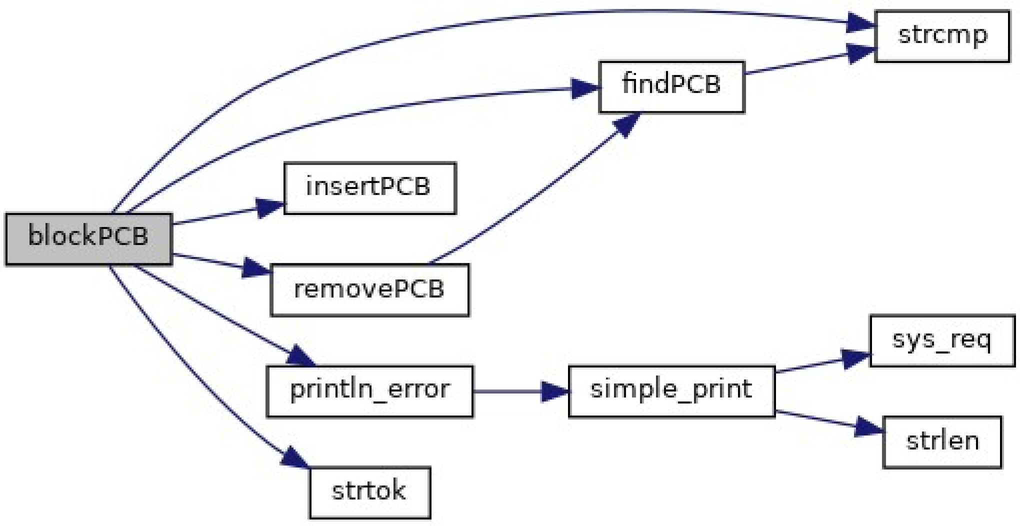
Finds PCB and sets its stae to blocked and reinserts into the appropriate queue

Parameters

char	*name

Here is the call graph for this function:

IMAGE



void createPCB (char * ***params*****)**

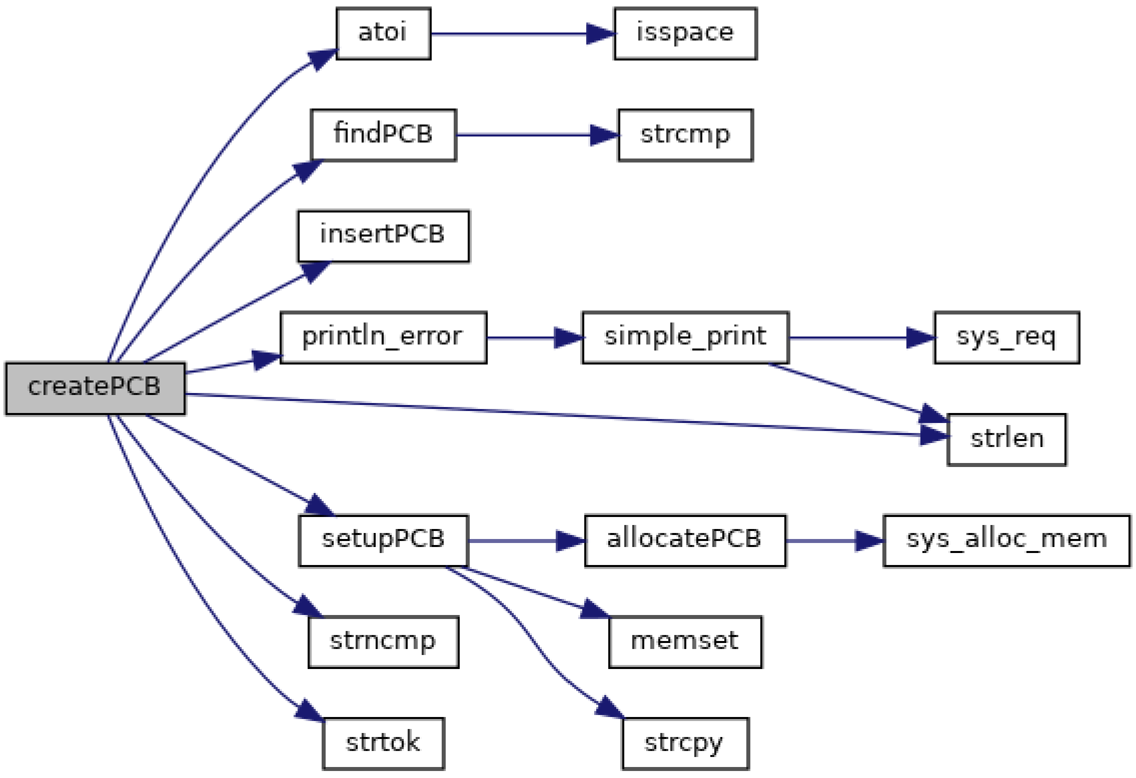
Creates PCB and inserts into the appropriate queue

Parameters

char	*params

Here is the call graph for this function:

IMAGE



void deletePCB (char * ***name*****)**

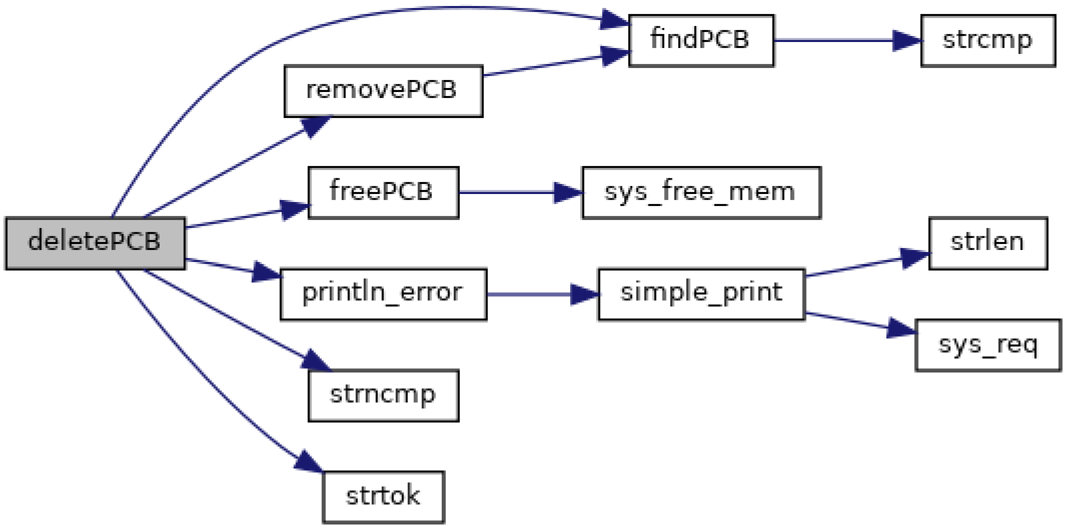
Removes PCB from appropriate queue and frees all associated memory

Parameters

char	*name

Here is the call graph for this function:

IMAGE



void unblockPCB (char * *name*)

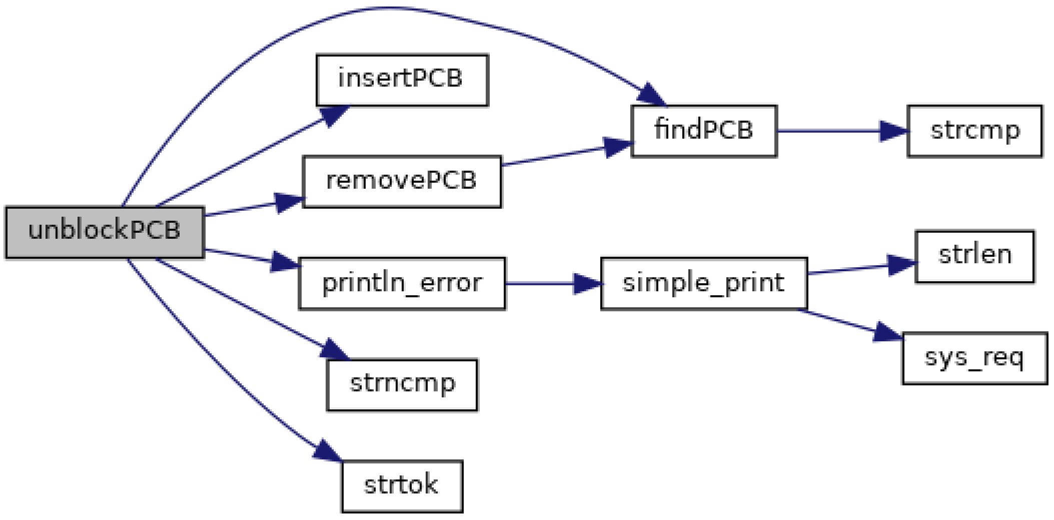
Makes PCB into the unblocked state and reinserts into the appropriate queue

Parameters

char	*name

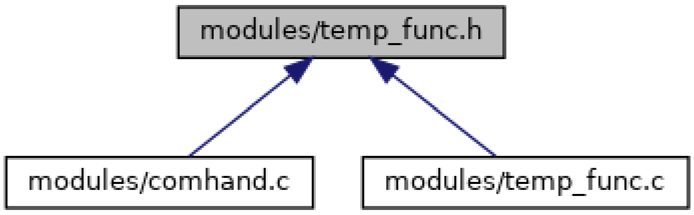
Here is the call graph for this function:

IMAGE



modules/temp_func.h File Reference

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

IMAGE 

Functions

void **createPCB** (char ***params**)
void **deletePCB** (char *name)
void **blockPCB** (char *name)
void **unblockPCB** (char *name)

Detailed Description

Function definitions for temporary commands R2

Function Documentation

void blockPCB (char * *name*)

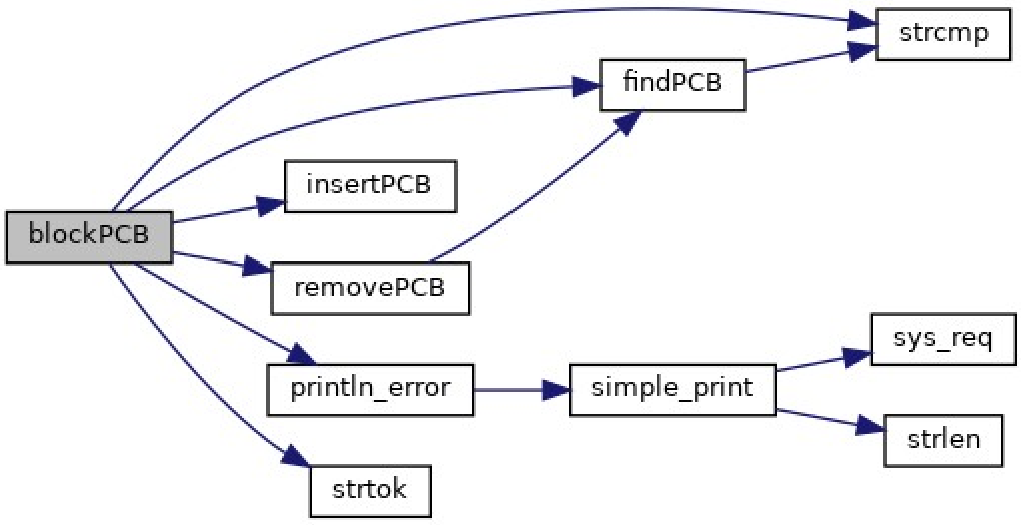
Finds PCB and sets its stae to blocked and reinserts into the appropriate queue

Parameters

char	*name

Here is the call graph for this function:

IMAGE



void createPCB (char * *params*)

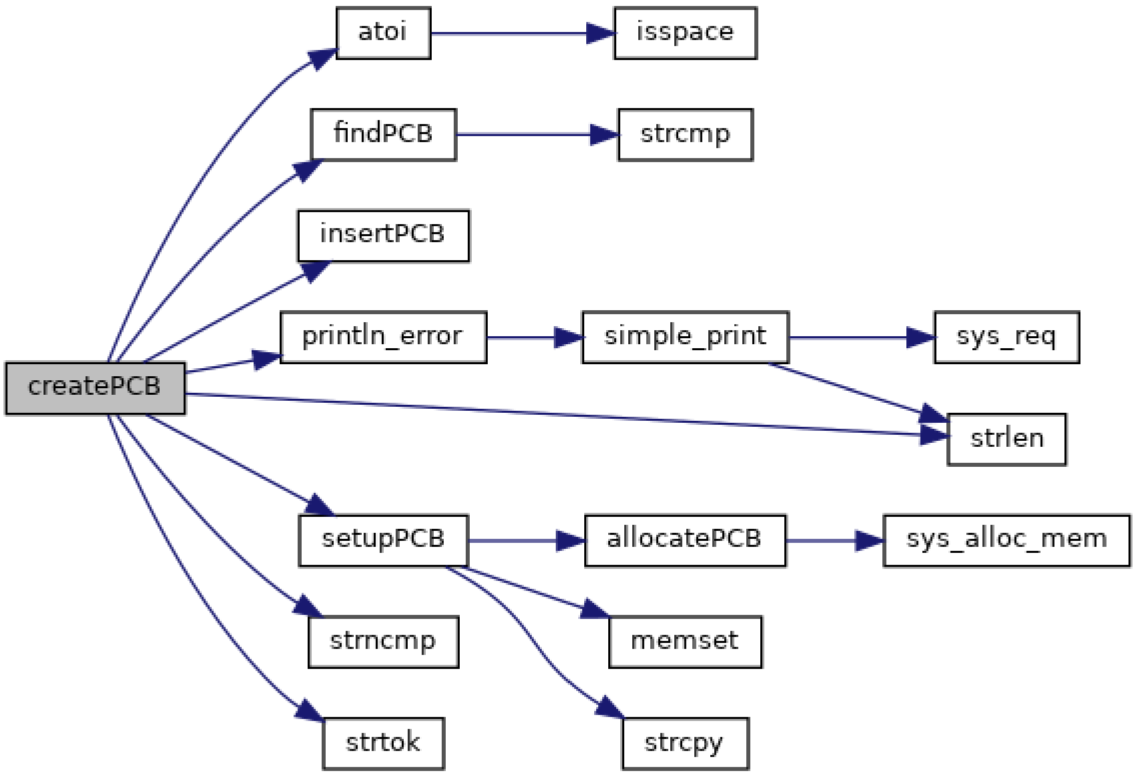
Creates PCB and inserts into the appropriate queue

Parameters

char	*params

Here is the call graph for this function:

IMAGE



void deletePCB (char * *name*)

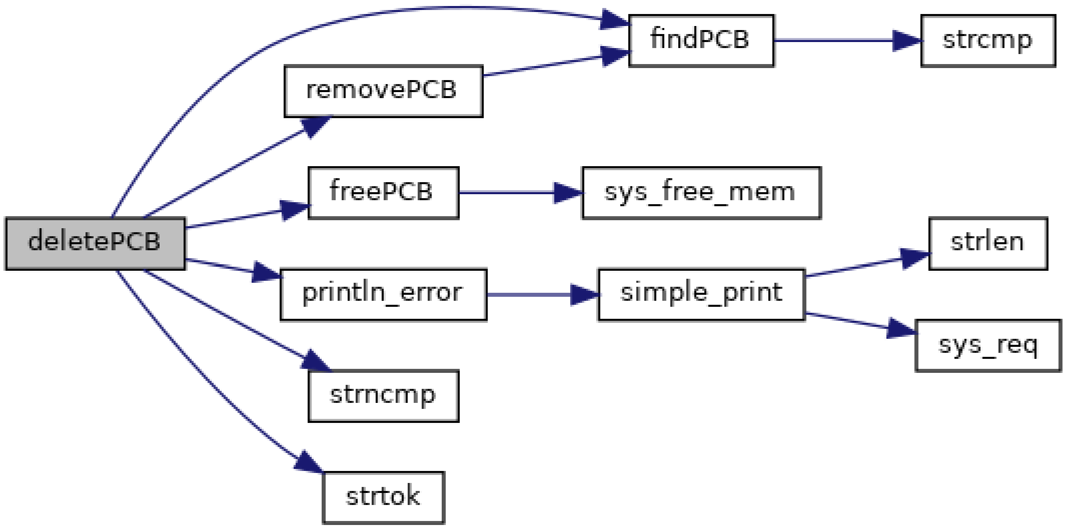
Removes PCB from appropriate queue and frees all associated memory

Parameters

char	*name

Here is the call graph for this function:

IMAGE



void unblockPCB (char * *name*)

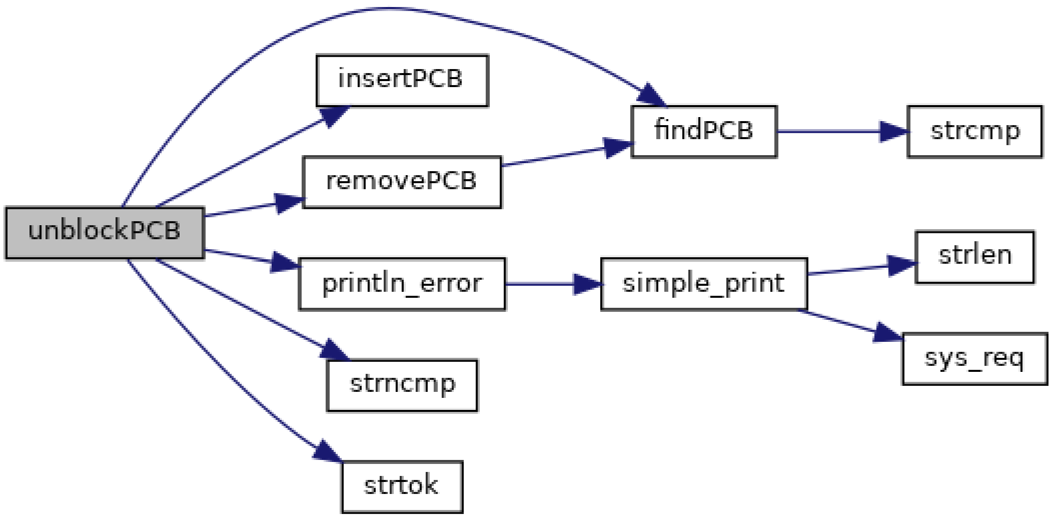
Makes PCB into the unblocked state and reinserts into the appropriate queue

Parameters

char	*name

Here is the call graph for this function:

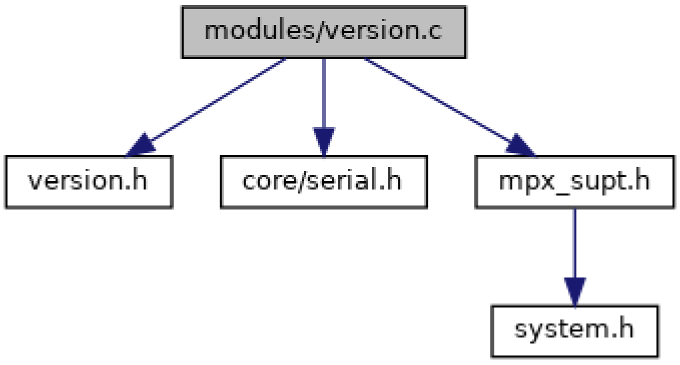
IMAGE


unblockPCB
insertPCB
removePCB
findPCB
strcmp
println_error
simple_print
strlen
sys_req
strncmp
strtok

modules/version.c File Reference

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

Include dependency graph for version.c:



IMAGE

Functions

int **version** ()

Detailed Description

Displays the version number of the mpx_core

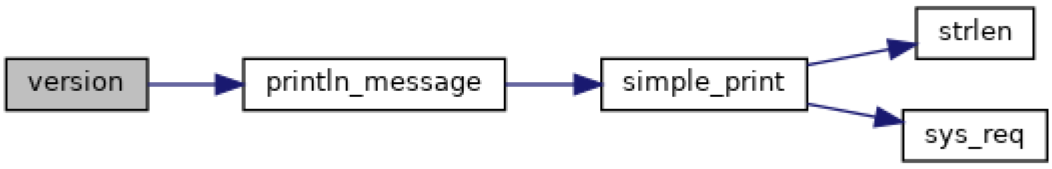
Function Documentation

int version ()

Helps display the version number of the current system.

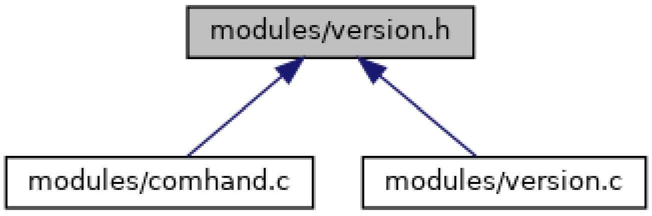
Here is the call graph for this function:

IMAGE



modules/version.h File Reference

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



IMAGE

Macros

#define **VERSION** "Version R2"

Functions

int **version** ()

Macro Definition Documentation

#define VERSION "Version R2"

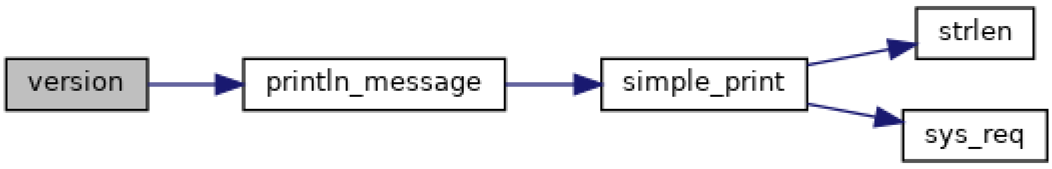
Function Documentation

int version ()

Helps display the version number of the current system.

Here is the call graph for this function:

IMAGE



Index

INDEX