Runtime Terror

Generated by Doxygen 1.9.1

1	Class Index	1
	1.1 Class List	1
2	File Index	3
	2.1 File List	3
3	Class Documentation	5
	3.1 date_time Struct Reference	5
	3.1.1 Detailed Description	5
	3.2 footer Struct Reference	5
	3.2.1 Detailed Description	5
	3.3 gdt_descriptor_struct Struct Reference	6
	3.3.1 Detailed Description	6
	3.4 gdt_entry_struct Struct Reference	6
	3.4.1 Detailed Description	6
	3.5 header Struct Reference	6
	3.5.1 Detailed Description	7
	3.6 heap Struct Reference	7
	3.6.1 Detailed Description	7
	3.7 idt_entry_struct Struct Reference	7
	3.7.1 Detailed Description	7
	3.8 idt_struct Struct Reference	8
	3.8.1 Detailed Description	8
	3.9 index_entry Struct Reference	8
	3.9.1 Detailed Description	8
	3.10 index_table Struct Reference	8
	3.10.1 Detailed Description	8
	3.11 page_dir Struct Reference	9
	3.11.1 Detailed Description	9
	3.12 page_entry Struct Reference	9
	3.12.1 Detailed Description	9
	3.13 page_table Struct Reference	9
	3.13.1 Detailed Description	10
	3.14 param Struct Reference	10
	3.14.1 Detailed Description	10
	3.15 PCB Struct Reference	10
	3.15.1 Detailed Description	10
	3.16 Queue Struct Reference	11
	3.16.1 Detailed Description	11
	onton Detailed Description	. 1
4	File Documentation	13
	4.1 include/core/asm.h File Reference	13
	4.2 include/core/interrupts.h File Reference	13

4.3 include/core/io.h File Reference	13
4.3.1 Macro Definition Documentation	13
4.3.1.1 inb	13
4.4 include/core/serial.h File Reference	14
4.5 include/core/tables.h File Reference	14
4.6 include/mem/heap.h File Reference	15
4.7 include/mem/paging.h File Reference	15
4.8 include/string.h File Reference	16
4.8.1 Function Documentation	16
4.8.1.1 atoi()	16
4.8.1.2 isspace()	17
4.8.1.3 memset()	17
4.8.1.4 strcat()	17
4.8.1.5 strcmp()	18
4.8.1.6 strcpy()	18
4.8.1.7 strlen()	19
4.8.1.8 strtok()	19
4.9 include/system.h File Reference	20
4.10 kernel/core/interrupts.c File Reference	21
4.11 kernel/core/kmain.c File Reference	22
4.12 kernel/core/serial.c File Reference	22
4.13 kernel/core/system.c File Reference	23
4.14 kernel/core/tables.c File Reference	23
4.15 kernel/mem/heap.c File Reference	23
4.16 kernel/mem/paging.c File Reference	24
4.17 lib/string.c File Reference	25
4.17.1 Function Documentation	25
4.17.1.1 atoi()	25
4.17.1.2 isspace()	26
4.17.1.3 memset()	26
4.17.1.4 strcat()	26
4.17.1.5 strcmp()	27
4.17.1.6 strcpy()	27
4.17.1.7 strlen()	28
4.17.1.8 strtok()	28
4.18 modules/mpx_supt.c File Reference	29
4.19 modules/mpx_supt.h File Reference	30
4.20 modules/R1/comHand.h File Reference	30
4.20.1 Function Documentation	31
4.20.1.1 comHand()	31
4.21 modules/R1/userFunctions.c File Reference	33
4.21.1 Function Documentation	34

	4.21.1.1 BCDtoDec()	34
	4.21.1.2 Block()	34
	4.21.1.3 Create_PCB()	35
	4.21.1.4 DectoBCD()	36
	4.21.1.5 Delete_PCB()	36
	4.21.1.6 EdgeCase()	36
	4.21.1.7 GetDate()	37
	4.21.1.8 GetTime()	38
	4.21.1.9 Help()	38
	4.21.1.10 itoa()	40
	4.21.1.11 Resume()	40
	4.21.1.12 Set_Priority()	41
	4.21.1.13 SetDate()	41
	4.21.1.14 SetTime()	42
	4.21.1.15 Show_All()	43
	4.21.1.16 Show_Blocked()	43
	4.21.1.17 Show_PCB()	45
	4.21.1.18 Show_Ready()	46
	4.21.1.19 Suspend()	48
	4.21.1.20 toLowercase()	48
	4.21.1.21 Unblock()	49
	4.21.1.22 Version()	49
4.22 modules	/R1/userFunctions.h File Reference	50
4.22.1 F	Function Documentation	50
	4.22.1.1 BCDtoDec()	50
	4.22.1.2 Block()	51
	4.22.1.3 Create_PCB()	51
	4.22.1.4 DectoBCD()	52
	4.22.1.5 Delete_PCB()	52
	4.22.1.6 EdgeCase()	53
	4.22.1.7 GetDate()	53
	4.22.1.8 GetTime()	54
	4.22.1.9 Help()	54
	4.22.1.10 itoa()	56
	4.22.1.11 Resume()	56
	4.22.1.12 Set_Priority()	57
	4.22.1.13 SetDate()	57
	4.22.1.14 SetTime()	58
	4.22.1.15 Show_All()	59
	4.22.1.16 Show_Blocked()	60
	4.22.1.17 Show_PCB()	61
	4.22.1.18 Show_Ready()	62

Index		67
	4.22.1.22 Version()	 66
	4.22.1.21 Unblock()	 66
	4.22.1.20 toLowercase()	 64
	4.22.1.19 Suspend()	 64

Chapter 1

Class Index

1.1 Class List

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

date_time	
footer	5
gdt_descriptor_struct	6
gdt_entry_struct	
header	
heap	
idt_entry_struct	7
idt_struct	8
index_entry	
index_table	
page_dir	
page_entry	
page_table	
param	10
PCB	10
Ougus	4.4

2 Class Index

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

include/string.h	16
include/system.h	20
include/core/asm.h	13
include/core/interrupts.h	13
include/core/io.h	13
include/core/serial.h	14
include/core/tables.h	14
include/mem/heap.h	15
include/mem/paging.h	15
kernel/core/interrupts.c	21
kernel/core/kmain.c	22
kernel/core/serial.c	22
kernel/core/system.c	23
kernel/core/tables.c	23
kernel/mem/heap.c	23
kernel/mem/paging.c	24
lib/string.c	25
modules/mpx_supt.c	29
modules/mpx_supt.h	30
modules/R1/comHand.c	??
modules/R1/comHand.h	30
modules/R1/userFunctions.c	33
modules/R1/userFunctions.h	50
modules/R2/ PCB.c	??
mandula a /DO/DOD h	

File Index

Chapter 3

Class Documentation

3.1 date_time Struct Reference

Public Attributes

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

3.1.1 Detailed Description

Definition at line 32 of file system.h.

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

• include/system.h

3.2 footer Struct Reference

Public Attributes

· header head

3.2.1 Detailed Description

Definition at line 18 of file heap.h.

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

• include/mem/heap.h

6 Class Documentation

3.3 gdt_descriptor_struct Struct Reference

Public Attributes

- u16int limit
- u32int base

3.3.1 Detailed Description

Definition at line 25 of file tables.h.

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

• include/core/tables.h

3.4 gdt_entry_struct Struct Reference

Public Attributes

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

3.4.1 Detailed Description

Definition at line 32 of file tables.h.

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

• include/core/tables.h

3.5 header Struct Reference

Public Attributes

- int size
- int index_id

3.5.1 Detailed Description

Definition at line 13 of file heap.h.

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

• include/mem/heap.h

3.6 heap Struct Reference

Public Attributes

- index_table index
- u32int base
- u32int max size
- u32int min_size

3.6.1 Detailed Description

Definition at line 35 of file heap.h.

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

• include/mem/heap.h

3.7 idt_entry_struct Struct Reference

Public Attributes

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

3.7.1 Detailed Description

Definition at line 8 of file tables.h.

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

· include/core/tables.h

8 Class Documentation

3.8 idt struct Struct Reference

Public Attributes

- u16int limit
- u32int base

3.8.1 Detailed Description

Definition at line 18 of file tables.h.

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

· include/core/tables.h

3.9 index_entry Struct Reference

Public Attributes

- int size
- · int empty
- u32int block

3.9.1 Detailed Description

Definition at line 22 of file heap.h.

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

· include/mem/heap.h

3.10 index_table Struct Reference

Public Attributes

- index entry table [TABLE SIZE]
- int id

3.10.1 Detailed Description

Definition at line 29 of file heap.h.

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

• include/mem/heap.h

3.11 page_dir Struct Reference

Public Attributes

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

3.11.1 Detailed Description

Definition at line 36 of file paging.h.

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

• include/mem/paging.h

3.12 page_entry Struct Reference

Public Attributes

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

3.12.1 Detailed Description

Definition at line 14 of file paging.h.

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

· include/mem/paging.h

3.13 page_table Struct Reference

Public Attributes

• page_entry pages [1024]

10 Class Documentation

3.13.1 Detailed Description

Definition at line 28 of file paging.h.

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

· include/mem/paging.h

3.14 param Struct Reference

Public Attributes

- int op code
- · int device_id
- char * buffer ptr
- int * count_ptr

3.14.1 Detailed Description

Definition at line 33 of file mpx_supt.h.

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

· modules/mpx_supt.h

3.15 PCB Struct Reference

Public Attributes

- unsigned char stack [MEM1K]
- unsigned char * stackTop
- struct PCB * prev
- struct PCB * next
- char Process_Name [10]
- · int Process_Class
- int **Priority**
- · int ReadyState
- · int SuspendedState

3.15.1 Detailed Description

Definition at line 14 of file PCB.h.

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

· modules/R2/PCB.h

3.16 Queue Struct Reference

Public Attributes

- int count
- PCB * head
- PCB * tail

3.16.1 Detailed Description

Definition at line 26 of file PCB.h.

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

· modules/R2/PCB.h

12 Class Documentation

Chapter 4

File Documentation

4.1 include/core/asm.h File Reference

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

4.2 include/core/interrupts.h File Reference

Functions

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

4.3 include/core/io.h File Reference

Macros

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

4.3.1 Macro Definition Documentation

4.3.1.1 inb

Definition at line 17 of file io.h.

4.4 include/core/serial.h File Reference

Macros

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

Functions

- int init_serial (int device)
- int serial_println (const char *msg)
- int serial_print (const char *msg)
- int set_serial_out (int device)
- int **set_serial_in** (int device)
- int * polling (char *buffer, int *count)

4.5 include/core/tables.h File Reference

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

Classes

- struct idt_entry_struct
- struct idt struct
- struct gdt_descriptor_struct
- struct gdt_entry_struct

Functions

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

Variables

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

4.6 include/mem/heap.h File Reference

Classes

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

Macros

- #define TABLE SIZE 0x1000
- #define KHEAP_BASE 0xD000000
- #define KHEAP_MIN 0x10000
- #define KHEAP_SIZE 0x1000000

Functions

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

4.7 include/mem/paging.h File Reference

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

Classes

- struct page_entry
- · struct page table
- struct page_dir

Macros

• #define PAGE_SIZE 0x1000

Functions

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

4.8 include/string.h File Reference

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

Functions

```
• int isspace (const char *c)
```

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

- char * strcpy (char *s1, const char *s2)
- char * strcat (char *s1, const char *s2)
- int strlen (const char *s)
- int strcmp (const char *s1, const char *s2)
- char * strtok (char *s1, const char *s2)
- int atoi (const char *s)

4.8.1 Function Documentation

4.8.1.1 atoi()

```
int atoi ( {\rm const\ char\ *\ s\ )}
```

Description: Convert an ASCII string to an integer

Parameters

```
s String
```

Definition at line 50 of file string.c.

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

4.8.1.2 isspace()

```
int isspace ( {\tt const\ char\ *\ c\ )}
```

Description: Determine if a character is whitespace.

Parameters

```
c character to check
```

Definition at line 121 of file string.c.

4.8.1.3 memset()

```
void* memset ( \label{eq:void*} \mbox{void} * s, \\ \mbox{int } c, \\ \mbox{size\_t } n \mbox{)}
```

Description: Set a region of memory.

Parameters

s	destination
С	byte to write
n	count

Definition at line 139 of file string.c.

4.8.1.4 strcat()

Description: Concatenate the contents of one string onto another.

Parameters

s1	destination
s2	source

Definition at line 108 of file string.c.

```
109 {
110    char *rc = s1;
111    if (*s1) while(*++s1);
112    while( (*s1++ = *s2++) );
113    return rc;
114 }
```

4.8.1.5 strcmp()

```
int strcmp (  \mbox{const char} \ * \ s1, \\ \mbox{const char} \ * \ s2 \ )
```

Description: String comparison

Parameters

s1	string 1
s2	string 2

Definition at line 81 of file string.c.

4.8.1.6 strcpy()

```
char* strcpy (  \mbox{char} * s1, \\ \mbox{const char} * s2 )
```

Description: Copy one string to another.

Parameters

s1	destination
s2	source

Definition at line 38 of file string.c.

```
39 {
40    char *rc = s1;
41    while( (*s1++ = *s2++) );
42    return rc; // return pointer to destination string
43 }
```

4.8.1.7 strlen()

```
int strlen ( {\rm const\ char\ *\ s\ )}
```

Description: Returns the length of a string.

Parameters

```
s input string
```

Definition at line 26 of file string.c.

```
27 {
28   int r1 = 0;
29   if (*s) while(*s++) r1++;
30   return r1;//return length of string
31 }
```

4.8.1.8 strtok()

```
char* strtok ( \label{eq:char} \mbox{char} \ * \ s1 \mbox{,} \mbox{const} \ \mbox{char} \ * \ s2 \mbox{)}
```

Description: Split string into tokens

Parameters

s1	String
s2	delimiter

Definition at line 153 of file string.c.

```
154 {
       static char *tok_tmp = NULL;
const char *p = s2;
155
156
157
158
        //new string
159
        if (s1!=NULL) {
160
        tok\_tmp = s1;
161
        //old string cont'd
162
163
       else {
  if (tok_tmp==NULL) {
    return NULL;
164
165
166
167
         s1 = tok\_tmp;
168
       }
169
       //skip leading s2 characters
while ( *p && *s1 ) {
170
```

```
if (*s1==*p) {
         ++s1;
173
         p = s2;
continue;
174
175
176
177
         ++p;
178
179
180
      //no more to parse
181
      if (!*s1) {
        return (tok_tmp = NULL);
182
183
184
185
      //skip non-s2 characters
186
      tok\_tmp = s1;
      while (*tok_tmp) {
  p = s2;
  while (*p) {
    if (*tok_tmp==*p++) {
187
188
189
190
191
        *tok_tmp++ = ' \setminus 0';
192
        return s1;
193
           }
194
195
        ++tok_tmp;
196
197
198
      //end of string
199 tok_tmp = NULL;
200
      return s1;
201 }
```

4.9 include/system.h File Reference

Classes

· struct date_time

Macros

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

Typedefs

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

Functions

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

4.10 kernel/core/interrupts.c File Reference

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

Macros

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

Functions

- void divide_error ()
- void debug ()
- void **nmi** ()
- void breakpoint ()
- void overflow ()
- void bounds ()
- void invalid_op ()
- void device not available ()
- void double_fault ()
- void coprocessor_segment ()
- void invalid_tss ()
- void segment_not_present ()
- void stack_segment ()
- · void general protection ()
- void page fault ()
- void reserved ()
- void coprocessor ()
- void rtc_isr ()
- void **isr0** ()
- · void do_isr ()
- void init_irq (void)
- void init pic (void)
- void do divide error ()
- void do_debug ()
- void do_nmi ()
- void do breakpoint ()
- void do_overflow ()
- void do_bounds ()
- void do invalid op ()
- void do_device_not_available ()
- void do_double_fault ()
- · void do coprocessor segment ()
- void do invalid tss ()
- void do_segment_not_present ()
- void do_stack_segment ()
- void do_general_protection ()
- void do page fault ()
- void do_reserved ()
- void do_coprocessor ()

Variables

• idt_entry idt_entries [256]

4.11 kernel/core/kmain.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <system.h>
#include <core/io.h>
#include <core/serial.h>
#include <core/tables.h>
#include <core/interrupts.h>
#include <mem/heap.h>
#include <mem/paging.h>
#include <modules/mpx_supt.h>
#include "modules/R1/comHand.h"
```

Functions

void kmain (void)

4.12 kernel/core/serial.c File Reference

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

Macros

• #define NO_ERROR 0

Functions

- int init_serial (int device)
- int serial_println (const char *msg)
- int serial_print (const char *msg)
- int set_serial_out (int device)
- int set_serial_in (int device)
- int * polling (char *cmdBuffer, int *count)

Variables

- int serial_port_out = 0
- int serial_port_in = 0

4.13 kernel/core/system.c File Reference

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

Functions

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

4.14 kernel/core/tables.c File Reference

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

Functions

- void write_gdt_ptr (u32int, size_t)
- void write_idt_ptr (u32int)
- void idt_set_gate (u8int idx, u32int base, u16int sel, u8int flags)
- void init_idt ()
- void **gdt init entry** (int idx, u32int base, u32int limit, u8int access, u8int flags)
- void init gdt ()

Variables

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

4.15 kernel/mem/heap.c File Reference

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

Functions

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

Variables

```
heap * kheap = 0
heap * curr_heap = 0
page_dir * kdir
void * end
void _end
void _end
u32int phys alloc addr = (u32int)&end
```

4.16 kernel/mem/paging.c File Reference

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

Functions

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

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

void new_frame (page_entry *page)

4.17 lib/string.c File Reference

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

Functions

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

4.17.1 Function Documentation

4.17.1.1 atoi()

```
int atoi ( {\rm const\ char\ *\ s\ )}
```

Description: Convert an ASCII string to an integer

Parameters

```
s String
```

Definition at line 50 of file string.c.

```
51 {
      int charVal=0;
char sign = ' ';
53
54
       char c = *s;
55
56
      while(isspace(&c)){ ++s; c = *s;} // advance past whitespace
60
       if (*s == '-' || *s == '+') sign = *(s++); // save the sign
61
62
63
      while(*s != '\0') {
    charVal = *s - 48;
res = res * 10 + charVal;
66
67
68
69
       }
70
72
      if ( sign == '-') res=res * -1;
73
     return res; // return integer
74
75 }
```

4.17.1.2 isspace()

```
int isspace ( {\tt const\ char\ *\ c\ )}
```

Description: Determine if a character is whitespace.

Parameters

```
c character to check
```

Definition at line 121 of file string.c.

4.17.1.3 memset()

```
void* memset ( \label{eq:void*} \mbox{void} * s, \\ \mbox{int } c, \\ \mbox{size\_t } n \mbox{)}
```

Description: Set a region of memory.

Parameters

s	destination
С	byte to write
n	count

Definition at line 139 of file string.c.

```
140 {
141    unsigned char *p = (unsigned char *) s;
142    while (n--) {
143         *p++ = (unsigned char) c;
144    }
145    return s;
146 }
```

4.17.1.4 strcat()

```
char* strcat (
```

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

Description: Concatenate the contents of one string onto another.

Parameters

s1	destination
s2	source

Definition at line 108 of file string.c.

```
100 {
110 char *rc = s1;
111 if (*s1) while(*++s1);
112 while( (*s1++ = *s2++) );
113 return rc;
114 }
```

4.17.1.5 strcmp()

```
int strcmp (  {\rm const~char} \ * \ s1, \\ {\rm const~char} \ * \ s2 \ )
```

Description: String comparison

Parameters

s1	string 1
s2	string 2

Definition at line 81 of file string.c.

4.17.1.6 strcpy()

```
char* strcpy (  {\it char} \ * \ s1, \\ {\it const} \ {\it char} \ * \ s2 \ )
```

Description: Copy one string to another.

Parameters

s1	destination
s2	source

Definition at line 38 of file string.c.

```
39 {
40    char *rc = s1;
41    while( (*s1++ = *s2++) );
42    return rc; // return pointer to destination string
43 }
```

4.17.1.7 strlen()

```
int strlen ( {\rm const\ char\ *\ s\ )}
```

Description: Returns the length of a string.

Parameters

```
s input string
```

Definition at line 26 of file string.c.

```
27 {
28   int r1 = 0;
29   if (*s) while(*s++) r1++;
30   return r1;//return length of string
31 }
```

4.17.1.8 strtok()

Description: Split string into tokens

Parameters

s1	String
s2	delimiter

Definition at line 153 of file string.c.

```
154 {
155    static char *tok_tmp = NULL;
156    const char *p = s2;
157
158    //new string
159    if (s1!=NULL) {
160       tok_tmp = s1;
161    }
```

```
162
      //old string cont'd
163
      else {
       if (tok_tmp==NULL) {
164
165
         return NULL;
166
        s1 = tok_tmp;
167
168
169
170
       //skip leading s2 characters
       while ( *p && *s1 ) {
  if (*s1==*p) {
171
172
173
         ++s1;
       p = s2;
continue;
}
174
175
176
177
178
179
      //no more to parse
180
      if (!*s1) {
182
        return (tok_tmp = NULL);
183
184
      //skip non-s2 characters
185
186
      tok_tmp = s1;
while (*tok_tmp){
187
       p = s2;
while (*p){
188
189
           if (*tok_tmp==*p++) {
190
        *tok_tmp++ = '\0';
191
192
        return s1;
        .cu
}
}
193
194
195
         ++tok_tmp;
196 }
197
198 //end of string
199 tok_tmp = NULL;
200 return s1;
200 return s1;
201 }
```

4.18 modules/mpx_supt.c File Reference

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

Functions

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

Variables

- · param params
- int current_module = -1
- u32int(* student_malloc)(u32int)
- int(* student_free)(void *)

4.19 modules/mpx supt.h File Reference

#include <system.h>

Classes

struct param

Macros

- #define EXIT 0
- #define IDLE 1
- #define READ 2
- #define WRITE 3
- #define INVALID_OPERATION 4
- #define TRUE 1
- #define FALSE 0
- #define MODULE_R1 0
- #define MODULE R2 1
- #define MODULE_R3 2
- #define MODULE_R4 4
- #define MODULE R5 8
- #define **MODULE_F** 9
- #define IO_MODULE 10
- #define MEM_MODULE 11
- #define INVALID_BUFFER 1000
- #define INVALID_COUNT 2000
- #define **DEFAULT_DEVICE** 111
- #define COM_PORT 222

Functions

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

4.20 modules/R1/comHand.h File Reference

Functions

• int comHand ()

4.20.1 Function Documentation

4.20.1.1 comHand()

```
int comHand ( )
```

Description: Interprets user input to call the appropriate user functions.

Definition at line 22 of file comHand.c.

```
23
           Help("\0");
2.4
25
26
           char cmdBuffer[100];
           int bufferSize = 99;
28
           int quit = 0;
29
           int shutdown = 0;
30
           while(quit != 1)
31
               memset (cmdBuffer, '\0', 100);
32
             sys_req(READ, DEFAULT_DEVICE, cmdBuffer, &bufferSize);
char* FirstToken = strtok(cmdBuffer, "-");
33
34
           char* SecondToken = strtok(NULL, "-");
    char* ThirdToken = strtok(NULL, "-");
35
36
               char* FourthToken = strtok(NULL, "-");
char* FifthToken = strtok(NULL, "-");
if(shutdown == 0) {
37
38
40
41
                   R1 comHand
42
                        if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, NULL) == 0)
4.3
44
                            Help("\0");
45
46
                        //R1 Commands
47
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "version") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
48
                            Help("Version");
49
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "getDate") == 0 &&
50
       strcmp(ThirdToken, NULL) == 0) {
51
                            Help("GetDate");
52
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "setDate") == 0 &&
53
       strcmp(ThirdToken, NULL) == 0) {
                            Help("SetDate");
54
56
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "getTime") == 0 &&
       strcmp(ThirdToken, NULL) == 0) {
                            Help("GetTime");
57
58
59
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "setTime") == 0 &&
       strcmp(ThirdToken, NULL) == 0) {
60
                            Help("SetTime");
61
62
                        // R2 Commands
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "suspend") == 0 &&
63
       strcmp(ThirdToken, NULL) == 0) {
64
                            Help("suspend");
65
66
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "resume") == 0 &&
       strcmp(ThirdToken, NULL) == 0) {
                            Help("resume");
67
68
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "setPriority") == 0 &&
69
       strcmp(ThirdToken, NULL) == 0)
70
                            Help("setPriority");
71
       72
73
                            Help("showPCB");
74
75
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "showAll") == 0 &&
       strcmp(ThirdToken, NULL) == 0) {
                            Help("showAll");
76
78
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "showReady") == 0 &&
       strcmp(ThirdToken, NULL) == 0)
```

```
79
                            Help("showReady");
80
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "showBlocked") == 0 &&
81
       strcmp(ThirdToken, NULL) == 0) {
                            Help("showBlocked");
82
83
84
                        // Temporary R2 commands
85
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "createPCB") == 0 &&
       strcmp(ThirdToken, NULL) == 0)
86
                            Help("createPCB");
                        }
87
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "deletePCB") == 0 &&
88
       strcmp(ThirdToken, NULL) == 0)
89
                            Help("deletePCB");
90
91
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "block") == 0 &&
       strcmp(ThirdToken, NULL) == 0)
92
                            Help("block");
93
94
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "unblock") == 0 &&
       strcmp(ThirdToken, NULL) == 0) {
                            Help("unblock");
95
96
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "shutdown") == 0 &&
97
       strcmp(ThirdToken, NULL) == 0)
98
                           Help("shutdown");
99
100
                         else if(strcmp(FirstToken, "version") == 0 && strcmp(SecondToken, NULL) == 0)
101
102
                             Version();
103
104
                         else if(strcmp(FirstToken, "getDate") == 0 && strcmp(SecondToken, NULL) == 0)
105
                             GetDate();
106
107
                         else if(strcmp(FirstToken, "setDate") == 0){
                             if (EdgeCase(SecondToken) == 1 && EdgeCase(ThirdToken) == 1 &&
108
       EdgeCase(FourthToken) == 1 && EdgeCase(FifthToken) == 1)
                                 SetDate(atoi(SecondToken), atoi(ThirdToken), atoi(FourthToken),
109
       atoi(FifthToken));
110
111
                             else
                                 printf("\x1b[31m""\nERROR: Invalid parameters for setDate \n""\x1b[0m");
112
113
114
                         else if(strcmp(FirstToken, "getTime") == 0 && strcmp(SecondToken, NULL) == 0) //Return
       the current time held by the registers.
115
                             GetTime();
116
                         else if(strcmp(FirstToken, "setTime") == 0 && strcmp(FifthToken, NULL) == 0){
117
                             if (EdgeCase(SecondToken) == 1 && EdgeCase(ThirdToken) == 1 &&
       EdgeCase(FourthToken) == 1)
118
                                      SetTime(atoi(SecondToken), atoi(ThirdToken), atoi(FourthToken));
       //input as Hour-Minute-Seconds
119
120
121
                                 printf("\times1b[31m""\setminusnERROR: Invalid parameters for setTime \setminusn""\setminusx1b[0m");
122
123
124
125
126
127
128
129
130
                        R2 comHand
131
132
                         else if(strcmp(FirstToken, "suspend") == 0 && strcmp(ThirdToken, NULL) == 0 &&
       strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
133
                             Suspend (SecondToken);
134
                         else if(strcmp(FirstToken, "resume") == 0 && strcmp(ThirdToken, NULL) == 0 &&
135
       strcmp(FourthToken, NULL) == 0 && strcmp(FifthToken, NULL) == 0) {
136
                            Resume (SecondToken);
137
138
                         else if(strcmp(FirstToken, "setPriority") == 0 && strcmp(FourthToken, NULL) == 0 &&
       strcmp(FifthToken, NULL) == 0) {
139
                             if (EdgeCase (ThirdToken) == 1)
                                 Set_Priority(SecondToken, atoi(ThirdToken)); //input as
140
       setPriority-Process_Name-Priority
141
142
                             else
                                 printf("\x1b[31m""\nERROR: Invalid parameters for setPriority, priority must
143
       be entered as a integer. n"" \times 1b[0m");
144
                         else if(strcmp(FirstToken, "showPCB") == 0 && strcmp(ThirdToken, NULL) == 0 &&
145
       strcmp (FourthToken, NULL) == 0 && strcmp (FifthToken, NULL) == 0) {
146
                             Show_PCB(SecondToken);
                             printf("\n");
147
148
                         }
```

```
149
                            if(strcmp(FirstToken, "showAll") == 0 && strcmp(SecondToken, NULL) == 0 &&
       strcmp(ThirdToken, NULL) == 0 && strcmp(FourthToken, NULL) == 0 && strcmp(FifthToken, NULL) == 0) {
150
                           Show_All();
151
                           printf("\n");
152
                       else if (strcmp (FirstToken, "showReady") == 0 && strcmp (SecondToken, NULL) == 0 &&
153
       strcmp(ThirdToken, NULL) == 0 && strcmp(FourthToken, NULL) == 0 && strcmp(FifthToken, NULL) == 0) {
154
                           Show_Ready();
155
                           printf("\n");
156
                       else if(strcmp(FirstToken, "showBlocked") == 0 && strcmp(SecondToken, NULL) == 0 &&
157
       strcmp(ThirdToken, NULL) == 0 && strcmp(FourthToken, NULL) == 0 && strcmp(FifthToken, NULL) == 0) {
                           Show_Blocked();
158
                           printf("\n");
159
160
161
                       /****** R2 Temp Commands *******/
162
                       else if(strcmp(FirstToken, "createPCB") == 0) {
163
164
                           if( strlen(SecondToken) < 11) {</pre>
                                   Create_PCB(SecondToken, atoi(ThirdToken), atoi(FourthToken));
165
       //input as Process_Name-Priority-Class
166
167
                           else
       printf("\x1b[31m""\nERROR: Invalid parameters for createPCB, Process\_name must only contain 10 or fewer characters. <math display="block">"""\x1b[0m"];
168
169
170
                       else if(strcmp(FirstToken, "deletePCB") == 0 && strcmp(ThirdToken, NULL) == 0 &&
       strcmp(FourthToken, NULL) == 0 && strcmp(FifthToken, NULL) == 0) {
171
                           Delete_PCB(SecondToken);
172
173
                       else if(strcmp(FirstToken, "block") == 0 && strcmp(ThirdToken, NULL) == 0 &&
       strcmp(FourthToken, NULL) == 0 && strcmp(FifthToken, NULL) == 0) {
174
                          Block (SecondToken);
175
       else if(strcmp(FirstToken, "unblock") == 0 && strcmp(ThirdToken, NULL) == 0 &&
strcmp(FourthToken, NULL) == 0 && strcmp(FifthToken, NULL) == 0) {
176
177
                           Unblock (SecondToken);
178
179
180
        shutdown comHand
181
182
        else if(strcmp(FirstToken, "shutdown") == 0 && strcmp(SecondToken, NULL) == 0){
183
                           printf("\x1b[33m""\nAre you sure you want to shutdown? [yes/no]\n""\x1b[0m");
184
185
                           shutdown = 1;
186
187
                           printf("\x1b[31m""\nERROR: Not a valid command \n""\x1b[0m");
188
189
190
191
192
                       if(strcmp(FirstToken, "yes") == 0 && shutdown == 1)
193
                           quit = 1;
194
                       else if(strcmp(FirstToken, "no") == 0){
195
                           printf("\x1b[33m""\nShutdown Cancelled\x1b[0m \n");
196
197
                           shutdown = 0;
198
199
                           200
201
                   }
202
203
           return 0;
                       //shutdown procedure
```

4.21 modules/R1/userFunctions.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include <core/io.h>
#include "../mpx_supt.h"
#include "../R2/PCB.h"
#include "userFunctions.h"
```

Functions

- char * itoa (int num)
- int BCDtoDec (int BCD)
- int DectoBCD (int Decimal)
- void **printf** (char msg[])
- int EdgeCase (char *pointer)
- void SetTime (int hours, int minutes, int seconds)
- void GetTime ()
- void SetDate (int day, int month, int millennium, int year)
- · void GetDate ()
- void Version ()
- char toLowercase (char c)
- void Help (char *request)
- void Suspend (char *ProcessName)
- void Resume (char *ProcessName)
- void Set_Priority (char *ProcessName, int Priority)
- void Show_PCB (char *ProcessName)
- void Show_All ()
- void Show_Ready ()
- void Show_Blocked ()
- void Create_PCB (char *ProcessName, int Priority, int Class)
- void Delete_PCB (char *ProcessName)
- void Block (char *ProcessName)
- void Unblock (char *ProcessName)

4.21.1 Function Documentation

4.21.1.1 BCDtoDec()

```
int BCDtoDec ( \label{eq:bcd} \text{int } \textit{BCD} \ )
```

Description: Changes binary number to decimal numbers.

Parameters

value Binary number to be changed to decimal

Definition at line 69 of file userFunctions.c.

4.21.1.2 Block()

Brief Description: Places a PCD in the blocked state and reinserts it into the correct queue.

Description: Can except a string as a pointer that is the Process Name. The specified PCB will be places in a blocked state and reinserted into the appropriate queue. An error check for a valid name occurs.

Parameters

Process_Name Character pointer that matches the name of	cess.
---	-------

Definition at line 858 of file userFunctions.c.

```
PCB* pcb = FindPCB(ProcessName);
if (pcb == NULL) {
859
860
861
        printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
862
         if(pcb->ReadyState == BLOCKED) {
    printf("\x1b[32m""\nThis Process is already BLOCKED \n""\x1b[0m");
864
865
866
867
         RemovePCB(pcb);
868
869
           pcb->ReadyState = BLOCKED;
870
            InsertPCB(pcb);
871
872
      }
873 }
```

4.21.1.3 Create_PCB()

Brief Description: Calls SetupPCB() and inserts PCB into appropriate queue.

Description: Can except a string as a pointer that is the Process Name. Can accept two integers, Priority and Class. SetupPCB() will be called and the PCB will be inserted into the appropriate queue. An error check for unique and valid Process Name, an error check for valid process class, and an error check for process priority.

Parameters

Process_Name	
Priority	integer that matches the priority number.
Class	integer that matches the class number.

Definition at line 812 of file userFunctions.c.

```
812
813
      if (FindPCB(ProcessName) == NULL)
        if(Priority >= 0 && Priority < 10) {
  if(Class == 0 || Class == 1) {</pre>
814
815
             PCB* pcb = SetupPCB(ProcessName, Class, Priority);
816
817
             InsertPCB(pcb);
818
             printf("\x1b[31m""\\nERROR: Not a valid Class <math>\n""\\x1b[0m");
819
820
821
        } else{
          printf("\x1b[31m""\\nERROR: Not a valid Priority \n""\\x1b[0m");
822
823
824
        printf("\x1b[31m""\nERROR: This Process Name already exists \n""\x1b[0m");
825
```

```
826 }
827 }
```

4.21.1.4 DectoBCD()

Description: Changes decimal numbers to binary numbers.

Parameters

Decimal Decimal number to be changed to binary

Definition at line 76 of file userFunctions.c.

4.21.1.5 Delete_PCB()

Brief Description: Removes PCB from appropriate queue and frees all associated memory.

Description: Can except a string as a pointer that is the Process Name. Removes PCB from the appropriate queue and then frees all associated memory. An error check to make sure process name is valid.

Parameters

Process_Name Character pointer that matches the name of process.

Definition at line 838 of file userFunctions.c.

```
838
839
      PCB* pcb = FindPCB(ProcessName);
      if (pcb == NULL)
840
       .f (pcb == NULL) {
   printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
841
842
843
      else {
844
         RemovePCB(pcb);
845
          FreePCB(pcb);
      }
846
847 }
```

4.21.1.6 EdgeCase()

Description: Compares pointer char to validate if it is a number or not.

Parameters

Compares pointer char to validate if it is a number or not.

Definition at line 97 of file userFunctions.c.

```
98
       int valid = 0;
99
       if (strcmp(pointer, "00") == 0) {
100
         valid = 1;
         return valid;
101
102
       else if (strcmp(pointer, "0") == 0) {
103
        valid = 1;
104
105
         return valid;
106
107
        else
108
        int j;
109
            valid = 0;
110
            for(j = 0; j <= 99; j++) {
                if(strcmp(pointer,itoa(j)) == 0)
112
                    valid = 1;
113
            if(valid == 0) {
114
115
             return valid:
116
117
118
        return valid;
119
```

4.21.1.7 GetDate()

```
void GetDate ( )
```

Description: Returns the full date back to the user in decimal form.

No parameters.

Definition at line 254 of file userFunctions.c.

```
254
255
           int check = 2;
           outb(0x70,0x07);
256
257
            unsigned char day = BCDtoDec(inb(0x71));
258
             outb(0x70,0x08);
259
             unsigned char month = BCDtoDec(inb(0x71));
            outb(0x70,0x32);
unsigned char millennium = BCDtoDec(inb(0x71));
char msg[2] = "-";
260
261
262
             char msg3[10] = "Date: ";
264
            printf(msg3);
265
             sys_req(WRITE, COM1, itoa(day), &check);
266
            printf(msg);
sys_req(WRITE, COM1, itoa(month), &check);
2.67
268
             printf(msg);
             sys_req(WRITE, COM1, itoa(millennium), &check);
269
270
         outb (0x70, 0x09);
        if (BCDtoDec(inb(0x71)) == 0) {
   sys_req(WRITE, COM1, "00", &check);
271
272
273
274
             unsigned char year = BCDtoDec(inb(0x71));
276
             sys_req(WRITE, COM1, itoa(year), &check);
277
278
             printf("\n");
279
```

4.21.1.8 GetTime()

```
void GetTime ( )
```

Description: retrieve and return the time values for hours, minutes, and seconds form the clock register using inb(Port,address).

No parameters.

Definition at line 176 of file userFunctions.c.

```
177
        int check = 2;
178
        int hour;
179
        int minute:
180
        int second;
            outb(0x70,0x04);
181
182
            unsigned char hours = inb(0x71);
183
            outb(0x70,0x02);
184
            unsigned char minutes = inb(0x71);
185
            outb(0x70,0x00);
            unsigned char seconds = inb(0x71);
char msg1[2] = ":";
186
187
            char msg2[10] = "Time: ";
188
            printf(msg2);
190
            hour = BCDtoDec(hours);
191
            sys_req(WRITE, COM1, itoa(hour), &check);
192
            printf(msg1);
193
            minute = BCDtoDec(minutes);
194
            sys_req(WRITE, COM1, itoa(minute), &check);
195
            printf(msg1);
            second = BCDtoDec(seconds);
196
            sys_req(WRITE, COM1, itoa(second), &check);
197
198
          printf("\n");
199
```

4.21.1.9 Help()

Brief Description: Gives helpful information for one of the functions

Description: Can except a string as a pointer, if the pointer is null then the function will print a complete list of avaliable commands to the console. If the pointer is a avaliable commands then instructions on how to use the command will be printed. If the command does not exist then a message explaining that it is not a valid command will be displayed.

Parameters

request Character pointer that matches the name of the function that you need help with.

Definition at line 308 of file userFunctions.c.

```
if (request[0] == '\0')
309
310
                printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
311
             printf("\n getDate \n setDate \n getTime \n setTime \n version \n suspend \n resume \n
     unblock \n shutdown \n'n;
312
          else if (strcmp(request, "GetDate") == 0) {
314
                \verb|printf("\n getDate returns the current date that is loaded onto the operating|
     system.\n");
315
          else if (strcmp(request, "SetDate") == 0) {
316
```

```
317
                    printf("\n setDate allows the user to reset the correct date into the system, as follows
       setDate-"BLU"day"RESET"-"BLU"month"RESET"-"BLU"year"RESET".\n Time must be inputed as a two digit
       number, Example 02 or 00");
318
            else if (strcmp(request, "GetTime") == 0) {
319
                   printf("\n getTime returns the current time as hours, minutes, seconds that is loaded
320
       onto the operating system. \n");
321
            else if (strcmp(request, "SetTime") == 0) {
322
       printf("\n setTime allows the user to reset the correct time into the system, as follows setTime-"BLU"hour"RESET"-"BLU"minute"RESET"-"BLU"second"RESET".\n Time must be inputed as a two digit
323
       number, Example 02 or 00");
324
325
            else if (strcmp(request, "Version") == 0) {
326
                   printf("\n" version returns the current operating software version that the system is
       running.\n");
327
        else if(strcmp(request, "shutdown") == 0) {
328
         printf("\n shutdown shuts down the system.\n");
329
330
331
332
333
       334
                   R2 Commands
335
336
        else if(strcmp(request, "suspend") == 0) {
        printf("\n Suspend takes in the name of a PCB (suspend-NAME) then places it into the suspended state
337
       and reinserts it into the correct queue.\n");
338
339
             if (strcmp(request, "resume") == 0) {
        printf("\n Resume takes in the name of a PCB (resume-NAME) then removes it from the suspended state
340
       and adds it to the correct queue.\n");
341
        else if(strcmp(request, "setPriority") == 0) {
342
        printf("\n SetPriority takes in the name of a PCB and the priority (setPrioriry-NAME-PRIORITY) it
343
       needs to be set to then reinstates the specified PCB into a new location by priority.\n");
344
        else if(strcmp(request, "showPCB") == 0) {
345
346
        printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
       user.\n");
347
348
        else if(strcmp(request, "showAll") == 0) {
349
        printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
       queues.\n");
350
        else if(strcmp(request, "showReady") == 0) {
351
        printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
352
       currently are in the ready state.\n");
353
        else if(strcmp(request, "showBlocked") == 0) {
354
355
        \texttt{printf("} \\ \texttt{N} \\ \texttt{ ShowBlocked takes in no parameters but returns all PCB's and there attributes that}
       currently are in the blocked state.\n");
356
357
358 /******* R2 Temp Commands
       359
        else if(strcmp(request, "createPCB") == 0) {
360
        printf("\n CreatePCB takes in the process_name, process_class, and
       process_priority.(createPCB-NAME-PRIORITY-CLASS) Then assigns this new process into the correct
       queue.\n");
361
362
        else if(strcmp(request, "deletePCB") == 0) {
363
        printf("\n DeletePCB takes in the process_name (deletePCB-NAME) then deletes it from the queue and
       free's all the memory that was previously allocated to the specified PCB.\n");
364
365
        else if(strcmp(request, "block") == 0) {
        printf("\n Block takes in the process_name (block-NAME) then sets it's state to blocked and
366
       reinserts it back into the correct queue.\n");
367
368
        else if(strcmp(request, "unblock") == 0) {
369
        \texttt{printf("} \\ \texttt{N} \\ \texttt{Unblock takes in the process\_name (unblock-NAME)} \\ \texttt{then sets it's state to ready and} \\
       reinserts it back into the correct queue.\n");
370
371
         printf("\x1b[3lm""\nThe requested command does not exist please refer to the Help function for a
372
       full list of commands.\n""\x1b[0m");
373
374 }
```

4.21.1.10 itoa()

```
char* itoa (
                int num )
```

Description: An integer is taken and seperated into individual chars and then all placed into a character array. Adapted from geeksforgeeks.org.

Parameters

```
num integer to be put into array Title: itoa Author: Neha Mahajan Date: 29 May, 2017 Availability: https://www.geeksforgeeks.org/implement-itoa/
```

Definition at line 38 of file userFunctions.c.

```
40
                  int i, j, k, count;
41
                  i = num;
                  j = 0;
            count = 0;
while(i){ // count number of digits
43
44
45
                  count++;
                 i /= 10;
46
            }
             char* arr1;
49
50
             char arr2[count];
            arr1 = (char*)sys_alloc_mem(count); //memory allocation
51
52
53
             while(num){ // seperate last digit from number and add ASCII
                 arr2[++j] = num%10 + '0';
num /= 10;
55
56
57
             for (k = 0; k < j; k++) \{ // \text{ reverse array results } arr1[k] = arr2[j-k];
58
59
60
             arr1[k] = ' \setminus 0';
             return(char*)arr1;
63
64
```

4.21.1.11 Resume()

Brief Description: Places a PCD in the not suspended state and reinserts it into the appropriate queue.

Description: Can except a string as a pointer that is the Process Name. Places a PCB in the not suspended state and reinserts it into the appropriate queue. An error check for valid Process Name.

Parameters

Process_Name Character pointer that matches the name of process.

Definition at line 416 of file userFunctions.c.

```
416
417    PCB* pcb = FindPCB(ProcessName);
418    if (pcb == NULL) {
419        printf(RED"\nERROR: Not a valid process name \n"RESET);
420    }
```

4.21.1.12 Set Priority()

Brief Description: Sets PCB priority and reinserts the process into the correct place in the correct queue.

Description: Can except a string as a pointer that is the Process Name. Can accept and integer than is the Priority. Sets a PCB's priority and reinserts the process into the correct place in the correct queue. An error check for valid Process Name and an error check for a valid priority 1 - 9.

Parameters

Process_Name	Character pointer that matches the name of process.
Priority	integer that matches the priority number.

Definition at line 441 of file userFunctions.c.

```
442
          PCB* pcb = FindPCB(ProcessName);
443
          if (pcb == NULL)
            printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
444
445
          else if(Priority >= 10){
   printf("\xlb[31m""\nERROR: Not a valid Priority \n""\xlb[0m");
446
447
448
          else (
449
          RemovePCB(pcb);
450
451
                pcb->Priority = Priority;
                 InsertPCB(pcb);
453
454 }
```

4.21.1.13 SetDate()

Description: Sets the date register to the new values that the user inputed, all values must be inputed as Set

Dime(day, month, millenial, year).

Parameters

day	Integer to be set in the Day position
month	Integer to be set in the Month position
millenial	Integer to be set in the Millenial position
year	Integer to be set in the Year position

Definition at line 207 of file userFunctions.c.

```
207
208
        outb (0x70, 0x07);
        int tempDay = BCDtoDec(inb(0x71));
209
210
        outb (0x70, 0x08);
211
        int tempMonth = BCDtoDec(inb(0x71));
212
        outb(0x70,0x32);
213
        int tempMillennium = BCDtoDec(inb(0x71));
214
        outb (0x70, 0x09);
215
        int tempYear = BCDtoDec(inb(0x71));
216
        cli();
            outb(0x70,0x07);
218
            outb(0x71,DectoBCD (day));
219
            outb (0x70, 0x08);
            outb(0x71,DectoBCD (month));
outb(0x70,0x32);
220
221
222
            outb(0x71,DectoBCD (millennium));
223
            outb(0x70,0x09);
224
            outb(0x71, DectoBCD (year));
225
            sti();
226
        outb (0x70, 0x07);
227
        unsigned char newDay = BCDtoDec(inb(0x71));
228
        outb(0x70,0x08);
229
        unsigned char newMonth = BCDtoDec(inb(0x71));
230
        outb(0x70,0x32);
231
        unsigned char newMillennium = BCDtoDec(inb(0x71));
232
        outb (0x70, 0x09);
        unsigned char newYear = BCDtoDec(inb(0x71));
233
        if (newDay != day || newMonth != month || newMillennium != millennium || newYear != year) {
234
235
          printf("Your input was invalid\n");
237
            outb (0x70, 0x07);
            outb(0x71,DectoBCD (tempDay));
238
            outb(0x70,0x08);
outb(0x71,DectoBCD (tempMonth));
239
240
            outb(0x70,0x32);
241
242
            outb(0x71,DectoBCD (tempMillennium));
243
            outb(0x70,0x09);
244
            outb(0x71,DectoBCD (tempYear));
245
            sti();
246
247
        else
248
          printf("Date Set\n");
249
```

4.21.1.14 SetTime()

Description: sets the time register to the new values that the user inputed, all values must be inputed as SetTime(← Hours, Minutes, Seconds).

Parameters

hours	Integer to be set in the Hour position
minutes	Integer to be set in the Minutes position
seconds	Integer to be set in the Seconds position

Definition at line 137 of file userFunctions.c.

```
138
         outb (0x70, 0x04);
         unsigned char tempHours = BCDtoDec(inb(0x71));
139
140
         outb (0x70, 0x02);
141
         unsigned char tempMinutes = BCDtoDec(inb(0x71));
142
         outb(0x70,0x00);
143
         unsigned char tempSeconds = BCDtoDec(inb(0x71));
144
            cli(); //outb(device + 1, 0x00); //disable interrupts
145
             outb (0x70, 0x04);
             outb(0x71, DectoBCD(hours));// change to bcd
146
147
             outb (0x70, 0x02);
             outb(0x71, DectoBCD(minutes));
148
149
             outb(0x70,0x00);
             outb(0x71, DectoBCD(seconds));
sti(); //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
150
151
         outb (0x70.0x04):
152
         unsigned char newHours = BCDtoDec(inb(0x71));
153
154
         outb (0x70, 0x02);
155
         unsigned char newMinutes = BCDtoDec(inb(0x71));
156
         outb (0x70, 0x00);
157
         unsigned char newSeconds = BCDtoDec(inb(0x71));
         if(newHours != hours || newMinutes != minutes || newSeconds != seconds){
  printf("Your input was invalid\n");
  cli(); //outb(device + 1, 0x00); //disable interrupts
158
159
160
161
             outb(0x70,0x04);
162
             outb(0x71, DectoBCD(tempHours));// change to bcd
163
             outb(0x70,0x02);
             outb(0x71, DectoBCD(tempMinutes));
164
             outb(0x70,0x00);
165
             outb(0x71, DectoBCD(tempSeconds));
166
167
             sti(); //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
168
169
         else
        printf("Time Set\n");
}
170
```

4.21.1.15 Show_All()

```
void Show_All ( )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the ready and blocked queues.

Description: The process name, claas, state, suspend status, and priority of each of he PCB's in the ready and blocked queues.

Definition at line 535 of file userFunctions.c.

4.21.1.16 Show_Blocked()

```
void Show_Blocked ( )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the blocked queue.

Description: The process name, claas, state, suspend status, and priority of each of he PCB's in the blocked queue.

Definition at line 679 of file userFunctions.c.

```
679 {
680 if(getBlocked()->head == NULL) {
```

```
printf("\x1b[32m""\n The Blocked Queue is empty \n""\x1b[0m");
681
682
683
         else
                int class, check, state, prior, status;
char name[20];
char block[] = "\x1B[34m""Blocked Queue: \n""\x1b[0m";
char cname[] = "Name: ";
char cclass[] = "Class: ";
684
685
686
687
688
                char cstate[] = "State: ";
char cstatus[] = "Status: ";
689
690
                char cstatus[] = "status. ,
char cprior[] = "Priority: ";
char line[] = "\n";
691
692
693
                check = 15;
694
695
                 sys_req(WRITE, COM1, block, &check);
696
697
                PCB* pcb = getBlocked()->head;
698
                if(pcb->next == NULL) {
699
700
                   class = pcb->Process_Class;
701
                        strcpy(name,pcb->Process_Name);
                        state = pcb->ReadyState;
status = pcb->SuspendedState;
702
703
                        prior = pcb->Priority;
704
705
706
                       printf(cname);
                        printf(name);
707
708
                       printf(line);
709
710
                        printf(cclass);
711
                        if(pcb->Process_Class == 0) {
712
                         printf("0");
713
714
                        else {
715
716
                          sys_req(WRITE, COM1, itoa(class), &check);
717
                        printf(line);
718
719
                        printf(cstate);
720
                        if(pcb->ReadyState == 0) {
721
                          printf("0");
722
723
                        else
724
                          sys_req(WRITE, COM1, itoa(state), &check);
725
726
                        printf(line);
727
728
                        printf(cstatus);
                        if(pcb->SuspendedState == 0) {
  printf("0");
729
730
731
732
733
                          sys_req(WRITE, COM1, itoa(status), &check);
734
735
                        printf(line);
736
737
                        printf(cprior);
738
                        if (pcb->Priority == 0) {
                          printf("0");
printf("\n\n");
739
740
741
742
                        else {
743
                          sys_req(WRITE, COM1, itoa(prior), &check);
744
                          printf("\n\n");
745
746
747
                else {
                   while (pcb != NULL) {
748
749
                        class = pcb->Process_Class;
750
                            strcpy(name,pcb->Process_Name);
                             state = pcb->ReadyState;
status = pcb->SuspendedState;
751
752
                            prior = pcb->Priority;
753
754
755
                             printf(cname);
756
                            printf(name);
757
                            printf(line);
758
759
                             printf(cclass);
760
                             if (pcb->Process_Class == 0) {
                               printf("0");
761
762
763
                               sys_req(WRITE, COM1, itoa(class), &check);
764
765
766
                             printf(line);
767
```

```
768
                         printf(cstate);
769
                         if(pcb->ReadyState == 0) {
770
                           printf("0");
771
772
                         else (
                           sys_req(WRITE, COM1, itoa(state), &check);
773
774
775
                         printf(line);
776
777
                         printf(cstatus);
                         if(pcb->SuspendedState == 0) {
  printf("0");
778
779
780
781
782
                           sys_req(WRITE, COM1, itoa(status), &check);
783
784
                         printf(line);
785
786
                         printf(cprior);
787
                         if(pcb->Priority == 0) {
788
                           printf("0");
789
                           printf("\n\n");
790
791
                         else {
792
                           sys_reg(WRITE, COM1, itoa(prior), &check);
793
                           printf("\n\n");
794
795
                    pcb = pcb->next;
796
                }
797
             }
798
        }
799 }
```

4.21.1.17 Show PCB()

Brief Description: Displays the process name, class, state, suspended status, and priority of a PCB.

Description: Can except a string as a pointer that is the Process Name. The process name, claas, state, suspend status, and priority of a PCB are displayed. An error check for a valid name occurs.

Parameters

Process_Name | Character pointer that matches the name of process

Definition at line 464 of file userFunctions.c.

```
464
           if (FindPCB(ProcessName) == NULL)
465
466
                  printf("\x1b[31m""\nERROR: PCB does not exist \n""\x1b[0m");
467
468
           else
                   int check = 5;
469
                   char name[10];
470
                   char cname[] = "Name: ";
471
                   char cclass[] = "Class: ";
                   char cclass[] = "Class: ";
char cstate[] = "State: ";
char cstatus[] = "Status: ";
char cprior[] = "Priority: ";
char line[] = "\n";
PCB* pcb = FindPCB(ProcessName);
473
474
475
476
477
478
                   strcpy(name,pcb->Process_Name);
                   int class = pcb->Process_Class;
int state = pcb->ReadyState;
int status = pcb->SuspendedState;
479
480
481
                   int prior = pcb->Priority;
482
483
484
                    if(name == NULL){
485
                            printf("\x1b[31m""\nERROR: Not a valid process name <math>\n""\x1b[0m");
486
                    }
```

```
else {
488
                   printf(cname);
489
                printf(ProcessName);
490
                printf(line);
491
                printf(cclass);
492
                 if(pcb->Process_Class == 0) {
                   printf("0");
493
494
495
                 else {
                   sys_req(WRITE, COM1, itoa(class), &check);
496
                 }
497
                   printf(line);
498
499
                printf(cstate);
500
                 if (pcb->ReadyState == 0) {
501
                   printf("0");
502
503
                 else {
                   sys_req(WRITE, COM1, itoa(state), &check);
504
505
506
                   printf(line);
507
                printf(cstatus);
508
                 if(pcb->SuspendedState == 0) {
509
                   printf("0");
510
511
                 else
                   sys_req(WRITE, COM1, itoa(status), &check);
512
513
                   printf(line);
514
515
                printf(cprior);
                 if(pcb->Priority == 0) {
516
517
                   printf("0");
518
                   printf("\n\n");
519
520
                 else
521
                   sys_req(WRITE, COM1, itoa(prior), &check);
                   printf("\n\n");
522
523
524
           }
525
       }
526 }
```

4.21.1.18 Show_Ready()

```
void Show_Ready ( )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the ready queue.

Description: The process name, claas, state, suspend status, and priority of each of he PCB's in the ready queue.

Definition at line 547 of file userFunctions.c.

```
548
549
550
551
        else
552
553
554
              int class, check, state, prior, status;
              char name[10];
char ready[] = "\x1B[34m""\nReady Queue:\n""\x1B[0m";
char cname[] = "Name: ";
555
556
557
              char cclass[] = "Class: ";
char cstate[] = "State: ";
558
559
              char cstatus[] = "Status: ";
char cprior[] = "Priority: ";
560
561
               char line[] = "\n";
562
563
              check = 5;
564
565
              sys_req(WRITE, COM1, ready, &check);
566
567
              PCB* pcb = getReady()->head;
568
569
               if(pcb->next == NULL) {
570
                class = pcb->Process_Class;
                     strcpy(name,pcb->Process_Name);
```

```
state = pcb->ReadyState;
573
                      status = pcb->SuspendedState;
                      prior = pcb->Priority;
574
575
576
                      printf(cname);
577
                      printf(name);
578
                      printf(line);
579
580
                      printf(cclass);
                      if (pcb->Process_Class == 0) {
  printf("0");
581
582
583
584
                      else {
585
                       sys_req(WRITE, COM1, itoa(class), &check);
586
587
                      printf(line);
588
589
                      printf(cstate);
                      if (pcb->ReadyState == 0) {
590
591
                        printf("0");
592
593
                      else {
                        sys_req(WRITE, COM1, itoa(state), &check);
594
595
596
                     printf(line);
597
598
                      printf(cstatus);
599
                      if(pcb->SuspendedState == 0) {
600
                       printf("0");
601
602
                      else {
603
                        sys_req(WRITE, COM1, itoa(status), &check);
604
605
                      printf(line);
606
                      printf(cprior);
607
608
                      if (pcb->Priority == 0) {
                        printf("0");
609
610
                        printf("\n\n");
611
612
                      else {
                        sys_req(WRITE, COM1, itoa(prior), &check); printf("\n\");
613
614
615
616
617
               else {
618
                 while (pcb != NULL) {
                     class = pcb->Process_Class;
619
                          strcpy(name,pcb->Process_Name);
state = pcb->ReadyState;
status = pcb->SuspendedState;
620
621
622
623
                          prior = pcb->Priority;
624
625
                          printf(cname);
626
                          printf(name);
627
                          printf(line);
628
629
                          printf(cclass);
630
                          if(pcb->Process_Class == 0) {
631
                            printf("0");
632
633
                          else {
634
                            sys_req(WRITE, COM1, itoa(class), &check);
635
636
                          printf(line);
637
638
                          printf(cstate);
                          if(pcb->ReadyState == 0) {
639
                            printf("0");
640
641
642
                          else {
643
                            sys_req(WRITE, COM1, itoa(state), &check);
644
                          printf(line);
645
646
647
                          printf(cstatus);
648
                          if (pcb->SuspendedState == 0)
649
                            printf("0");
650
651
                          else
                            sys_req(WRITE, COM1, itoa(status), &check);
652
653
654
                          printf(line);
655
656
                          printf(cprior);
                          if (pcb->Priority == 0) {
  printf("0");
657
658
```

```
printf("\n\n");
660
661
                             sys_req(WRITE, COM1, itoa(prior), &check); printf("\n^n");
662
663
664
665
                      pcb = pcb->next;
666
                 }
667
668
        }
669
670 }
```

4.21.1.19 Suspend()

Brief Description: Places a PCD in the suspended state and reinserts it into the appropriate queue.

Description: Can except a string as a pointer that is the Process Name. Places a PCB in the suspended state and reinserts it into the appropriate queue. An error check for valid Process Name.

Parameters

Process_Name Character pointer that matches the name of process.

Definition at line 391 of file userFunctions.c.

```
391
392
          PCB* pcb = FindPCB(ProcessName);
393
           if (pcb == NULL)
394
            printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
395
396
             if(pcb->SuspendedState == YES) {
    printf("\x1b[32m""\nThis Process is already SUSPENDED \n""\x1b[0m");
397
398
399
400
           else
401
                pcb->SuspendedState = YES;
402
403
          }
404
405 }
```

4.21.1.20 toLowercase()

```
char toLowercase ( char c )
```

Description: If a letter is uppercase, it changes it to lowercase. (char)

Parameters

c Character that is to be changed to its lowercase equivalent

Definition at line 291 of file userFunctions.c.

4.21.1.21 Unblock()

Brief Description: Places a PCD in the unblocked state and reinserts it into the correct queue.

Description: Can except a string as a pointer that is the Process Name. The specified PCB will be places in an unblocked state and reinserted into the appropriate queue. An error check for a valid name occurs.

Parameters

Process_Name Character pointer that matches the name of process.

Definition at line 884 of file userFunctions.c.

```
884
885
      PCB* pcb = FindPCB(ProcessName);
      if (pcb == NULL) {
886
887
       printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
888
889
      if(pcb->ReadyState == READY) {
    printf("\x1b[32m""\nThis Process is already in the READY state \n""\x1b[0m");
890
891
892
       else
893
       RemovePCB (pcb);
894
895
          pcb->ReadyState = READY;
896
          InsertPCB(pcb);
897
898 }
899 }
```

4.21.1.22 Version()

```
void Version ( )
```

Description: Simply returns a char containing "Version: R(module).(the iteration that module is currently on).

No parameters.

Definition at line 284 of file userFunctions.c.

```
284 {
285 printf("Version: R2.6 \n");
286 }
```

4.22 modules/R1/userFunctions.h File Reference

Macros

```
    #define RED "\x1B[31m"
```

- #define GRN "\x1B[32m"
- #define YEL "\x1B[33m"
- #define **BLU** "\x1B[34m"
- #define MAG "\x1B[35m"
- #define CYN "\x1B[36m"
- #define WHT "\x1B[37m"
- #define RESET "\x1B[0m"

Functions

- void SetTime (int hours, int minutes, int seconds)
- void GetTime ()
- int DectoBCD (int Decimal)
- char * itoa (int num)
- void SetDate (int day, int month, int millennium, int year)
- int BCDtoDec (int BCD)
- void GetDate ()
- void Version ()
- void Help (char *request)
- void printf (char msg[])
- int EdgeCase (char *pointer)
- char toLowercase (char c)
- void Suspend (char *ProcessName)
- void Resume (char *ProcessName)
- void Set_Priority (char *ProcessName, int Priority)
- void Show_PCB (char *ProcessName)
- void Show_All ()
- void Show_Ready ()
- void Show_Blocked ()
- void Create_PCB (char *ProcessName, int Priority, int Class)
- void Delete_PCB (char *ProcessName)
- void Block (char *ProcessName)
- void Unblock (char *ProcessName)

4.22.1 Function Documentation

4.22.1.1 BCDtoDec()

Description: Changes binary number to decimal numbers.

Parameters

value Binary number to be changed to decimal

Definition at line 69 of file userFunctions.c.

4.22.1.2 Block()

Brief Description: Places a PCD in the blocked state and reinserts it into the correct queue.

Description: Can except a string as a pointer that is the Process Name. The specified PCB will be places in a blocked state and reinserted into the appropriate queue. An error check for a valid name occurs.

Parameters

Process_Name Character pointer that matches the name of process.

Definition at line 858 of file userFunctions.c.

```
859
      PCB* pcb = FindPCB(ProcessName);
860
      if (pcb == NULL)
        printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
861
862
863
        if(pcb->ReadyState == BLOCKED) {
   printf("\x1b[32m""\nThis Process is already BLOCKED \n""\x1b[0m");
864
865
866
       else
867
        RemovePCB (pcb);
869
          pcb->ReadyState = BLOCKED;
870
           InsertPCB(pcb);
871
872
      }
873 }
```

4.22.1.3 Create_PCB()

Brief Description: Calls SetupPCB() and inserts PCB into appropriate queue.

Description: Can except a string as a pointer that is the Process Name. Can accept two integers, Priority and Class. SetupPCB() will be called and the PCB will be inserted into the appropriate queue. An error check for unique and valid Process Name, an error check for valid process class, and an error check for process priority.

Parameters

Process_Name	Character pointer that matches the name of process.
Priority	integer that matches the priority number.
Class	integer that matches the class number.

Definition at line 812 of file userFunctions.c.

```
813
      if (FindPCB(ProcessName) == NULL) {
       if (Priority >= 0 && Priority < 10) {
   if (Class == 0 || Class == 1) {</pre>
814
815
816
            PCB* pcb = SetupPCB(ProcessName, Class, Priority);
817
            InsertPCB(pcb);
818
          } else{
            printf("\x1b[31m""\nERROR: Not a valid Class \n""\x1b[0m");
819
820
822
          printf("\x1b[31m""\\nERROR: Not a valid Priority \n""\\x1b[0m");
823
825 printf("\x1b[31m""\nERROR: This Process Name already exists \n""\x1b[0m"); 826 }
827 }
```

4.22.1.4 DectoBCD()

```
int DectoBCD (
          int Decimal )
```

Description: Changes decimal numbers to binary numbers.

Parameters

De	ecimal	Decimal number to be changed to binary
----	--------	--

Definition at line 76 of file userFunctions.c.

4.22.1.5 Delete_PCB()

Brief Description: Removes PCB from appropriate queue and frees all associated memory.

Description: Can except a string as a pointer that is the Process Name. Removes PCB from the appropriate queue and then frees all associated memory. An error check to make sure process name is valid.

Parameters

Process Name	Character pointer that matches the name of process.
	Character penner that materies the name of process.

Definition at line 838 of file userFunctions.c.

4.22.1.6 EdgeCase()

Description: Compares pointer char to validate if it is a number or not.

Parameters

Compares pointer char to validate if it is a number or not.

Definition at line 97 of file userFunctions.c.

```
int valid = 0;
       if (strcmp(pointer, "00") == 0) {
99
100
          valid = 1;
          return valid;
101
102
        else if (strcmp(pointer, "0") == 0) {
103
104
          valid = 1;
         return valid;
105
106
        else {
107
108
        int j;
            valid = 0;
for(j = 0; j <= 99; j++) {
    if(strcmp(pointer,itoa(j)) == 0)</pre>
109
110
111
112
                      valid = 1;
113
            if(valid == 0) {
114
            return valid;
115
116
117
118
        return valid;
      }
119
```

4.22.1.7 GetDate()

```
void GetDate ( )
```

Description: Returns the full date back to the user in decimal form.

No parameters.

Definition at line 254 of file userFunctions.c.

```
254 {
255    int check = 2;
256    outb(0x70,0x07);
257    unsigned char day = BCDtoDec(inb(0x71));
258    outb(0x70,0x08);
```

```
unsigned char month = BCDtoDec(inb(0x71));
            outb (0x70, 0x32);
261
            unsigned char millennium = BCDtoDec(inb(0x71));
262
            char msg[2] = "-";
            char msg3[10] = "Date: ";
263
264
           printf(msq3);
265
            sys_req(WRITE, COM1, itoa(day), &check);
266
            printf(msg);
            sys_req(WRITE, COM1, itoa(month), &check);
267
268
            printf(msg);
            sys_req(WRITE, COM1, itoa(millennium), &check);
269
270
       outb(0x70,0x09);
       if(BCDtoDec(inb(0x71)) == 0){
271
272
         sys_req(WRITE, COM1, "00", &check);
273
274
            unsigned char year = BCDtoDec(inb(0x71));
275
            sys_req(WRITE, COM1, itoa(year), &check);
276
           printf("\n");
```

4.22.1.8 GetTime()

```
void GetTime ( )
```

Description: retrieve and return the time values for hours, minutes, and seconds form the clock register using inb(Port,address).

No parameters.

Definition at line 176 of file userFunctions.c.

```
176
177
        int check = 2;
178
        int hour;
179
        int minute;
180
        int second;
            outb (0x70, 0x04);
181
182
            unsigned char hours = inb(0x71);
           outb(0x70,0x02);
183
184
           unsigned char minutes = inb(0x71);
185
           outb(0x70,0x00);
186
            unsigned char seconds = inb(0x71);
187
            char msg1[2] = ":";
           char msg2[10] = "Time: ";
188
189
            printf(msg2);
190
            hour = BCDtoDec(hours);
191
            sys_req(WRITE, COM1, itoa(hour), &check);
192
           printf(msg1);
193
            minute = BCDtoDec(minutes);
           sys_req(WRITE, COM1, itoa(minute), &check);
194
195
           printf(msg1);
196
            second = BCDtoDec(seconds);
197
            sys_req(WRITE, COM1, itoa(second), &check);
         printf("\n");
198
199
```

4.22.1.9 Help()

Brief Description: Gives helpful information for one of the functions

Description: Can except a string as a pointer, if the pointer is null then the function will print a complete list of avaliable commands to the console. If the pointer is a avaliable commands then instructions on how to use the command will be printed. If the command does not exist then a message explaining that it is not a valid command will be displayed.

Parameters

request

Character pointer that matches the name of the function that you need help with.

```
Definition at line 308 of file userFunctions.c.
```

```
if (request[0] == '\0')
309
310
                                printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
            printf("\n getDate \n setDate \n setTime \n setTime \n suspend \n resume \n setPriority \n showPCB \n showAll \n showReady \n showBlocked \n createPCB \n deletePCB \n block \n showBlocked \n showBlo
311
           unblock \n shutdown \n'");
312
                    else if (strcmp(request, "GetDate") == 0) {
                               printf("\n getDate returns the current date that is loaded onto the operating
314
           system.\n");
315
                   else if (strcmp(request, "SetDate") == 0) {
316
           printf("\n setDate allows the user to reset the correct date into the system, as follows setDate-"BLU"day"RESET"-"BLU"month"RESET"-"BLU"year"RESET".\n Time must be inputed as a two digit
317
           number, Example 02 or 00");
318
                   else if (strcmp(request, "GetTime") == 0) {
    printf("\n getTime returns the current time as hours, minutes, seconds that is loaded
319
320
           onto the operating system. \n");
321
                   else if (strcmp(request, "SetTime") == 0) {
322
           printf("\n setTime allows the user to reset the correct time into the system, as follows setTime-"BLU"hour"RESET"-"BLU"minute"RESET"-"BLU"second"RESET".\n Time must be inputed as a two digit
323
           number, Example 02 or 00");
324
325
                   else if (strcmp(request, "Version") == 0) {
326
                                printf("\n version returns the current operating software version that the system is
           running.\n");
327
            else if(strcmp(request, "shutdown") == 0) {
             printf("\n shutdown shuts down the system.\n");
}
328
329
330
331
333
334
                                R2 Commands
335
           336
            else if(strcmp(request, "suspend") == 0) {
337
             printf("\n Suspend takes in the name of a PCB (suspend-NAME) then places it into the suspended state
           and reinserts it into the correct queue.\n");
338
             else if(strcmp(request, "resume") == 0) {
339
            printf("\n Resume takes in the name of a PCB (resume-NAME) then removes it from the suspended state
           and adds it to the correct queue.\n");
341
342
             else if(strcmp(request, "setPriority") == 0) {
            printf("\n SetPriority takes in the name of a PCB and the priority (setPrioriry-NAME-PRIORITY) it
343
           needs to be set to then reinstates the specified PCB into a new location by priority.\n");
344
345
             else if(strcmp(request, "showPCB") == 0) {
346
            printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
           user.\n");
347
            else if(strcmp(request, "showAll") == 0) {
348
            printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
349
           queues.\n");
350
             else if(strcmp(request, "showReady") == 0) {
351
352
             \verb|printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that \\
           currently are in the ready state. \n");
353
354
            else if(strcmp(request, "showBlocked") == 0) {
            \verb|printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that \\
355
           currently are in the blocked state.\n");
356
357
358 /******* R2 Temp Commands
            else if(strcmp(request, "createPCB") == 0) {
359
360
            printf("\n CreatePCB takes in the process_name, process_class, and
           process_priority.(createPCB-NAME-PRIORITY-CLASS) Then assigns this new process into the correct
           queue.\n");
361
            }
            else if(strcmp(request, "deletePCB") == 0) {
362
363
            free's all the memory that was previously allocated to the specified PCB.\n");
```

```
364
        else if(strcmp(request, "block") == 0) {
        printf("\n Block takes in the process_name (block-NAME) then sets it's state to blocked and
366
       reinserts it back into the correct queue.\n");
367
        else if(strcmp(request, "unblock") == 0) {
368
        printf("\n Unblock takes in the process_name (unblock-NAME) then sets it's state to ready and
369
       reinserts it back into the correct queue.\n");
370
371
       printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a full list of commands.\n""\x1b[0m");
372
373
374 }
```

4.22.1.10 itoa()

```
char* itoa (
          int num )
```

Description: An integer is taken and seperated into individual chars and then all placed into a character array. Adapted from geeksforgeeks.org.

Parameters

```
num integer to be put into array Title: itoa Author: Neha Mahajan Date: 29 May, 2017 Availability: https://www.geeksforgeeks.org/implement-itoa/
```

Definition at line 38 of file userFunctions.c.

```
40
                 int i, j, k, count;
41
                 i = num;
                 j = 0;
42
                 count = 0;
44
            while(i){ // count number of digits
45
                 count++;
46
                 i /= 10;
47
            }
48
            char* arr1;
50
            char arr2[count];
            arr1 = (char*)sys_alloc_mem(count); //memory allocation
52
            while(num){ // seperate last digit from number and add ASCII
    arr2[++j] = num%10 + '0';
5.3
54
                 num /= 10;
55
57
            for (k = 0; k < j; k++) \{ // \text{ reverse array results} \}
58
59
                 arr1[k] = arr2[j-k];
60
            arr1[k] = ' \setminus 0';
61
            return(char*)arr1;
64
```

4.22.1.11 Resume()

Brief Description: Places a PCD in the not suspended state and reinserts it into the appropriate queue.

Description: Can except a string as a pointer that is the Process Name. Places a PCB in the not suspended state and reinserts it into the appropriate queue. An error check for valid Process Name.

Parameters

Process Name Character pointer that mate	hes the name of process.
--	--------------------------

Definition at line 416 of file userFunctions.c.

```
417
         PCB* pcb = FindPCB(ProcessName);
418
         if (pcb == NULL) {
           printf(RED"\nERROR: Not a valid process name \n"RESET);
419
420
421
         else {
           if (pcb->SuspendedState == NO)
422
              printf(GRN"\nThis Process is already in the NONSUSPENDED state \n"RESET);
423
424
425
           else
               pcb->SuspendedState = NO;
426
427
428
         }
429 }
```

4.22.1.12 Set_Priority()

Brief Description: Sets PCB priority and reinserts the process into the correct place in the correct queue.

Description: Can except a string as a pointer that is the Process Name. Can accept and integer than is the Priority. Sets a PCB's priority and reinserts the process into the correct place in the correct queue. An error check for valid Process Name and an error check for a valid priority 1 - 9.

Parameters

Process_Name	Character pointer that matches the name of process.
Priority	integer that matches the priority number.

Definition at line 441 of file userFunctions.c.

```
441
442
          PCB* pcb = FindPCB(ProcessName);
          if (pcb == NULL)
443
            printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
          else if(Priority >= 10) {
  printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
}
445
446
447
448
          else {
449
450
          RemovePCB(pcb);
451
                pcb->Priority = Priority;
452
                 InsertPCB(pcb);
453
          }
454 }
```

4.22.1.13 SetDate()

```
void SetDate (
          int day,
```

```
int month,
int millennium,
int year )
```

Description: Sets the date register to the new values that the user inputed, all values must be inputed as Set

Dime(day, month, millenial, year).

Parameters

day	Integer to be set in the Day position
month	Integer to be set in the Month position
millenial	Integer to be set in the Millenial position
year	Integer to be set in the Year position

Definition at line 207 of file userFunctions.c.

```
207
        outb (0x70, 0x07);
208
209
        int tempDay = BCDtoDec(inb(0x71));
210
        outb(0x70,0x08);
        int tempMonth = BCDtoDec(inb(0x71));
211
212
        outb (0x70, 0x32);
213
        int tempMillennium = BCDtoDec(inb(0x71));
214
        outb (0x70, 0x09);
215
        int tempYear = BCDtoDec(inb(0x71));
216
        cli();
217
            outb(0x70,0x07);
            outb(0x71, DectoBCD (day));
outb(0x70, 0x08);
218
219
            outb(0x71,DectoBCD (month));
220
            outb(0x70,0x32);
221
222
            outb(0x71,DectoBCD (millennium));
223
            outb (0x70, 0x09);
224
            outb(0x71, DectoBCD (year));
225
            sti();
        outb(0x70,0x07);
226
227
        unsigned char newDay = BCDtoDec(inb(0x71));
228
        outb (0x70, 0x08);
229
        unsigned char newMonth = BCDtoDec(inb(0x71));
230
        outb (0x70, 0x32);
231
        unsigned char newMillennium = BCDtoDec(inb(0x71));
        outb(0x70,0x09);
232
233
        unsigned char newYear = BCDtoDec(inb(0x71));
        if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
   printf("Your input was invalid\n");
234
235
236
          cli();
            outb(0x70,0x07);
237
            outb(0x71,DectoBCD (tempDay));
238
239
            outb (0x70, 0x08);
240
            outb(0x71,DectoBCD (tempMonth));
241
            outb (0x70, 0x32);
            outb(0x71,DectoBCD (tempMillennium));
242
            outb(0x70,0x09);
243
            outb(0x71,DectoBCD (tempYear));
244
245
            sti();
246
247
        printf("Date Set\n");
}
        else
248
249
```

4.22.1.14 SetTime()

Description: sets the time register to the new values that the user inputed, all values must be inputed as SetTime(← Hours, Minutes, Seconds).

Parameters

hours	Integer to be set in the Hour position
minutes	Integer to be set in the Minutes position
seconds	Integer to be set in the Seconds position

Definition at line 137 of file userFunctions.c.

```
138
        outb (0x70, 0x04);
139
        unsigned char tempHours = BCDtoDec(inb(0x71));
140
        outb(0x70,0x02);
141
        unsigned char tempMinutes = BCDtoDec(inb(0x71));
142
        outb (0x70, 0x00);
        unsigned char tempSeconds = BCDtoDec(inb(0x71));
143
            cli(); //outb(device + 1, 0x00); //disable interrupts
144
145
            outb(0x70,0x04);
146
            outb(0x71, DectoBCD(hours));// change to bcd
147
            outb(0x70,0x02);
148
            outb(0x71, DectoBCD(minutes));
149
            outb(0x70,0x00);
            outb(0x71, DectoBCD(seconds));
sti(); //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
150
151
        outb (0x70, 0x04);
152
153
        unsigned char newHours = BCDtoDec(inb(0x71));
154
        outb (0x70, 0x02);
        unsigned char newMinutes = BCDtoDec(inb(0x71));
155
156
        outb(0x70,0x00);
157
        unsigned char newSeconds = BCDtoDec(inb(0x71));
        if (newHours != hours || newMinutes != minutes || newSeconds != seconds) {
    printf("Your input was invalid\n");
158
159
160
          cli(); //outb(device + 1, 0x00); //disable interrupts
161
            outb (0x70, 0x04);
            outb(0x71, DectoBCD(tempHours));// change to bcd
outb(0x70,0x02);
162
163
            outb(0x71, DectoBCD(tempMinutes));
164
165
            outb(0x70,0x00);
166
             outb(0x71, DectoBCD(tempSeconds));
167
            sti(); //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
168
        else
169
170
          printf("Time Set\n");
```

4.22.1.15 Show_All()

```
void Show_All ( )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the ready and blocked queues.

Description: The process name, claas, state, suspend status, and priority of each of he PCB's in the ready and blocked queues.

Definition at line 535 of file userFunctions.c.

4.22.1.16 Show_Blocked()

```
void Show_Blocked ( )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the blocked queue.

Description: The process name, claas, state, suspend status, and priority of each of he PCB's in the blocked queue.

Definition at line 679 of file userFunctions.c.

```
680
         if (getBlocked()->head == NULL)
              printf("\x1b[32m""\n The Blocked Queue is empty \n""\x1b[0m");
681
682
683
        else
               int class, check, state, prior, status;
char name[20];
char block[] = "\x1B[34m""Blocked Queue: \n""\x1b[0m";
char cname[] = "Name: ";
684
685
686
687
               char cclass[] = "Class: ";
               char cstate[] = "State: ";
689
               char cstatus[] = "Status: ";
char cprior[] = "Priority: ";
char line[] = "\n";
690
691
692
693
               check = 15;
694
695
               sys_req(WRITE, COM1, block, &check);
696
               PCB* pcb = getBlocked()->head;
697
698
699
               if(pcb->next == NULL) {
                  class = pcb->Process_Class;
701
                    strcpy(name,pcb->Process_Name);
702
                      state = pcb->ReadyState;
                      status = pcb->SuspendedState;
703
                      prior = pcb->Priority;
704
705
706
                      printf(cname);
707
                      printf(name);
708
                      printf(line);
709
710
                      printf(cclass);
711
                      if (pcb->Process_Class == 0) {
712
                        printf("0");
713
714
                      else
715
                        sys_req(WRITE, COM1, itoa(class), &check);
716
                      printf(line);
717
718
719
                      printf(cstate);
720
                      if(pcb->ReadyState == 0) {
721
                        printf("0");
722
723
                      else {
724
                        sys_req(WRITE, COM1, itoa(state), &check);
725
726
                      printf(line);
727
728
                      printf(cstatus);
                      if (pcb->SuspendedState == 0) {
729
730
                        printf("0");
731
732
                      else {
733
                        sys_req(WRITE, COM1, itoa(status), &check);
734
                      printf(line);
735
736
737
                      printf(cprior);
738
                      if (pcb->Priority == 0)
739
                         printf("0");
                        printf("\n\n");
740
741
742
                      else {
                        sys_req(WRITE, COM1, itoa(prior), &check);
743
744
                        printf("\n\n");
745
746
747
               else {
748
                  while (pcb != NULL) {
749
                     class = pcb->Process_Class;
                          strcpy(name,pcb->Process_Name);
```

```
751
                         state = pcb->ReadyState;
752
                         status = pcb->SuspendedState;
753
                        prior = pcb->Priority;
754
755
                         printf(cname);
756
                         printf(name);
                        printf(line);
758
759
                         printf(cclass);
                         if(pcb->Process_Class == 0) {
760
                          printf("0");
761
762
763
                         else {
764
                          sys_req(WRITE, COM1, itoa(class), &check);
765
766
                         printf(line);
767
768
                         printf(cstate);
769
                         if(pcb->ReadyState == 0) {
770
                          printf("0");
771
772
                         else {
773
                          sys_req(WRITE, COM1, itoa(state), &check);
774
775
                        printf(line);
776
777
                         printf(cstatus);
778
                         if(pcb->SuspendedState == 0) {
779
                          printf("0");
780
781
                         else {
782
                          sys_req(WRITE, COM1, itoa(status), &check);
783
784
                         printf(line);
785
                         printf(cprior);
786
787
                         if(pcb->Priority == 0) {
                          printf("0");
788
789
                           printf("\n\n");
790
791
                         else {
                          sys_req(WRITE, COM1, itoa(prior), &check); printf("\n^n");
792
793
794
                    pcb = pcb->next;
796
               }
797
798
        }
799 }
```

4.22.1.17 Show_PCB()

Brief Description: Displays the process name, class, state, suspended status, and priority of a PCB.

Description: Can except a string as a pointer that is the Process Name. The process name, claas, state, suspend status, and priority of a PCB are displayed. An error check for a valid name occurs.

Parameters

Process_Name | Character pointer that matches the name of process

Definition at line 464 of file userFunctions.c.

```
char name[10];
               char cname[] = "Name: ";
char cclass[] = "Class: ";
471
472
               char cstate[] = "State: ";
473
               char cstatus[] = "Status: ";
474
               char cstatus[] = Status. ,
char cprior[] = "Priority: ";
char line[] = "\n";
475
476
477
               PCB* pcb = FindPCB(ProcessName);
478
               strcpy(name,pcb->Process_Name);
               int class = pcb->Process_Class;
int state = pcb->ReadyState;
479
480
               int status = pcb->SuspendedState;
481
               int prior = pcb->Priority;
482
483
484
               if (name == NULL) {
                      printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
485
486
487
               else
488
                      printf(cname);
                 printf(ProcessName);
490
                 printf(line);
491
                 printf(cclass);
                   if(pcb->Process_Class == 0) {
492
                     printf("0");
493
494
495
                   else {
496
                      sys_req(WRITE, COM1, itoa(class), &check);
497
498
                      printf(line);
499
                 printf(cstate);
                   if (pcb->ReadyState == 0) {
500
501
                     printf("0");
502
503
                   else {
504
                      sys_req(WRITE, COM1, itoa(state), &check);
505
                      printf(line);
506
                 printf(cstatus);
507
508
                   if(pcb->SuspendedState == 0)
509
                     printf("0");
510
511
                   else {
                     sys_req(WRITE, COM1, itoa(status), &check);
512
513
                   }
514
                     printf(line);
515
                 printf(cprior);
516
                   if (pcb->Priority == 0) {
                     printf("0");
printf("\n\n");
517
518
519
520
                   else {
521
                      sys_req(WRITE, COM1, itoa(prior), &check);
522
                     printf("\n\n");
523
524
             }
        }
525
```

4.22.1.18 Show_Ready()

```
void Show_Ready ( )
```

Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the ready queue.

Description: The process name, claas, state, suspend status, and priority of each of he PCB's in the ready queue.

Definition at line 547 of file userFunctions.c.

```
555
                char name[10];
               char rame[10];
char ready[] = "\x1B[34m""\nReady Queue:\n""\x1B[0m";
char cname[] = "Name: ";
char cclass[] = "Class: ";
char cstate[] = "State: ";
char cstatus[] = "Status: ";
556
557
558
559
560
                char cprior[] = "Priority:
561
562
                char line[] = "\n";
563
                check = 5;
564
                sys_req(WRITE, COM1, ready, &check);
565
566
567
                PCB* pcb = getReady()->head;
568
569
                if (pcb->next == NULL) {
570
                  class = pcb->Process_Class;
571
                       strcpy(name,pcb->Process_Name);
                       state = pcb->ReadyState;
status = pcb->SuspendedState;
572
573
574
                      prior = pcb->Priority;
575
576
                       printf(cname);
577
                       printf(name);
578
                       printf(line);
579
580
                       printf(cclass);
581
                       if (pcb->Process_Class == 0) {
                         printf("0");
582
583
584
                       else {
                         sys_req(WRITE, COM1, itoa(class), &check);
585
586
587
                       printf(line);
588
589
                       printf(cstate);
                       if(pcb->ReadyState == 0) {
590
                         printf("0");
591
592
593
594
                        sys_req(WRITE, COM1, itoa(state), &check);
595
596
                       printf(line);
597
598
                       printf(cstatus);
599
                       if(pcb->SuspendedState == 0) {
                         printf("0");
600
601
602
                       else {
                         sys_req(WRITE, COM1, itoa(status), &check);
603
604
605
                       printf(line);
606
607
                       printf(cprior);
                       if(pcb->Priority == 0) {
  printf("0");
608
609
                         printf("\n\n");
610
612
613
                         sys_req(WRITE, COM1, itoa(prior), &check);
                         printf("\n\n");
614
615
616
617
                else {
618
                  while (pcb != NULL) {
619
                       class = pcb->Process_Class;
620
                           strcpy(name,pcb->Process_Name);
                           state = pcb->ReadyState;
status = pcb->SuspendedState;
621
622
623
                           prior = pcb->Priority;
624
625
                           printf(cname);
62.6
                            printf(name);
62.7
                           printf(line);
628
629
                            printf(cclass);
630
                            if (pcb->Process_Class == 0) {
631
                             printf("0");
632
633
                            else {
                             sys_req(WRITE, COM1, itoa(class), &check);
634
635
636
                           printf(line);
637
638
                            printf(cstate);
639
                            if(pcb->ReadyState == 0) {
                              printf("0");
640
641
```

```
643
                          sys_req(WRITE, COM1, itoa(state), &check);
644
                         printf(line);
645
646
                         printf(cstatus);
647
648
                         if (pcb->SuspendedState == 0) {
649
                           printf("0");
650
651
                         else {
                           sys_req(WRITE, COM1, itoa(status), &check);
652
653
654
                         printf(line);
655
656
                         printf(cprior);
                         if(pcb->Priority == 0) {
  printf("0");
657
658
                           printf("\n\n");
659
660
661
                         else {
662
                           sys_req(WRITE, COM1, itoa(prior), &check);
                           printf("\n\n");
663
664
                     pcb = pcb->next;
665
                }
666
667
668
669
        }
670 }
```

4.22.1.19 Suspend()

Brief Description: Places a PCD in the suspended state and reinserts it into the appropriate queue.

Description: Can except a string as a pointer that is the Process Name. Places a PCB in the suspended state and reinserts it into the appropriate queue. An error check for valid Process Name.

Parameters

Process_Name | Character pointer that matches the name of process.

Definition at line 391 of file userFunctions.c.

```
391
392
          PCB* pcb = FindPCB(ProcessName);
          if (pcb == NULL) {
  printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
393
394
395
396
          else {
            if(pcb->SuspendedState == YES) {
397
                printf("\x1b[32m""\nThis Process is already SUSPENDED \n""\x1b[0m");
398
399
            else
401
                pcb->SuspendedState = YES;
402
403
          }
404
405 }
```

4.22.1.20 toLowercase()

```
char toLowercase ( {\tt char}\ c\ )
```

Description: If a letter is uppercase, it changes it to lowercase. (char)

Parameters

c Character that is to be changed to its lowercase equivalent

Definition at line 291 of file userFunctions.c.

4.22.1.21 Unblock()

Brief Description: Places a PCD in the unblocked state and reinserts it into the correct queue.

Description: Can except a string as a pointer that is the Process Name. The specified PCB will be places in an unblocked state and reinserted into the appropriate queue. An error check for a valid name occurs.

Parameters

Process_Name | Character pointer that matches the name of process.

Definition at line 884 of file userFunctions.c.

```
PCB* pcb = FindPCB(ProcessName);
886
      if (pcb == NULL) {
       printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
887
888
889
     else {
       if(pcb->ReadyState == READY) {
    printf("\x1b[32m""\nThis Process is already in the READY state \n""\x1b[0m");
890
891
892
       else
893
894
        RemovePCB(pcb);
          pcb->ReadyState = READY;
895
896
          InsertPCB(pcb);
897
        }
898 }
899 }
```

4.22.1.22 Version()

```
void Version ( )
```

Description: Simply returns a char containing "Version: R(module). (the iteration that module is currently on).

No parameters.

Definition at line 284 of file userFunctions.c.

```
284 {
285 printf("Version: R2.6 \n");
286 }
```

Index

atoi	io.h, 13
string.c, 25	include/core/asm.h, 13
string.h, 16	include/core/interrupts.h, 13
	include/core/io.h, 13
BCDtoDec	include/core/serial.h, 14
userFunctions.c, 34	include/core/tables.h, 14
userFunctions.h, 50	include/mem/heap.h, 15
Block	include/mem/paging.h, 15
userFunctions.c, 34	include/string.h, 16
userFunctions.h, 51	include/system.h, 20
	index_entry, 8
comHand	index_table, 8
comHand.h, 31	io.h
comHand.h	inb, 13
comHand, 31	isspace
Create_PCB	string.c, 26
userFunctions.c, 35	string.h, 16
userFunctions.h, 51	itoa
data tima E	userFunctions.c, 39
date_time, 5	userFunctions.h, 56
DectoBCD	
userFunctions.c, 36	kernel/core/interrupts.c, 21
userFunctions.h, 52	kernel/core/kmain.c, 22
Delete_PCB	kernel/core/serial.c, 22
userFunctions.c, 36	kernel/core/system.c, 23
userFunctions.h, 52	kernel/core/tables.c, 23
EdgaCasa	kernel/mem/heap.c, 23
EdgeCase userFunctions.c, 36	kernel/mem/paging.c, 24
userFunctions.h, 53	
useri unctions.ii, 55	lib/string.c, 25
footer, 5	
	memset
gdt_descriptor_struct, 6	string.c, 26
gdt_entry_struct, 6	string.h, 17
GetDate	modules/mpx_supt.c, 29
userFunctions.c, 37	modules/mpx_supt.h, 30
userFunctions.h, 53	modules/R1/comHand.h, 30
GetTime	modules/R1/userFunctions.c, 33
userFunctions.c, 37	modules/R1/userFunctions.h, 50
userFunctions.h, 54	page_dir, 9
, ,	
header, 6	page_entry, 9
heap, 7	page_table, 9
Help	param, 10
userFunctions.c, 38	PCB, 10
userFunctions.h, 54	Queue, 11
, -	Queue, 11
idt_entry_struct, 7	Resume
idt_struct, 8	userFunctions.c, 40
inb	userFunctions.h, 56

68 INDEX

Set_Priority	userFunctions.c, 48
userFunctions.c, 41	userFunctions.h, 64
userFunctions.h, 57	
SetDate	Unblock
userFunctions.c, 41	userFunctions.c, 49
userFunctions.h, 57	userFunctions.h, 66
SetTime	userFunctions.c
userFunctions.c, 42	BCDtoDec, 34
userFunctions.h, 58	Block, 34
Show All	Create_PCB, 35
userFunctions.c, 43	DectoBCD, 36
userFunctions.h, 59	Delete PCB, 36
Show Blocked	EdgeCase, 36
userFunctions.c, 43	GetDate, 37
	GetTime, 37
userFunctions.h, 59	Help, 38
Show_PCB	itoa, 39
userFunctions.c, 45	Resume, 40
userFunctions.h, 61	•
Show_Ready	Set_Priority, 41
userFunctions.c, 46	SetDate, 41
userFunctions.h, 62	SetTime, 42
strcat	Show_All, 43
string.c, 26	Show_Blocked, 43
string.h, 17	Show_PCB, 45
strcmp	Show_Ready, 46
string.c, 27	Suspend, 48
string.h, 18	toLowercase, 48
strcpy	Unblock, 49
	Version, 49
string.c, 27	version, 40
string.c, 27 string.h, 18	userFunctions.h
string.h, 18	
string.h, 18 string.c	userFunctions.h
string.h, 18 string.c atoi, 25	userFunctions.h BCDtoDec, 50
string.h, 18 string.c atoi, 25 isspace, 26	userFunctions.h BCDtoDec, 50 Block, 51
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTime, 58
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcmp, 18	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTime, 58 Show_All, 59
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcmp, 18 strcpy, 18	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTime, 58 Show_All, 59 Show_Blocked, 59
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcmp, 18 strcpy, 18 strlen, 19	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTime, 58 Show_All, 59 Show_Blocked, 59 Show_PCB, 61
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcat, 17 strcmp, 18 strlen, 19 strtok, 19	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTime, 58 Show_All, 59 Show_Blocked, 59 Show_PCB, 61 Show_Ready, 62
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcat, 17 strcmp, 18 strlen, 19 strlen, 19 strlen	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTime, 58 Show_All, 59 Show_Blocked, 59 Show_PCB, 61 Show_Ready, 62 Suspend, 64
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcat, 17 strcmp, 18 strlen, 19 strlen, 19 strlen string.c, 28	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTate, 57 SetTime, 58 Show_All, 59 Show_Blocked, 59 Show_PCB, 61 Show_Ready, 62 Suspend, 64 toLowercase, 64
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcat, 17 strcmp, 18 strcpy, 18 strlen, 19 strlen string.c, 28 string.h, 19	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTime, 58 Show_All, 59 Show_Blocked, 59 Show_PCB, 61 Show_Ready, 62 Suspend, 64 toLowercase, 64 Unblock, 66
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcat, 17 strcmp, 18 strlen, 19 strlen, 19 strlen string.c, 28 string.h, 19 strtok	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTate, 57 SetTime, 58 Show_All, 59 Show_Blocked, 59 Show_PCB, 61 Show_Ready, 62 Suspend, 64 toLowercase, 64
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcmp, 18 strcpy, 18 strlen, 19 strlen, 19 strtok, 19 strlen string.c, 28 string.h, 19 strtok string.c, 28	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTime, 58 Show_All, 59 Show_Blocked, 59 Show_PCB, 61 Show_Ready, 62 Suspend, 64 toLowercase, 64 Unblock, 66 Version, 66
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcat, 17 strcmp, 18 strcpy, 18 strlen, 19 strlen, 19 strlen, 19 strlen string.c, 28 string.h, 19 strtok string.c, 28 string.h, 19	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTime, 58 Show_All, 59 Show_Blocked, 59 Show_PCB, 61 Show_Ready, 62 Suspend, 64 toLowercase, 64 Unblock, 66 Version
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcat, 17 strcmp, 18 strcpy, 18 strlen, 19 strtok, 19 strlen string.c, 28 string.h, 19 strtok string.c, 28 string.h, 19 Suspend	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTime, 58 Show_All, 59 Show_Blocked, 59 Show_PCB, 61 Show_Ready, 62 Suspend, 64 toLowercase, 64 Unblock, 66 Version userFunctions.c, 49
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcat, 17 strcmp, 18 strcpy, 18 strlen, 19 strlen, 19 strlen, 19 strlen string.c, 28 string.h, 19 strtok string.c, 28 string.h, 19	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTime, 58 Show_All, 59 Show_Blocked, 59 Show_PCB, 61 Show_Ready, 62 Suspend, 64 toLowercase, 64 Unblock, 66 Version
string.h, 18 string.c atoi, 25 isspace, 26 memset, 26 strcat, 26 strcmp, 27 strcpy, 27 strlen, 28 strtok, 28 string.h atoi, 16 isspace, 16 memset, 17 strcat, 17 strcat, 17 strcmp, 18 strcpy, 18 strlen, 19 strtok, 19 strlen string.c, 28 string.h, 19 strtok string.c, 28 string.h, 19 Suspend	userFunctions.h BCDtoDec, 50 Block, 51 Create_PCB, 51 DectoBCD, 52 Delete_PCB, 52 EdgeCase, 53 GetDate, 53 GetTime, 54 Help, 54 itoa, 56 Resume, 56 Set_Priority, 57 SetDate, 57 SetTime, 58 Show_All, 59 Show_Blocked, 59 Show_PCB, 61 Show_Ready, 62 Suspend, 64 toLowercase, 64 Unblock, 66 Version userFunctions.c, 49

toLowercase