Runtime Terror

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

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

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

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]

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

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

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

Value:

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

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 (
            const char * s )
```

Description: Convert an ASCII string to an integer

Parameters

s	String

Definition at line 50 of file string.c.

```
51 {
52   int res=0;
53    int charVal=0;
54    char sign = ' ';
55    char c = *s;
56 
57 
58    while(isspace(&c)){ ++s; c = *s;} // advance past whitespace
59 
60 
61    if (*s == '-' || *s == '+') sign = *(s++); // save the sign
62 
63 
64    while(*s != '\0'){
65         charVal = *s - 48;
66     res = res * 10 + charVal;
67     s++;
68 
69    }
70 
71 
72    if ( sign == '-') res=res * -1;
73 
74   return res; // return integer
75 }
```

4.8.1.2 isspace()

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

Description: Determine if a character is whitespace.

Parameters

c	character to check

Definition at line 121 of file string.c.

```
122 {
123   if (*c == ' '  ||
124       *c == '\n' ||
125       *c == '\r' ||
126       *c == '\f' ||
127       *c == '\t' ||
128       *c == '\v'){
129     return 1;
130   }
131   return 0;
132 }
```

4.8.1.3 memset()

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

Description: Set a region of memory.

Parameters

s	destination
c	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 }
```

4.8.1.4 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.

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

Description: String comparison

Parameters

s1	string 1
s2	string 2

Definition at line 81 of file string.c.

```
82 {
83
84   // Remarks:
85   // 1) If we made it to the end of both strings (i. e. our pointer points to a
86   //    '\0' character), the function will return 0
87   // 2) If we didn't make it to the end of both strings, the function will
88   //    return the difference of the characters at the first index of
89   //    indifference.
90   while ( (*s1) && (*s1==*s2) ){
91     ++s1;
92     ++s2;
93   }
94   return ( *(unsigned char *)s1 - *(unsigned char *)s2 );
95 }
```

4.8.1.6 strcpy()

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

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 {
164     if (tok_tmp==NULL){
165       return NULL;
166     }
167     s1 = tok_tmp;
168   }
169
170   //skip leading s2 characters
171   while ( *p && *s1 ){
```

```
172     if (*s1==*p){
173       ++s1;
174       p = s2;
175       continue;
176     }
177     ++p;
178   }
179
180   //no more to parse
181   if (!*s1){
182     return (tok_tmp = NULL);
183   }
184
185   //skip non-s2 characters
186   tok_tmp = s1;
187   while (*tok_tmp){
188     p = s2;
189     while (*p){
190       if (*tok_tmp==*p++){
191     *tok_tmp++ = '\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)
- void **new_frame** (page_entry *page)

Variables

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

4.17 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 (
            const char * s )
```

Description: Convert an ASCII string to an integer

Parameters

s	String

Definition at line 50 of file string.c.

```
51 {
52   int res=0;
53    int charVal=0;
54    char sign = ' ';
55    char c = *s;
56 
57 
58    while(isspace(&c)){ ++s; c = *s;} // advance past whitespace
59 
60 
61    if (*s == '-' || *s == '+') sign = *(s++); // save the sign
62 
63 
64    while(*s != '\0'){
65         charVal = *s - 48;
66     res = res * 10 + charVal;
67     s++;
68 
69    }
70 
71 
72    if ( sign == '-') res=res * -1;
73 
74   return res; // return integer
75 }
```

4.17.1.2 isspace()

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

Description: Determine if a character is whitespace.

Parameters

c	character to check

Definition at line 121 of file string.c.

```
122 {
123   if (*c == ' '  ||
124       *c == '\n' ||
125       *c == '\r' ||
126       *c == '\f' ||
127       *c == '\t' ||
128       *c == '\v'){
129     return 1;
130   }
131   return 0;
132 }
```

4.17.1.3 memset()

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

Description: Set a region of memory.

Parameters

s	destination
c	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.

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

4.17.1.5 strcmp()

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

Description: String comparison

Parameters

s1	string 1
s2	string 2

Definition at line 81 of file string.c.

```
82 {
83
84   // Remarks:
85   // 1) If we made it to the end of both strings (i. e. our pointer points to a
86   //    '\0' character), the function will return 0
87   // 2) If we didn't make it to the end of both strings, the function will
88   //    return the difference of the characters at the first index of
89   //    indifference.
90   while ( (*s1) && (*s1==*s2) ){
91     ++s1;
92     ++s2;
93   }
94   return ( *(unsigned char *)s1 - *(unsigned char *)s2 );
95 }
```

4.17.1.6 strcpy()

```
char* strcpy (
            char * s1,
            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 }
```

4.17.1.7 strlen()

```
int strlen (
            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()

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

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

```
22                      {
23
24          Help("\0");
25
26          char cmdBuffer[100];
27          int bufferSize = 99;
28          int quit = 0;
29          int shutdown = 0;
30
31          while(quit != 1)    {
32              memset(cmdBuffer, '\0', 100);
33            sys_req(READ, DEFAULT_DEVICE, cmdBuffer, &bufferSize);
34              char* FirstToken = strtok(cmdBuffer, "-");
35          char* SecondToken = strtok(NULL, "-");
36              char* ThirdToken = strtok(NULL, "-");
37              char* FourthToken = strtok(NULL, "-");
38              char* FifthToken = strtok(NULL, "-");
39              if(shutdown == 0)   {
40 /*****************************************************************************
41                  R1 comHand
42 *****************************************************************************/
43                      if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,NULL) == 0)     {
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                      }
50                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"getDate") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
51                          Help("GetDate");
52                      }
53                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"setDate") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
54                          Help("SetDate");
55                      }
56                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"getTime") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
57                          Help("GetTime");
58                      }
59                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"setTime") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
60                          Help("SetTime");
61                      }
62                      // R2 Commands
63                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"suspend") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
64                          Help("suspend");
65                      }
66                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"resume") == 0 &&
       strcmp(ThirdToken,NULL) == 0)  {
67                          Help("resume");
68                      }
69                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"setPriority") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
70                          Help("setPriority");
71                      }
72                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showPCB") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
73                          Help("showPCB");
74                      }
75                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showAll") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
76                          Help("showAll");
77                      }
78                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showReady") == 0 &&
       strcmp(ThirdToken,NULL) == 0)   {
```

```
79                      Help("showReady");
80                  }
81                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showBlocked") == 0 &&
    strcmp(ThirdToken,NULL) == 0) {
82                      Help("showBlocked");
83                  }
84                  // Temporary R2 commands
85                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"createPCB") == 0 &&
    strcmp(ThirdToken,NULL) == 0)   {
86                      Help("createPCB");
87                  }
88                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"deletePCB") == 0 &&
    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) {
95                      Help("unblock");
96                  }
97                  else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"shutdown") == 0 &&
    strcmp(ThirdToken,NULL) == 0)    {
98                      Help("shutdown");
99                  }
100
101                  else if(strcmp(FirstToken,"version") == 0 && strcmp(SecondToken,NULL) == 0)
102                      Version();
103
104                  else if(strcmp(FirstToken,"getDate") == 0 && strcmp(SecondToken,NULL) == 0)
105                      GetDate();
106
107                  else if(strcmp(FirstToken,"setDate") == 0){
108                      if (EdgeCase(SecondToken) == 1 && EdgeCase(ThirdToken) == 1 &&
    EdgeCase(FourthToken) == 1 && EdgeCase(FifthToken) == 1)    {
109                          SetDate(atoi(SecondToken), atoi(ThirdToken), atoi(FourthToken),
    atoi(FifthToken));
110                      }
111                      else
112                          printf("\x1b[31m""\nERROR: Invalid parameters for setDate \n""\x1b[0m");
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                      else
121                          printf("\x1b[31m""\nERROR: Invalid parameters for setTime \n""\x1b[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                  }
135                  else if(strcmp(FirstToken,"resume") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    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)   {
140                          Set_Priority(SecondToken, atoi(ThirdToken));    //input as
    setPriority-Process_Name-Priority
141                      }
142                      else
143                          printf("\x1b[31m""\nERROR: Invalid parameters for setPriority, priority must
    be entered as a integer. \n""\x1b[0m");
144                  }
145                  else if(strcmp(FirstToken,"showPCB") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
146                      Show_PCB(SecondToken);
147                      printf("\n");
148                  }
```

```
149                     else 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                     }
153                     else if(strcmp(FirstToken,"showReady") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
154                         Show_Ready();
155                         printf("\n");
156                     }
157                     else if(strcmp(FirstToken,"showBlocked") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
158                         Show_Blocked();
159                         printf("\n");
160                     }
161 
162                     /********** R2 Temp Commands **********/
163                     else if(strcmp(FirstToken,"createPCB") == 0) {
164                         if( strlen(SecondToken) < 11) {
165                                 Create_PCB(SecondToken, atoi(ThirdToken), atoi(FourthToken));
    //input as Process_Name-Priority-Class
166                         }
167                         else
168                             printf("\x1b[31m""\nERROR: Invalid parameters for createPCB, Process_name
    must only contain 10 or fewer characters. \n""\x1b[0m");
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                     }
176                     else if(strcmp(FirstToken,"unblock") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
177                         Unblock(SecondToken);
178                     }
179 
180      /*****************************************************************************
181                     shutdown comHand
182      *****************************************************************************/
183                     else if(strcmp(FirstToken,"shutdown") == 0 && strcmp(SecondToken,NULL) == 0){
184                         printf("\x1b[33m""\nAre you sure you want to shutdown? [yes/no]\n""\x1b[0m");
185                         shutdown = 1;
186                     }
187                     else    {
188                         printf("\x1b[31m""\nERROR: Not a valid command \n""\x1b[0m");
189                     }
190                 }
191                 else{
192                     if(strcmp(FirstToken,"yes") == 0 && shutdown == 1)    {
193                         quit = 1;
194                     }
195                     else if(strcmp(FirstToken,"no") == 0){
196                         printf("\x1b[33m""\nShutdown Cancelled\x1b[0m \n");
197                         shutdown = 0;
198                     }
199                     else
200                         printf("\x1b[31m""\nERROR: Please enter \"yes\" or \"no\" \n""\x1b[0m");
201                 }
202         }
203         return 0;   //shutdown procedure
204     }
```

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 (
            int 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.

```
69                              {
70          return (((BCD»4)*10) + (BCD & 0xF));
71      }
```

4.21.1.2 Block()

```
void Block (
            char * ProcessName )
```

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.

```
858                                     {
859   PCB* pcb = FindPCB(ProcessName);
860   if (pcb == NULL)  {
861     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
862   }
863   else {
864     if(pcb->ReadyState == BLOCKED)  {
865         printf("\x1b[32m""\nThis Process is already BLOCKED \n""\x1b[0m");
866     }
867     else    {
868       RemovePCB(pcb);
869       pcb->ReadyState = BLOCKED;
870       InsertPCB(pcb);
871     }
872   }
873 }
```

4.21.1.3 Create_PCB()

```
void Create_PCB (
             char * ProcessName,
             int Priority,
             int Class )
```

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.

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

```
826   }
827 }
```

4.21.1.4 DectoBCD()

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

Description: Changes decimal numbers to binary numbers.

Parameters

Decimal	Decimal number to be changed to binary

Definition at line 76 of file userFunctions.c.

```
76                                  {
77          return (((Decimal/10) « 4) | (Decimal % 10));
78     }
```

4.21.1.5 Delete_PCB()

```
void Delete_PCB (
              char * ProcessName )
```

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

4.21.1.6 EdgeCase()

```
int EdgeCase (
              char * pointer )
```

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.

```
97                                  {
98      int valid = 0;
99      if (strcmp(pointer, "00") == 0) {
100        valid = 1;
101        return valid;
102     }
103     else if (strcmp(pointer, "0") == 0) {
104       valid = 1;
105       return valid;
106     }
107     else  {
108     int j;
109         valid = 0;
110         for(j = 0; j <= 99; j++)  {
111             if(strcmp(pointer,itoa(j)) == 0)
112                 valid = 1;
113         }
114         if(valid == 0)  {
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;
256       outb(0x70,0x07);
257         unsigned char day = BCDtoDec(inb(0x71));
258         outb(0x70,0x08);
259         unsigned char month = BCDtoDec(inb(0x71));
260         outb(0x70,0x32);
261         unsigned char millennium = BCDtoDec(inb(0x71));
262         char msg[2] = "-";
263         char msg3[10] = "Date: ";
264         printf(msg3);
265         sys_req(WRITE, COM1, itoa(day), &check);
266         printf(msg);
267         sys_req(WRITE, COM1, itoa(month), &check);
268         printf(msg);
269         sys_req(WRITE, COM1, itoa(millennium), &check);
270     outb(0x70,0x09);
271     if(BCDtoDec(inb(0x71)) == 0){
272       sys_req(WRITE, COM1, "00", &check);
273     }
274     else {
275         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.

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

4.21.1.9 Help()

```
void Help (
              char * request )
```

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.

```
308                             {
309         if (request[0] == '\0') {
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
        setPriority \n showPCB \n showAll \n showReady \n showBlocked \n createPCB \n deletePCB \n block \n
        unblock \n shutdown \n\n");
312         }
313         else if (strcmp(request, "GetDate") == 0) {
314                 printf("\n getDate returns the current date that is loaded onto the operating
        system.\n");
315         }
316         else if (strcmp(request, "SetDate") == 0) {
```

```
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         }
319         else if (strcmp(request, "GetTime") == 0) {
320                 printf("\n getTime returns the current time as hours, minutes, seconds that is loaded
    onto the operating system.\n");
321         }
322         else if (strcmp(request, "SetTime") == 0) {
323                 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
    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         }
328      else if(strcmp(request, "shutdown") == 0) {
329        printf("\n shutdown shuts down the system.\n");
330      }
331
332
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      }
339      else if(strcmp(request,"resume") == 0) {
340      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) {
343      printf("\n SetPriority takes in the name of a PCB and the priority (setPrioriry-NAME-PRIORITY) it
    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      }
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      }
351      else if(strcmp(request,"showReady") == 0) {
352      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) {
355      printf("\n 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) {
366      printf("\n Block takes in the process_name (block-NAME) then sets it's state to blocked and
    reinserts it back into the correct queue.\n");
367      }
368      else if(strcmp(request,"unblock") == 0) {
369      printf("\n Unblock takes in the process_name (unblock-NAME) then sets it's state to ready and
    reinserts it back into the correct queue.\n");
370      }
371      else    {
372       printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
    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.

```
39     {
40             int i,j,k,count;
41             i = num;
42             j = 0;
43             count = 0;
44         while(i){ // count number of digits
45             count++;
46             i /= 10;
47         }
48
49         char* arr1;
50         char arr2[count];
51         arr1 = (char*)sys_alloc_mem(count); //memory allocation
52
53         while(num){ // seperate last digit from number and add ASCII
54             arr2[++j] = num%10 + '0';
55             num /= 10;
56         }
57
58         for(k = 0; k < j; k++){ // reverse array results
59             arr1[k] = arr2[j-k];
60         }
61         arr1[k] = '\0';
62
63         return(char*)arr1;
64     }
```

4.21.1.11 Resume()

```
void Resume (
            char * ProcessName )
```

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       }
```

```
421       else {
422         if(pcb->SuspendedState == NO)   {
423             printf(GRN"\nThis Process is already in the NONSUSPENDED state \n"RESET);
424         }
425         else    {
426             pcb->SuspendedState = NO;
427         }
428       }
429 }
```

4.21.1.12 Set_Priority()

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

4.21.1.13 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                                                                 {
208     outb(0x70,0x07);
209     int tempDay = BCDtoDec(inb(0x71));
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();
217         outb(0x70,0x07);
218         outb(0x71,DectoBCD (day));
219         outb(0x70,0x08);
220         outb(0x71,DectoBCD (month));
221         outb(0x70,0x32);
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);
233     unsigned char newYear = BCDtoDec(inb(0x71));
234     if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
235       printf("Your input was invalid\n");
236       cli();
237         outb(0x70,0x07);
238         outb(0x71,DectoBCD (tempDay));
239         outb(0x70,0x08);
240         outb(0x71,DectoBCD (tempMonth));
241         outb(0x70,0x32);
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()

```
void SetTime (
             int hours,
             int minutes,
             int seconds )
```

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.

```
137                                                     {
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);
143     unsigned char tempSeconds = BCDtoDec(inb(0x71));
144         cli(); //outb(device + 1, 0x00); //disable interrupts
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);
150         outb(0x71, DectoBCD(seconds));
151         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
152     outb(0x70,0x04);
153     unsigned char newHours = BCDtoDec(inb(0x71));
154     outb(0x70,0x02);
155     unsigned char newMinutes = BCDtoDec(inb(0x71));
156     outb(0x70,0x00);
157     unsigned char newSeconds = BCDtoDec(inb(0x71));
158     if(newHours != hours || newMinutes != minutes || newSeconds != seconds){
159       printf("Your input was invalid\n");
160       cli(); //outb(device + 1, 0x00); //disable interrupts
161         outb(0x70,0x04);
162         outb(0x71, DectoBCD(tempHours));// change to bcd
163         outb(0x70,0x02);
164         outb(0x71, DectoBCD(tempMinutes));
165         outb(0x70,0x00);
166         outb(0x71, DectoBCD(tempSeconds));
167         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
168     }
169     else
170       printf("Time Set\n");
171     }
```

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.

```
535                 {
536     Show_Ready();
537     Show_Blocked();
538 }
```

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)  {
```

```
681           printf("\x1b[32m""\n The Blocked Queue is empty \n""\x1b[0m");
682       }
683       else    {
684           int class, check, state, prior, status;
685           char name[20];
686           char block[] = "\x1B[34m""Blocked Queue: \n""\x1b[0m";
687           char cname[] = "Name: ";
688           char cclass[] = "Class: ";
689           char cstate[] = "State: ";
690           char cstatus[] = "Status: ";
691           char cprior[] = "Priority: ";
692           char line[] = "\n";
693           check = 15;
694
695           sys_req(WRITE, COM1, block, &check );
696
697           PCB* pcb = getBlocked()->head;
698
699           if(pcb->next == NULL) {
700             class = pcb->Process_Class;
701                 strcpy(name,pcb->Process_Name);
702                 state = pcb->ReadyState;
703                 status = pcb->SuspendedState;
704                 prior = pcb->Priority;
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                 }
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);
729                 if(pcb->SuspendedState == 0)  {
730                   printf("0");
731                 }
732                 else  {
733                   sys_req(WRITE, COM1, itoa(status), &check);
734                 }
735                 printf(line);
736
737                 printf(cprior);
738                 if(pcb->Priority == 0)  {
739                   printf("0");
740                   printf("\n\n");
741                 }
742                 else  {
743                   sys_req(WRITE, COM1, itoa(prior), &check);
744                   printf("\n\n");
745                 }
746           }
747           else  {
748             while(pcb != NULL)  {
749                 class = pcb->Process_Class;
750                     strcpy(name,pcb->Process_Name);
751                     state = pcb->ReadyState;
752                     status = pcb->SuspendedState;
753                     prior = pcb->Priority;
754
755                     printf(cname);
756                     printf(name);
757                     printf(line);
758
759                     printf(cclass);
760                     if(pcb->Process_Class == 0)  {
761                       printf("0");
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 
786                     printf(cprior);
787                     if(pcb->Priority == 0)  {
788                       printf("0");
789                       printf("\n\n");
790                     }
791                     else  {
792                       sys_req(WRITE, COM1, itoa(prior), &check);
793                       printf("\n\n");
794                     }
795                 pcb = pcb->next;
796             }
797         }
798     }
799 }
```

4.21.1.17 Show_PCB()

```
void Show_PCB (
              char * ProcessName )
```

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

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

```
547                   {
548     if(getReady()->head == NULL)    {
549          printf("\x1b[32m""\n The Ready Queue is empty \n""\x1b[0m");
550
551     }
552     else    {
553
554           int class, check, state, prior, status;
555           char name[10];
556           char ready[] = "\x1B[34m""\nReady Queue:\n""\x1B[0m";
557           char cname[] = "Name: ";
558           char cclass[] = "Class: ";
559           char cstate[] = "State: ";
560           char cstatus[] = "Status: ";
561           char cprior[] = "Priority: ";
562           char line[] = "\n";
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;
571                 strcpy(name,pcb->Process_Name);
```

```
572                 state = pcb->ReadyState;
573                 status = pcb->SuspendedState;
574                 prior = pcb->Priority;
575 
576                 printf(cname);
577                 printf(name);
578                 printf(line);
579 
580                 printf(cclass);
581                 if(pcb->Process_Class == 0)  {
582                   printf("0");
583                 }
584                 else  {
585                   sys_req(WRITE, COM1, itoa(class), &check);
586                 }
587                 printf(line);
588 
589                 printf(cstate);
590                 if(pcb->ReadyState == 0)  {
591                   printf("0");
592                 }
593                 else  {
594                   sys_req(WRITE, COM1, itoa(state), &check);
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 
607                 printf(cprior);
608                 if(pcb->Priority == 0)  {
609                   printf("0");
610                   printf("\n\n");
611                 }
612                 else  {
613                   sys_req(WRITE, COM1, itoa(prior), &check);
614                   printf("\n\n");
615                 }
616           }
617           else  {
618             while(pcb != NULL)  {
619                 class = pcb->Process_Class;
620                     strcpy(name,pcb->Process_Name);
621                     state = pcb->ReadyState;
622                     status = pcb->SuspendedState;
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);
639                     if(pcb->ReadyState == 0)  {
640                       printf("0");
641                     }
642                     else  {
643                       sys_req(WRITE, COM1, itoa(state), &check);
644                     }
645                     printf(line);
646 
647                     printf(cstatus);
648                     if(pcb->SuspendedState == 0)  {
649                       printf("0");
650                     }
651                     else  {
652                       sys_req(WRITE, COM1, itoa(status), &check);
653                     }
654                     printf(line);
655 
656                     printf(cprior);
657                     if(pcb->Priority == 0)  {
658                       printf("0");
```

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

4.21.1.19 Suspend()

```
void Suspend (
             char * ProcessName )
```

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       else {
397         if(pcb->SuspendedState == YES)  {
398             printf("\x1b[32m""\nThis Process is already SUSPENDED \n""\x1b[0m");
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.

```
291                                 {
292         if((c >= 65) && (c <= 90))  {
293                 c = c + 32;
294             }
295         return c;
296         }
```

4.21.1.21 Unblock()

```
void Unblock (
             char * ProcessName )
```

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

```
int BCDtoDec (
            int 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.

```
69                                {
70          return (((BCD»4)*10) + (BCD & 0xF));
71      }
```

4.22.1.2 Block()

```
void Block (
              char * ProcessName )
```

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.

```
858                                    {
859    PCB* pcb = FindPCB(ProcessName);
860    if (pcb == NULL)  {
861      printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
862    }
863    else {
864      if(pcb->ReadyState == BLOCKED)  {
865          printf("\x1b[32m""\nThis Process is already BLOCKED \n""\x1b[0m");
866      }
867      else    {
868        RemovePCB(pcb);
869        pcb->ReadyState = BLOCKED;
870        InsertPCB(pcb);
871      }
872    }
873 }
```

4.22.1.3 Create_PCB()

```
void Create_PCB (
              char * ProcessName,
              int Priority,
              int Class )
```

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.

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

Decimal	Decimal number to be changed to binary

Definition at line 76 of file userFunctions.c.

```
76                                  {
77          return (((Decimal/10) << 4) | (Decimal % 10));
78      }
```

4.22.1.5 Delete_PCB()

```
void Delete_PCB (
              char * ProcessName )
```

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

4.22.1.6 EdgeCase()

```
int EdgeCase (
              char * pointer )
```

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.

```
97                                    {
98     int valid = 0;
99     if (strcmp(pointer, "00") == 0) {
100       valid = 1;
101       return valid;
102     }
103     else if (strcmp(pointer, "0") == 0) {
104       valid = 1;
105       return valid;
106     }
107     else  {
108     int j;
109         valid = 0;
110         for(j = 0; j <= 99; j++)  {
111             if(strcmp(pointer,itoa(j)) == 0)
112                 valid = 1;
113         }
114         if(valid == 0)  {
115           return valid;
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);
```

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

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;
181         outb(0x70,0x04);
182         unsigned char hours = inb(0x71);
183         outb(0x70,0x02);
184         unsigned char minutes = inb(0x71);
185         outb(0x70,0x00);
186         unsigned char seconds = inb(0x71);
187         char msg1[2] = ":";
188         char msg2[10] = "Time: ";
189         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);
196         second = BCDtoDec(seconds);
197         sys_req(WRITE, COM1, itoa(second), &check);
198       printf("\n");
199     }
```

4.22.1.9 Help()

```
void Help (
              char * request )
```

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.

```
308                             {
309         if (request[0] == '\0') {
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
    setPriority \n showPCB \n showAll \n showReady \n showBlocked \n createPCB \n deletePCB \n block \n
    unblock \n shutdown \n\n");
312         }
313         else if (strcmp(request, "GetDate") == 0) {
314                 printf("\n getDate returns the current date that is loaded onto the operating
    system.\n");
315         }
316         else if (strcmp(request, "SetDate") == 0) {
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         }
319         else if (strcmp(request, "GetTime") == 0) {
320                 printf("\n getTime returns the current time as hours, minutes, seconds that is loaded
    onto the operating system.\n");
321         }
322         else if (strcmp(request, "SetTime") == 0) {
323                 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
    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         }
328      else if(strcmp(request, "shutdown") == 0) {
329        printf("\n shutdown shuts down the system.\n");
330      }
331
332
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      }
339      else if(strcmp(request,"resume") == 0) {
340      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) {
343      printf("\n SetPriority takes in the name of a PCB and the priority (setPrioriry-NAME-PRIORITY) it
    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      }
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      }
351      else if(strcmp(request,"showReady") == 0) {
352      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) {
355      printf("\n 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) {
366     printf("\n Block takes in the process_name (block-NAME) then sets it's state to blocked and
        reinserts it back into the correct queue.\n");
367     }
368     else if(strcmp(request,"unblock") == 0) {
369     printf("\n Unblock takes in the process_name (unblock-NAME) then sets it's state to ready and
        reinserts it back into the correct queue.\n");
370     }
371     else    {
372      printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
        full list of commands.\n""\x1b[0m");
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.

```
39     {
40             int i,j,k,count;
41             i = num;
42             j = 0;
43             count = 0;
44         while(i){ // count number of digits
45             count++;
46             i /= 10;
47         }
48
49         char* arr1;
50         char arr2[count];
51         arr1 = (char*)sys_alloc_mem(count); //memory allocation
52
53         while(num){ // seperate last digit from number and add ASCII
54             arr2[++j] = num%10 + '0';
55             num /= 10;
56         }
57
58         for(k = 0; k < j; k++){ // reverse array results
59             arr1[k] = arr2[j-k];
60         }
61         arr1[k] = '\0';
62
63         return(char*)arr1;
64     }
```

4.22.1.11 Resume()

```
void Resume (
              char * ProcessName )
```

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       }
421       else {
422         if(pcb->SuspendedState == NO)   {
423             printf(GRN"\nThis Process is already in the NONSUSPENDED state \n"RESET);
424         }
425         else    {
426             pcb->SuspendedState = NO;
427         }
428       }
429 }
```

4.22.1.12 Set_Priority()

```
void Set_Priority (
             char * ProcessName,
             int 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);
443       if (pcb == NULL)    {
444         printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
445       }
446       else if(Priority >= 10){
447         printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
448       }
449       else {
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 SetDime(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);
209     int tempDay = BCDtoDec(inb(0x71));
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();
217         outb(0x70,0x07);
218         outb(0x71,DectoBCD (day));
219         outb(0x70,0x08);
220         outb(0x71,DectoBCD (month));
221         outb(0x70,0x32);
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);
233     unsigned char newYear = BCDtoDec(inb(0x71));
234     if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
235       printf("Your input was invalid\n");
236       cli();
237         outb(0x70,0x07);
238         outb(0x71,DectoBCD (tempDay));
239         outb(0x70,0x08);
240         outb(0x71,DectoBCD (tempMonth));
241         outb(0x70,0x32);
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     }
```

4.22.1.14 SetTime()

```
void SetTime (
            int hours,
            int minutes,
            int seconds )
```

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.

```
137                                                     {
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);
143     unsigned char tempSeconds = BCDtoDec(inb(0x71));
144         cli(); //outb(device + 1, 0x00); //disable interrupts
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);
150         outb(0x71, DectoBCD(seconds));
151         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
152     outb(0x70,0x04);
153     unsigned char newHours = BCDtoDec(inb(0x71));
154     outb(0x70,0x02);
155     unsigned char newMinutes = BCDtoDec(inb(0x71));
156     outb(0x70,0x00);
157     unsigned char newSeconds = BCDtoDec(inb(0x71));
158     if(newHours != hours || newMinutes != minutes || newSeconds != seconds){
159       printf("Your input was invalid\n");
160       cli(); //outb(device + 1, 0x00); //disable interrupts
161         outb(0x70,0x04);
162         outb(0x71, DectoBCD(tempHours));// change to bcd
163         outb(0x70,0x02);
164         outb(0x71, DectoBCD(tempMinutes));
165         outb(0x70,0x00);
166         outb(0x71, DectoBCD(tempSeconds));
167         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
168     }
169     else
170       printf("Time Set\n");
171     }
```

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.

```
535                 {
536     Show_Ready();
537     Show_Blocked();
538 }
```

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.

```
679                     {
680     if(getBlocked()->head == NULL)  {
681         printf("\x1b[32m""\n The Blocked Queue is empty \n""\x1b[0m");
682     }
683     else    {
684           int class, check, state, prior, status;
685           char name[20];
686           char block[] = "\x1B[34m""Blocked Queue: \n""\x1b[0m";
687           char cname[] = "Name: ";
688           char cclass[] = "Class: ";
689           char cstate[] = "State: ";
690           char cstatus[] = "Status: ";
691           char cprior[] = "Priority: ";
692           char line[] = "\n";
693           check = 15;
694
695           sys_req(WRITE, COM1, block, &check );
696
697           PCB* pcb = getBlocked()->head;
698
699           if(pcb->next == NULL) {
700             class = pcb->Process_Class;
701                 strcpy(name,pcb->Process_Name);
702                 state = pcb->ReadyState;
703                 status = pcb->SuspendedState;
704                 prior = pcb->Priority;
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                 }
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);
729                 if(pcb->SuspendedState == 0)  {
730                   printf("0");
731                 }
732                 else  {
733                   sys_req(WRITE, COM1, itoa(status), &check);
734                 }
735                 printf(line);
736
737                 printf(cprior);
738                 if(pcb->Priority == 0)  {
739                   printf("0");
740                   printf("\n\n");
741                 }
742                 else  {
743                   sys_req(WRITE, COM1, itoa(prior), &check);
744                   printf("\n\n");
745                 }
746           }
747           else  {
748             while(pcb != NULL)  {
749                 class = pcb->Process_Class;
750                     strcpy(name,pcb->Process_Name);
```

```
751                     state = pcb->ReadyState;
752                     status = pcb->SuspendedState;
753                     prior = pcb->Priority;
754 
755                     printf(cname);
756                     printf(name);
757                     printf(line);
758 
759                     printf(cclass);
760                     if(pcb->Process_Class == 0)  {
761                       printf("0");
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 
786                     printf(cprior);
787                     if(pcb->Priority == 0)  {
788                       printf("0");
789                       printf("\n\n");
790                     }
791                     else  {
792                       sys_req(WRITE, COM1, itoa(prior), &check);
793                       printf("\n\n");
794                     }
795                 pcb = pcb->next;
796             }
797         }
798     }
799 }
```

4.22.1.17 Show_PCB()

```
void Show_PCB (
             char * ProcessName )
```

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                                     {
465     if (FindPCB(ProcessName) == NULL)   {
466          printf("\x1b[31m""\nERROR: PCB does not exist \n""\x1b[0m");
467     }
468     else    {
469           int check = 5;
```

```
470           char name[10];
471           char cname[] = "Name: ";
472           char cclass[] = "Class: ";
473           char cstate[] = "State: ";
474           char cstatus[] = "Status: ";
475           char cprior[] = "Priority: ";
476           char line[] = "\n";
477           PCB* pcb = FindPCB(ProcessName);
478           strcpy(name,pcb->Process_Name);
479           int class = pcb->Process_Class;
480           int state = pcb->ReadyState;
481           int status = pcb->SuspendedState;
482           int prior = pcb->Priority;
483
484           if(name == NULL){
485                 printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
486           }
487           else  {
488                 printf(cname);
489             printf(ProcessName);
490             printf(line);
491             printf(cclass);
492               if(pcb->Process_Class == 0)  {
493                 printf("0");
494               }
495               else  {
496                 sys_req(WRITE, COM1, itoa(class), &check);
497               }
498                 printf(line);
499             printf(cstate);
500               if(pcb->ReadyState == 0)  {
501                 printf("0");
502               }
503               else  {
504                 sys_req(WRITE, COM1, itoa(state), &check);
505               }
506                 printf(line);
507             printf(cstatus);
508               if(pcb->SuspendedState == 0)  {
509                 printf("0");
510               }
511               else  {
512                 sys_req(WRITE, COM1, itoa(status), &check);
513               }
514                 printf(line);
515             printf(cprior);
516               if(pcb->Priority == 0)  {
517                 printf("0");
518                 printf("\n\n");
519               }
520               else  {
521                 sys_req(WRITE, COM1, itoa(prior), &check);
522                 printf("\n\n");
523               }
524         }
525     }
526 }
```

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.

```
547                   {
548     if(getReady()->head == NULL)     {
549          printf("\x1b[32m""\n The Ready Queue is empty \n""\x1b[0m");
550
551     }
552     else     {
553
554           int class, check, state, prior, status;
```

```
555           char name[10];
556           char ready[] = "\x1B[34m""\nReady Queue:\n""\x1B[0m";
557           char cname[] = "Name: ";
558           char cclass[] = "Class: ";
559           char cstate[] = "State: ";
560           char cstatus[] = "Status: ";
561           char cprior[] = "Priority: ";
562           char line[] = "\n";
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;
571                 strcpy(name,pcb->Process_Name);
572                 state = pcb->ReadyState;
573                 status = pcb->SuspendedState;
574                 prior = pcb->Priority;
575
576                 printf(cname);
577                 printf(name);
578                 printf(line);
579
580                 printf(cclass);
581                 if(pcb->Process_Class == 0)  {
582                   printf("0");
583                 }
584                 else  {
585                   sys_req(WRITE, COM1, itoa(class), &check);
586                 }
587                 printf(line);
588
589                 printf(cstate);
590                 if(pcb->ReadyState == 0)  {
591                   printf("0");
592                 }
593                 else  {
594                   sys_req(WRITE, COM1, itoa(state), &check);
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
607                 printf(cprior);
608                 if(pcb->Priority == 0)  {
609                   printf("0");
610                   printf("\n\n");
611                 }
612                 else  {
613                   sys_req(WRITE, COM1, itoa(prior), &check);
614                   printf("\n\n");
615                 }
616           }
617           else  {
618             while(pcb != NULL)  {
619                 class = pcb->Process_Class;
620                     strcpy(name,pcb->Process_Name);
621                     state = pcb->ReadyState;
622                     status = pcb->SuspendedState;
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);
639                     if(pcb->ReadyState == 0)  {
640                       printf("0");
641                     }
```

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

4.22.1.19 Suspend()

```
void Suspend (
             char * ProcessName )
```

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       else {
397         if(pcb->SuspendedState == YES)  {
398             printf("\x1b[32m""\nThis Process is already SUSPENDED \n""\x1b[0m");
399         }
400         else    {
401             pcb->SuspendedState = YES;
402         }
403       }
404 
405 }
```

4.22.1.20 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.

```
291                                 {
292         if((c >= 65) && (c <= 90))  {
293                 c = c + 32;
294             }
295         return c;
296         }
```

4.22.1.21 Unblock()

```
void Unblock (
              char * ProcessName )
```

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