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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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:

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/R2/PCB.h

Chapter 4

File Documentation

4.1 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/include/core/asm.h File Reference

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

4.2 D:/GITHUB/CS_450_RunTime_Terror/mpx_↩ core/include/core/interrupts.h File Reference

Functions

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

4.3 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_↩ core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/include/string.h File Reference

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

Functions

- int isspace (const char *c)

  Description: Determine if a character is whitespace.
- void * memset (void *s, int c, size_t n)

  Description: Set a region of memory.
- char * strcpy (char *s1, const char *s2)

  Description: Copy one string to another.
- char * strcat (char *s1, const char *s2)

  Description: Concatenate the contents of one string onto another.
- int strlen (const char *s)

  Description: Returns the length of a string.
- int strcmp (const char *s1, const char *s2)

  Description: String comparison.
- char * strtok (char *s1, const char *s2)

  Description: Split string into tokens.
- int atoi (const char *s)

  Description: Convert an ASCII string to an integer.

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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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"
#include "modules/R2/PCB.h"
```

Functions

- void **kmain** (void)

4.12 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_↩ core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/lib/string.c File Reference

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

Functions

- int strlen (const char *s)

  Description: Returns the length of a string.
- char * strcpy (char *s1, const char *s2)

  Description: Copy one string to another.
- int atoi (const char *s)

  Description: Convert an ASCII string to an integer.
- int strcmp (const char *s1, const char *s2)

  Description: String comparison.
- char * strcat (char *s1, const char *s2)

  Description: Concatenate the contents of one string onto another.
- int isspace (const char *c)

  Description: Determine if a character is whitespace.
- void * memset (void *s, int c, size_t n)

  Description: Set a region of memory.
- char * strtok (char *s1, const char *s2)

  Description: Split string into tokens.

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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/R1/comHand.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 "userFunctions.h"
```

Functions

- int comHand ()

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

4.21 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/R1/comHand.h File Reference

Functions

- int comHand ()

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

4.22 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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)

  Description: An integer is taken and seperated into individual chars and then all placed into a character array.
- int BCDtoDec (int BCD)

  Description: Changes binary number to decimal numbers.
- int DectoBCD (int Decimal)

  Description: Changes decimal numbers to binary numbers.
- void **printf** (char msg[ ])
- int EdgeCase (char *pointer)

  Description: Compares pointer char to validate if it is a number or not.
- 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).
- void GetTime ()

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

  Description: Returns the full date back to the user in decimal form.
- void Version ()

  Description: Simply returns a char containing "Version: R(module).
- char toLowercase (char c)

  Description: If a letter is uppercase, it changes it to lowercase.
- void Help (char *request)

  Brief Description: Gives helpful information for one of the functions.
- void Suspend (char *ProcessName)

  Brief Description: Places a PCD in the suspended state and reinserts it into the appropriate queue.
- void Resume (char *ProcessName)

  Brief Description: Places a PCD in the not suspended state and reinserts it into the appropriate queue.
- void Set_Priority (char *ProcessName, int Priority)

  Brief Description: Sets PCB priority and reinserts the process into the correct place in the correct queue.
- void Show_PCB (char *ProcessName)

  Brief Description: Displays the process name, class, state, suspended status, and priority of a PCB.
- void Show_All ()

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

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

  Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the blocked queue.
- void Create_PCB (char *ProcessName, int Priority, int Class)

  Brief Description: Calls SetupPCB() and inserts PCB into appropriate queue.
- void Delete_PCB (char *ProcessName)

  Brief Description: Removes PCB from appropriate queue and frees all associated memory.
- void Block (char *ProcessName)

  Brief Description: Places a PCD in the blocked state and reinserts it into the correct queue.
- void Unblock (char *ProcessName)

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

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 65 of file userFunctions.c.

```
65                              {
66          return (((BCD»4)*10) + (BCD & 0xF));
67      }
```

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 696 of file userFunctions.c.

```
696                                 {
697    // Name Error check
698    // Error check (Valid Name)
699    PCB* pcb = FindPCB(ProcessName);
700    if (pcb == NULL)  {
701      printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
702    }
703    else {
704      if(pcb->ReadyState == BLOCKED)  {
705          printf("\x1b[32m""\nThis Process is already BLOCKED \n""\x1b[0m");
706      }
707      else    {
708          pcb->ReadyState = BLOCKED;
709      }
710    }
711 }
```

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 658 of file userFunctions.c.

```
658                                                                    {
659   if (FindPCB(ProcessName) == NULL) {
660     if(Priority < 0 && Priority < 10){
661       if(Class == 0 || Class == 1){
662         PCB* pcb = SetupPCB(ProcessName, Class, Priority);
663         InsertPCB(pcb);
664       } else{
665         printf("\x1b[31m""\nERROR: Not a valid Class \n""\x1b[0m");
666       }
667     } else{
668       printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
669     }
670   } else{
671     printf("\x1b[31m""\nERROR: Not a valid Process Name \n""\x1b[0m");
672   }
673 }
```

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 72 of file userFunctions.c.

```
72                                  {
73          return (((Decimal/10) << 4) | (Decimal % 10));
74      }
```

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 680 of file userFunctions.c.

```
680                                         {
681   PCB* pcb = FindPCB(ProcessName);
682   if (pcb == NULL)  {
683     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
684   }
685   else {
686      RemovePCB(pcb);
687      FreePCB(pcb);
688   }
689 }
```

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 84 of file userFunctions.c.

```
84                                      {
85     int valid = 0;
86     if (strcmp(pointer, "00") == 0){
87       valid = 1;
88       return valid;
89     }
90     int i, j;
91     for (i = 0; i < strlen(pointer); i++){
92         valid = 0;
93         for(j = 0; j <= 99; j++){
94             if(strcmp(pointer,itoa(j)) == 0)
95                 valid = 1;
96         }
97         if(valid == 0){
98           return valid;
99         }
100     }
101     return valid;
102   }
```

4.22.1.7 GetDate()

```
void GetDate ( )
```

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

No parameters.

Definition at line 226 of file userFunctions.c.

```
226                     {
227       int check = 2;
228       outb(0x70,0x07);
229         unsigned char day = BCDtoDec(inb(0x71));
230         outb(0x70,0x08);
231         unsigned char month = BCDtoDec(inb(0x71));
232         outb(0x70,0x32);
233         unsigned char millennium = BCDtoDec(inb(0x71));
234         char msg[2] = "-";
235         char msg3[10] = "Date: ";
236         printf(msg3);
237         sys_req(WRITE, COM1, itoa(day), &check);
238         printf(msg);
239         sys_req(WRITE, COM1, itoa(month), &check);
240         printf(msg);
241         sys_req(WRITE, COM1, itoa(millennium), &check);
242     outb(0x70,0x09);
243     if(BCDtoDec(inb(0x71)) == 0){
244       sys_req(WRITE, COM1, "00", &check);
245     }
246     else {
247         unsigned char year = BCDtoDec(inb(0x71));
248         sys_req(WRITE, COM1, itoa(year), &check);
249     }
250         printf("\n");
251     }
```

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 148 of file userFunctions.c.

```
148                     {
149     int check = 2;
150     int hour;
151     int minute;
152     int second;
153         outb(0x70,0x04);
154         unsigned char hours = inb(0x71);
155         outb(0x70,0x02);
156         unsigned char minutes = inb(0x71);
157         outb(0x70,0x00);
158         unsigned char seconds = inb(0x71);
159         char msg1[2] = ":";
160         char msg2[10] = "Time: ";
161         printf(msg2);
162         hour = BCDtoDec(hours);
163         sys_req(WRITE, COM1, itoa(hour), &check);
164         printf(msg1);
165         minute = BCDtoDec(minutes);
166         sys_req(WRITE, COM1, itoa(minute), &check);
167         printf(msg1);
168         second = BCDtoDec(seconds);
169         sys_req(WRITE, COM1, itoa(second), &check);
170       printf("\n");
171     }
```

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 275 of file userFunctions.c.

```
275                         {
276         if (request[0] == '\0') {
277                 printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
278              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");
279         }
280         else if (strcmp(request, "GetDate") == 0) {
281                 printf("\n getDate returns the current date that is loaded onto the operating
      system.\n");
282         }
283         else if (strcmp(request, "SetDate") == 0) {
284                 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");
285         }
286         else if (strcmp(request, "GetTime") == 0) {
287                 printf("\n getTime returns the current time as hours, minutes, seconds that is loaded
      onto the operating system.\n");
288         }
289         else if (strcmp(request, "SetTime") == 0) {
290                 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");
291              }
292         else if (strcmp(request, "Version") == 0) {
293                 printf("\n version returns the current operating software version that the system is
      running.\n");
294         }
295      else if(strcmp(request, "shutdown") == 0) {
296        printf("\n shutdown shuts down the system.\n");
297      }
298
299 /************************R2 Commands*********************************************************/
300      else if(strcmp(request,"suspend") == 0) {
301      printf("\n Suspend takes in the name of a PCB then places it into the suspended state and reinserts
      it into the correct queue.\n");
302      }
303      else if(strcmp(request,"resume") == 0) {
304      printf("\n Resume takes in the name of a PCB then removes it from the suspended state and adds it to
      the correct queue.\n");
305      }
306      else if(strcmp(request,"setPriority") == 0) {
307      printf("\n SetPriority takes in the name of a PCB and the priority it needs to be set to then
      reinstates the specified PCB into a new location by priority.\n");
308      }
309      else if(strcmp(request,"showPCB") == 0) {
310      printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
      user.\n");
311      }
312      else if(strcmp(request,"showAll") == 0) {
313      printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
      queues.\n");
314      }
315      else if(strcmp(request,"showReady") == 0) {
316      printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
      currently are in the ready state.\n");
317      }
318      else if(strcmp(request,"showBlocked") == 0) {
```

```
319     printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
       currently are in the blocked state.\n");
320     }
321
322 /******************************** R2 Temp Commands
       *********************************************************/
323     else if(strcmp(request,"createPCB") == 0) {
324     printf("\n  CreatePCB takes in the process_name, process_class, and process_priority. Then assigns
       this new process into the correct queue.\n");
325     }
326     else if(strcmp(request,"deletePCB") == 0) {
327     printf("\n DeletePCB takes in the process_name then deletes it from the queue and free's all the
       memory that was previously allocated to the specified PCB.\n");
328     }
329     else if(strcmp(request,"block") == 0) {
330     printf("\n Block takes in the process_name then sets it's state to blocked and reinserts it back
       into the correct queue.\n");
331     }
332     else if(strcmp(request,"unblock") == 0) {
333     printf("\n Unblock takes in the process_name then sets it's state to ready and reinserts it back
       into the correct queue.\n");
334     }
335     else    {
336      printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
       full list of commands.\n""\x1b[0m");
337     }
338 }
```

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 34 of file userFunctions.c.

```
35     {
36             int i,j,k,count;
37             i = num;
38             j = 0;
39             count = 0;
40         while(i){ // count number of digits
41             count++;
42             i /= 10;
43         }
44
45         char* arr1;
46         char arr2[count];
47         arr1 = (char*)sys_alloc_mem(count); //memory allocation
48
49         while(num){ // seperate last digit from number and add ASCII
50             arr2[++j] = num%10 + '0';
51             num /= 10;
52         }
53
54         for(k = 0; k < j; k++){ // reverse array results
55             arr1[k] = arr2[j-k];
56         }
57         arr1[k] = '\0';
58
59         return(char*)arr1;
60     }
```

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 379 of file userFunctions.c.

```
379                                 {
380   // Name Error check
381   // Error check (Valid Name)
382   PCB* pcb = FindPCB(ProcessName);
383   if (pcb == NULL)  {
384     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
385   }
386   else {
387     if(pcb->SuspendedState == NO)   {
388         printf("\x1b[32m""\nThis Process is already in the RUNNING state \n""\x1b[0m");
389     }
390     else    {
391         pcb->SuspendedState = NO;
392     }
393   }
394 }
```

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 402 of file userFunctions.c.

```
402                                                 {
403   PCB* pcb = FindPCB(ProcessName);
404   if (pcb == NULL)    {
405     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
406   } else if(Priority < 10){
407         printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
408   } else {
409     RemovePCB(pcb);
410     pcb->Priority = Priority;
```

```
411     InsertPCB(pcb);
412   }
413 }
```

4.22.1.13 SetDate()

```
void SetDate (
              int day,
              int month,
              int millennium,
              int year )
```

Description: Sets the date register to the new values that the user inputed, all values must be inputed as Set↩ Dime(day, month, millenial, year).

Parameters

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

Definition at line 179 of file userFunctions.c.

```
179                                                     {
180     outb(0x70,0x07);
181     int tempDay = BCDtoDec(inb(0x71));
182     outb(0x70,0x08);
183     int tempMonth = BCDtoDec(inb(0x71));
184     outb(0x70,0x32);
185     int tempMillennium = BCDtoDec(inb(0x71));
186     outb(0x70,0x09);
187     int tempYear = BCDtoDec(inb(0x71));
188     cli();
189         outb(0x70,0x07);
190         outb(0x71,DectoBCD (day));
191         outb(0x70,0x08);
192         outb(0x71,DectoBCD (month));
193         outb(0x70,0x32);
194         outb(0x71,DectoBCD (millennium));
195         outb(0x70,0x09);
196         outb(0x71,DectoBCD (year));
197         sti();
198     outb(0x70,0x07);
199     unsigned char newDay = BCDtoDec(inb(0x71));
200     outb(0x70,0x08);
201     unsigned char newMonth = BCDtoDec(inb(0x71));
202     outb(0x70,0x32);
203     unsigned char newMillennium = BCDtoDec(inb(0x71));
204     outb(0x70,0x09);
205     unsigned char newYear = BCDtoDec(inb(0x71));
206     if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
207       printf("Your input was invalid\n");
208       cli();
209         outb(0x70,0x07);
210         outb(0x71,DectoBCD (tempDay));
211         outb(0x70,0x08);
212         outb(0x71,DectoBCD (tempMonth));
213         outb(0x70,0x32);
214         outb(0x71,DectoBCD (tempMillennium));
215         outb(0x70,0x09);
216         outb(0x71,DectoBCD (tempYear));
217         sti();
218     }
219     else
220       printf("Date Set\n");
221     }
```

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 109 of file userFunctions.c.

```
109                                                 {
110     outb(0x70,0x04);
111     unsigned char tempHours = BCDtoDec(inb(0x71));
112     outb(0x70,0x02);
113     unsigned char tempMinutes = BCDtoDec(inb(0x71));
114     outb(0x70,0x00);
115     unsigned char tempSeconds = BCDtoDec(inb(0x71));
116         cli(); //outb(device + 1, 0x00); //disable interrupts
117         outb(0x70,0x04);
118         outb(0x71, DectoBCD(hours));// change to bcd
119         outb(0x70,0x02);
120         outb(0x71, DectoBCD(minutes));
121         outb(0x70,0x00);
122         outb(0x71, DectoBCD(seconds));
123         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
124     outb(0x70,0x04);
125     unsigned char newHours = BCDtoDec(inb(0x71));
126     outb(0x70,0x02);
127     unsigned char newMinutes = BCDtoDec(inb(0x71));
128     outb(0x70,0x00);
129     unsigned char newSeconds = BCDtoDec(inb(0x71));
130     if(newHours != hours || newMinutes != minutes || newSeconds != seconds){
131       printf("Your input was invalid\n");
132       cli(); //outb(device + 1, 0x00); //disable interrupts
133         outb(0x70,0x04);
134         outb(0x71, DectoBCD(tempHours));// change to bcd
135         outb(0x70,0x02);
136         outb(0x71, DectoBCD(tempMinutes));
137         outb(0x70,0x00);
138         outb(0x71, DectoBCD(tempSeconds));
139         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
140     }
141     else
142       printf("Time Set\n");
143     }
```

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 463 of file userFunctions.c.

```
463                 {
464   int class, check, state, prior, status, i,j;
465   char name[10];
466   char ready[] = "Ready Queue:\n";
467   char block[] = "Blocked Queue: \n";
468   char cname[] = "Name: ";
469   char cclass[] = "Class: ";
470   char cstate[] = "State: ";
471   char cstatus[] = "Status: ";
472   char cprior[] = "Priority: ";
473   char line[] = "\n";
474   char dline[] = "\n\n";
475   check = 15;
476 
477   sys_req(WRITE, COM1, ready, &check );
478 
479   if(ReadyQueue.head != NULL)
480     PCB* pcb = ReadyQueue.head;
481 
482   do {
483     if(pcb != ReadyQueue.head)
484       pcb = pcb.next;
485 
486     class = pcb->Process_Class;
487     strcpy(name,pcb->Process_Name);
488     state = pcb->ReadyState;
489     status = pcb->SuspendedState;
490     prior = pcb->Priority;
491 
492     printf(cname);
493     printf(name);
494     printf(line);
495 
496     printf(cclass);
497     sys_req(WRITE, COM1, itoa(class), &check);
498     printf(line);
499 
500     printf(cstate);
501     sys_req(WRITE, COM1, itoa(state), &check);
502     printf(line);
503 
504     printf(cstatus);
505     sys_req(WRITE, COM1, itoa(status), &check);
506     printf(line);
507 
508     printf(cprior);
509     sys_req(WRITE, COM1, itoa(prior), &check);
510     printf(dline);
511 
512   } while(pcb.next != NULL);
513 
514   sys_req(WRITE, COM1, block, &check );
515 
516   if(BlockedQueue.head != NULL)
517     PCB* pcb = BlockedQueue.head;
518 
519   do {
520     if(pcb != BlockedQueue.head)
521       pcb = pcb.next;
522 
523     class = pcb->Process_Class;
524     strcpy(name,pcb->Process_Name);
525     state = pcb->ReadyState;
526     status = pcb->SuspendedState;
527     prior = pcb->Priority;
528 
529     printf(cname);
530     printf(name);
531     printf(line);
532 
533     printf(cclass);
534     sys_req(WRITE, COM1, itoa(class), &check);
535     printf(line);
536 
537     printf(cstate);
538     sys_req(WRITE, COM1, itoa(state), &check);
539     printf(line);
540 
541     printf(cstatus);
542     sys_req(WRITE, COM1, itoa(status), &check);
543     printf(line);
544 
545     printf(cprior);
546     sys_req(WRITE, COM1, itoa(prior), &check);
547   } while(pcb.next != NULL);
548 }
```

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 604 of file userFunctions.c.

```
604                     {
605   int class, check, state, prior, status, j;
606   char name[20];
607   char block[] = "Blocked Queue: \n";
608   char cname[] = "Name: ";
609   char cclass[] = "Class: ";
610   char cstate[] = "State: ";
611   char cstatus[] = "Status: ";
612   char cprior[] = "Priority: ";
613   char line[] = "\n";
614   check = 15;
615
616   sys_req(WRITE, COM1, block, &check );
617   if(BlockedQueue.head != NULL)
618     PCB* pcb = BlockedQueue.head;
619
620   do {
621     if(pcb != BlockedQueue.head)
622       pcb = pcb.next;
623
624     class = pcb->Process_Class;
625     strcpy(name,pcb->Process_Name);
626     state = pcb->ReadyState;
627     status = pcb->SuspendedState;
628     prior = pcb->Priority;
629
630     printf(cname);
631     printf(name);
632     printf(line);
633
634     printf(cclass);
635     sys_req(WRITE, COM1, itoa(class), &check);
636     printf(line);
637
638     printf(cstate);
639     sys_req(WRITE, COM1, itoa(state), &check);
640     printf(line);
641
642     printf(cstatus);
643     sys_req(WRITE, COM1, itoa(status), &check);
644     printf(line);
645
646     printf(cprior);
647     sys_req(WRITE, COM1, itoa(prior), &check);
648   } while(pcb.next != NULL);
649 }
```

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 420 of file userFunctions.c.

```
420                                     {
421   int check = 5;
422   char name[10];
423   char cname[] = "Name: ";
424   char cclass[] = "Class: ";
425   char cstate[] = "State: ";
426   char cstatus[] = "Status: ";
427   char cprior[] = "Priority: ";
428   char line[] = "\n";
429   PCB* pcb = FindPCB(ProcessName);
430   strcpy(name,pcb->Process_Name);
431   int class = pcb->Process_Class;
432   int state = pcb->ReadyState;
433   int status = pcb->SuspendedState;
434   int prior = pcb->Priority;
435   if(name == NULL){
436     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
437   } else{
438     printf(cname);
439     printf(ProcessName);
440     printf(line);
441
442     printf(cclass);
443     sys_req(WRITE, COM1, itoa(class), &check);
444     printf(line);
445
446     printf(cstate);
447     sys_req(WRITE, COM1, itoa(state), &check);
448     printf(line);
449
450     printf(cstatus);
451     sys_req(WRITE, COM1, itoa(status), &check);
452     printf(line);
453
454     printf(cprior);
455     sys_req(WRITE, COM1, itoa(prior), &check);
456   }
457 }
```

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 553 of file userFunctions.c.

```
553                       {
554   int class, check, state, prior, status, i;
555   char name[10];
556   char ready[] = "Ready Queue:\n";
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   if(ReadyQueue.head != NULL)
568     PCB* pcb = ReadyQueue.head;
569
570   do {
```

```
571     if(pcb != ReadyQueue.head)
572       pcb = pcb.next;
573
574     class = pcb->Process_Class;
575     strcpy(name,pcb->Process_Name);
576     state = pcb->ReadyState;
577     status = pcb->SuspendedState;
578     prior = pcb->Priority;
579
580     printf(cname);
581     printf(name);
582     printf(line);
583
584     printf(cclass);
585     sys_req(WRITE, COM1, itoa(class), &check);
586     printf(line);
587
588     printf(cstate);
589     sys_req(WRITE, COM1, itoa(state), &check);
590     printf(line);
591
592     printf(cstatus);
593     sys_req(WRITE, COM1, itoa(status), &check);
594     printf(line);
595
596     printf(cprior);
597     sys_req(WRITE, COM1, itoa(prior), &check);
598   } while(pcb.next != NULL);
599 }
```

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 356 of file userFunctions.c.

```
356                                 {
357   // Name Error check
358   // Error check (Valid Name)
359   PCB* pcb = FindPCB(ProcessName);
360   if (pcb == NULL)  {
361     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
362   }
363   else {
364     if(pcb->SuspendedState == YES)  {
365         printf("\x1b[32m""\nThis Process is already SUSPENDED \n""\x1b[0m");
366     }
367     else    {
368         pcb->SuspendedState = YES;
369     }
370   }
371
372 }
```

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 263 of file userFunctions.c.

```
263                                     {
264         if((c >= 65) && (c <= 90))  {
265                 c = c + 32;
266             }
267         return c;
268         }
```

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 718 of file userFunctions.c.

```
718                                     {
719   PCB* pcb = FindPCB(ProcessName);
720   if (pcb == NULL)  {
721     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
722   }
723   else {
724     if(pcb->ReadyState == READY)    {
725         printf("\x1b[32m""\nThis Process is already in the READY state \n""\x1b[0m");
726     }
727     else    {
728         pcb->ReadyState = READY;
729     }
730   }
731 }
```

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 256 of file userFunctions.c.

```
256                     {
257         printf("Version: R2.0 \n");
258     }
```

4.23 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/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)

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

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

  Description: Changes decimal numbers to binary numbers.
- char * itoa (int num)

  Description: An integer is taken and seperated into individual chars and then all placed into a character array.
- 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).
- int BCDtoDec (int BCD)

  Description: Changes binary number to decimal numbers.
- void GetDate ()

  Description: Returns the full date back to the user in decimal form.
- void Version ()

  Description: Simply returns a char containing "Version: R(module).
- void Help (char *request)

  Brief Description: Gives helpful information for one of the functions.
- void **printf** (char msg[ ])
- int EdgeCase (char *pointer)

  Description: Compares pointer char to validate if it is a number or not.
- char toLowercase (char c)

  Description: If a letter is uppercase, it changes it to lowercase.
- void Suspend (char *ProcessName)

  Brief Description: Places a PCD in the suspended state and reinserts it into the appropriate queue.
- void Resume (char *ProcessName)

  Brief Description: Places a PCD in the not suspended state and reinserts it into the appropriate queue.
- void Set_Priority (char *ProcessName, int Priority)

  Brief Description: Sets PCB priority and reinserts the process into the correct place in the correct queue.
- void Show_PCB (char *ProcessName)

  Brief Description: Displays the process name, class, state, suspended status, and priority of a PCB.
- void Show_All ()

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

- void Show_Ready ()

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

  Brief Description: Displays the process name, class, state, suspended status, and priority of all PCB in the blocked queue.
- void Create_PCB (char *ProcessName, int Priority, int Class)

  Brief Description: Calls SetupPCB() and inserts PCB into appropriate queue.
- void Delete_PCB (char *ProcessName)

  Brief Description: Removes PCB from appropriate queue and frees all associated memory.
- void Block (char *ProcessName)

  Brief Description: Places a PCD in the blocked state and reinserts it into the correct queue.
- void Unblock (char *ProcessName)

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

4.23.1 Function Documentation

4.23.1.1 BCDtoDec()

```
int BCDtoDec (
            int BCD )
```

Description: Changes binary number to decimal numbers.

Parameters

value	Binary number to be changed to decimal

Definition at line 65 of file userFunctions.c.

```
65                              {
66          return (((BCD»4)*10) + (BCD & 0xF));
67      }
```

4.23.1.2 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 696 of file userFunctions.c.

```
696                                 {
697   // Name Error check
698   // Error check (Valid Name)
699   PCB* pcb = FindPCB(ProcessName);
700   if (pcb == NULL)  {
701     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
702   }
703   else {
704     if(pcb->ReadyState == BLOCKED)  {
705         printf("\x1b[32m""\nThis Process is already BLOCKED \n""\x1b[0m");
706     }
707     else    {
708         pcb->ReadyState = BLOCKED;
709     }
710   }
711 }
```

4.23.1.3 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 658 of file userFunctions.c.

```
658                                                                 {
659   if (FindPCB(ProcessName) == NULL) {
660     if(Priority < 0 && Priority < 10){
661       if(Class == 0 || Class == 1){
662         PCB* pcb = SetupPCB(ProcessName, Class, Priority);
663         InsertPCB(pcb);
664       } else{
665         printf("\x1b[31m""\nERROR: Not a valid Class \n""\x1b[0m");
666       }
667     } else{
668       printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
669     }
670   } else{
671     printf("\x1b[31m""\nERROR: Not a valid Process Name \n""\x1b[0m");
672   }
673 }
```

4.23.1.4 DectoBCD()

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

Description: Changes decimal numbers to binary numbers.

Parameters

Decimal	Decimal number to be changed to binary

Definition at line 72 of file userFunctions.c.

```
72                                  {
73          return (((Decimal/10) « 4) | (Decimal % 10));
74      }
```

4.23.1.5 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 680 of file userFunctions.c.

```
680                                     {
681    PCB* pcb = FindPCB(ProcessName);
682    if (pcb == NULL)  {
683      printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
684    }
685    else {
686       RemovePCB(pcb);
687       FreePCB(pcb);
688    }
689 }
```

4.23.1.6 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 84 of file userFunctions.c.

```
84                                  {
85     int valid = 0;
86     if (strcmp(pointer, "00") == 0){
87       valid = 1;
88       return valid;
89     }
90     int i, j;
91     for (i = 0; i < strlen(pointer); i++){
92         valid = 0;
93         for(j = 0; j <= 99; j++){
94             if(strcmp(pointer,itoa(j)) == 0)
95                 valid = 1;
96         }
97         if(valid == 0){
98           return valid;
99         }
100     }
101     return valid;
102   }
```

4.23.1.7 GetDate()

```
void GetDate ( )
```

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

No parameters.

Definition at line 226 of file userFunctions.c.

```
226                       {
227       int check = 2;
228       outb(0x70,0x07);
229         unsigned char day = BCDtoDec(inb(0x71));
230         outb(0x70,0x08);
231         unsigned char month = BCDtoDec(inb(0x71));
232         outb(0x70,0x32);
233         unsigned char millennium = BCDtoDec(inb(0x71));
234         char msg[2] = "-";
235         char msg3[10] = "Date: ";
236         printf(msg3);
237         sys_req(WRITE, COM1, itoa(day), &check);
238         printf(msg);
239         sys_req(WRITE, COM1, itoa(month), &check);
240         printf(msg);
241         sys_req(WRITE, COM1, itoa(millennium), &check);
242     outb(0x70,0x09);
243     if(BCDtoDec(inb(0x71)) == 0){
244       sys_req(WRITE, COM1, "00", &check);
245     }
246     else {
247         unsigned char year = BCDtoDec(inb(0x71));
248         sys_req(WRITE, COM1, itoa(year), &check);
249     }
250         printf("\n");
251     }
```

4.23.1.8 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 148 of file userFunctions.c.

```
148                     {
149     int check = 2;
150     int hour;
151     int minute;
152     int second;
153         outb(0x70,0x04);
154         unsigned char hours = inb(0x71);
155         outb(0x70,0x02);
156         unsigned char minutes = inb(0x71);
157         outb(0x70,0x00);
158         unsigned char seconds = inb(0x71);
159         char msg1[2] = ":";
160         char msg2[10] = "Time: ";
161         printf(msg2);
162         hour = BCDtoDec(hours);
163         sys_req(WRITE, COM1, itoa(hour), &check);
164         printf(msg1);
165         minute = BCDtoDec(minutes);
166         sys_req(WRITE, COM1, itoa(minute), &check);
167         printf(msg1);
168         second = BCDtoDec(seconds);
169         sys_req(WRITE, COM1, itoa(second), &check);
170       printf("\n");
171     }
```

4.23.1.9 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 275 of file userFunctions.c.

```
275                             {
276         if (request[0] == '\0') {
277                 printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
278             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");
279         }
280         else if (strcmp(request, "GetDate") == 0) {
281                 printf("\n getDate returns the current date that is loaded onto the operating
        system.\n");
282         }
283         else if (strcmp(request, "SetDate") == 0) {
284                 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");
285         }
286         else if (strcmp(request, "GetTime") == 0) {
287                 printf("\n getTime returns the current time as hours, minutes, seconds that is loaded
        onto the operating system.\n");
288         }
289         else if (strcmp(request, "SetTime") == 0) {
290                 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");
291             }
292         else if (strcmp(request, "Version") == 0) {
293                 printf("\n version returns the current operating software version that the system is
        running.\n");
```

```
294         }
295     else if(strcmp(request, "shutdown") == 0) {
296       printf("\n shutdown shuts down the system.\n");
297     }
298
299 /************************R2 Commands*******************************************************/
300     else if(strcmp(request,"suspend") == 0) {
301     printf("\n Suspend takes in the name of a PCB then places it into the suspended state and reinserts
      it into the correct queue.\n");
302     }
303     else if(strcmp(request,"resume") == 0) {
304     printf("\n Resume takes in the name of a PCB then removes it from the suspended state and adds it to
      the correct queue.\n");
305     }
306     else if(strcmp(request,"setPriority") == 0) {
307     printf("\n SetPriority takes in the name of a PCB and the priority it needs to be set to then
      reinstates the specified PCB into a new location by priority.\n");
308     }
309     else if(strcmp(request,"showPCB") == 0) {
310     printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
      user.\n");
311     }
312     else if(strcmp(request,"showAll") == 0) {
313     printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
      queues.\n");
314     }
315     else if(strcmp(request,"showReady") == 0) {
316     printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
      currently are in the ready state.\n");
317     }
318     else if(strcmp(request,"showBlocked") == 0) {
319     printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
      currently are in the blocked state.\n");
320     }
321
322 /******************************** R2 Temp Commands
      ****************************************************/
323     else if(strcmp(request,"createPCB") == 0) {
324     printf("\n  CreatePCB takes in the process_name, process_class, and process_priority. Then assigns
      this new process into the correct queue.\n");
325     }
326     else if(strcmp(request,"deletePCB") == 0) {
327     printf("\n DeletePCB takes in the process_name then deletes it from the queue and free's all the
      memory that was previously allocated to the specified PCB.\n");
328     }
329     else if(strcmp(request,"block") == 0) {
330     printf("\n Block takes in the process_name then sets it's state to blocked and reinserts it back
      into the correct queue.\n");
331     }
332     else if(strcmp(request,"unblock") == 0) {
333     printf("\n Unblock takes in the process_name then sets it's state to ready and reinserts it back
      into the correct queue.\n");
334     }
335     else    {
336      printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
      full list of commands.\n""\x1b[0m");
337     }
338 }
```

4.23.1.10 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 34 of file userFunctions.c.

```
35     {
36             int i,j,k,count;
37             i = num;
38             j = 0;
39             count = 0;
40         while(i){ // count number of digits
41             count++;
42             i /= 10;
43         }
44
45         char* arr1;
46         char arr2[count];
47         arr1 = (char*)sys_alloc_mem(count); //memory allocation
48
49         while(num){ // seperate last digit from number and add ASCII
50             arr2[++j] = num%10 + '0';
51             num /= 10;
52         }
53
54         for(k = 0; k < j; k++){ // reverse array results
55             arr1[k] = arr2[j-k];
56         }
57         arr1[k] = '\0';
58
59         return(char*)arr1;
60     }
```

4.23.1.11 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 379 of file userFunctions.c.

```
379                                 {
380   // Name Error check
381   // Error check (Valid Name)
382   PCB* pcb = FindPCB(ProcessName);
383   if (pcb == NULL)  {
384     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
385   }
386   else {
387     if(pcb->SuspendedState == NO)   {
388         printf("\x1b[32m""\nThis Process is already in the RUNNING state \n""\x1b[0m");
389     }
390     else    {
391         pcb->SuspendedState = NO;
392     }
393   }
394 }
```

4.23.1.12 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 402 of file userFunctions.c.

```
402                                                     {
403   PCB* pcb = FindPCB(ProcessName);
404   if (pcb == NULL)    {
405     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
406   } else if(Priority < 10){
407         printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
408   } else {
409     RemovePCB(pcb);
410     pcb->Priority = Priority;
411     InsertPCB(pcb);
412   }
413 }
```

4.23.1.13 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 179 of file userFunctions.c.

```
179                                                     {
180     outb(0x70,0x07);
181     int tempDay = BCDtoDec(inb(0x71));
182     outb(0x70,0x08);
183     int tempMonth = BCDtoDec(inb(0x71));
184     outb(0x70,0x32);
185     int tempMillennium = BCDtoDec(inb(0x71));
186     outb(0x70,0x09);
187     int tempYear = BCDtoDec(inb(0x71));
188     cli();
189         outb(0x70,0x07);
190         outb(0x71,DectoBCD (day));
191         outb(0x70,0x08);
192         outb(0x71,DectoBCD (month));
193         outb(0x70,0x32);
194         outb(0x71,DectoBCD (millennium));
```

```
195         outb(0x70,0x09);
196         outb(0x71,DectoBCD (year));
197         sti();
198     outb(0x70,0x07);
199     unsigned char newDay = BCDtoDec(inb(0x71));
200     outb(0x70,0x08);
201     unsigned char newMonth = BCDtoDec(inb(0x71));
202     outb(0x70,0x32);
203     unsigned char newMillennium = BCDtoDec(inb(0x71));
204     outb(0x70,0x09);
205     unsigned char newYear = BCDtoDec(inb(0x71));
206     if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
207       printf("Your input was invalid\n");
208       cli();
209         outb(0x70,0x07);
210         outb(0x71,DectoBCD (tempDay));
211         outb(0x70,0x08);
212         outb(0x71,DectoBCD (tempMonth));
213         outb(0x70,0x32);
214         outb(0x71,DectoBCD (tempMillennium));
215         outb(0x70,0x09);
216         outb(0x71,DectoBCD (tempYear));
217         sti();
218     }
219     else
220       printf("Date Set\n");
221     }
```

4.23.1.14 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 109 of file userFunctions.c.

```
109                                                 {
110     outb(0x70,0x04);
111     unsigned char tempHours = BCDtoDec(inb(0x71));
112     outb(0x70,0x02);
113     unsigned char tempMinutes = BCDtoDec(inb(0x71));
114     outb(0x70,0x00);
115     unsigned char tempSeconds = BCDtoDec(inb(0x71));
116         cli(); //outb(device + 1, 0x00); //disable interrupts
117         outb(0x70,0x04);
118         outb(0x71, DectoBCD(hours));// change to bcd
119         outb(0x70,0x02);
120         outb(0x71, DectoBCD(minutes));
121         outb(0x70,0x00);
122         outb(0x71, DectoBCD(seconds));
123         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
124     outb(0x70,0x04);
125     unsigned char newHours = BCDtoDec(inb(0x71));
126     outb(0x70,0x02);
127     unsigned char newMinutes = BCDtoDec(inb(0x71));
128     outb(0x70,0x00);
129     unsigned char newSeconds = BCDtoDec(inb(0x71));
130     if(newHours != hours || newMinutes != minutes || newSeconds != seconds){
131       printf("Your input was invalid\n");
132       cli(); //outb(device + 1, 0x00); //disable interrupts
133         outb(0x70,0x04);
```

```
134         outb(0x71, DectoBCD(tempHours));// change to bcd
135         outb(0x70,0x02);
136         outb(0x71, DectoBCD(tempMinutes));
137         outb(0x70,0x00);
138         outb(0x71, DectoBCD(tempSeconds));
139         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
140     }
141     else
142       printf("Time Set\n");
143     }
```

4.23.1.15 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 463 of file userFunctions.c.

```
463                 {
464   int class, check, state, prior, status, i,j;
465   char name[10];
466   char ready[] = "Ready Queue:\n";
467   char block[] = "Blocked Queue: \n";
468   char cname[] = "Name: ";
469   char cclass[] = "Class: ";
470   char cstate[] = "State: ";
471   char cstatus[] = "Status: ";
472   char cprior[] = "Priority: ";
473   char line[] = "\n";
474   char dline[] = "\n\n";
475   check = 15;
476
477   sys_req(WRITE, COM1, ready, &check );
478
479   if(ReadyQueue.head != NULL)
480     PCB* pcb = ReadyQueue.head;
481
482   do {
483     if(pcb != ReadyQueue.head)
484       pcb = pcb.next;
485
486     class = pcb->Process_Class;
487     strcpy(name,pcb->Process_Name);
488     state = pcb->ReadyState;
489     status = pcb->SuspendedState;
490     prior = pcb->Priority;
491
492     printf(cname);
493     printf(name);
494     printf(line);
495
496     printf(cclass);
497     sys_req(WRITE, COM1, itoa(class), &check);
498     printf(line);
499
500     printf(cstate);
501     sys_req(WRITE, COM1, itoa(state), &check);
502     printf(line);
503
504     printf(cstatus);
505     sys_req(WRITE, COM1, itoa(status), &check);
506     printf(line);
507
508     printf(cprior);
509     sys_req(WRITE, COM1, itoa(prior), &check);
510     printf(dline);
511
512   } while(pcb.next != NULL);
513
514   sys_req(WRITE, COM1, block, &check );
515
516   if(BlockedQueue.head != NULL)
```

```
517     PCB* pcb = BlockedQueue.head;
518
519   do {
520     if(pcb != BlockedQueue.head)
521       pcb = pcb.next;
522
523     class = pcb->Process_Class;
524     strcpy(name,pcb->Process_Name);
525     state = pcb->ReadyState;
526     status = pcb->SuspendedState;
527     prior = pcb->Priority;
528
529     printf(cname);
530     printf(name);
531     printf(line);
532
533     printf(cclass);
534     sys_req(WRITE, COM1, itoa(class), &check);
535     printf(line);
536
537     printf(cstate);
538     sys_req(WRITE, COM1, itoa(state), &check);
539     printf(line);
540
541     printf(cstatus);
542     sys_req(WRITE, COM1, itoa(status), &check);
543     printf(line);
544
545     printf(cprior);
546     sys_req(WRITE, COM1, itoa(prior), &check);
547   } while(pcb.next != NULL);
548 }
```

4.23.1.16 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 604 of file userFunctions.c.

```
604                     {
605   int class, check, state, prior, status, j;
606   char name[20];
607   char block[] = "Blocked Queue: \n";
608   char cname[] = "Name: ";
609   char cclass[] = "Class: ";
610   char cstate[] = "State: ";
611   char cstatus[] = "Status: ";
612   char cprior[] = "Priority: ";
613   char line[] = "\n";
614   check = 15;
615
616   sys_req(WRITE, COM1, block, &check );
617   if(BlockedQueue.head != NULL)
618     PCB* pcb = BlockedQueue.head;
619
620   do {
621     if(pcb != BlockedQueue.head)
622       pcb = pcb.next;
623
624     class = pcb->Process_Class;
625     strcpy(name,pcb->Process_Name);
626     state = pcb->ReadyState;
627     status = pcb->SuspendedState;
628     prior = pcb->Priority;
629
630     printf(cname);
631     printf(name);
632     printf(line);
633
634     printf(cclass);
635     sys_req(WRITE, COM1, itoa(class), &check);
636     printf(line);
```

```
637
638     printf(cstate);
639     sys_req(WRITE, COM1, itoa(state), &check);
640     printf(line);
641
642     printf(cstatus);
643     sys_req(WRITE, COM1, itoa(status), &check);
644     printf(line);
645
646     printf(cprior);
647     sys_req(WRITE, COM1, itoa(prior), &check);
648   } while(pcb.next != NULL);
649 }
```

4.23.1.17 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 420 of file userFunctions.c.

```
420                                     {
421   int check = 5;
422   char name[10];
423   char cname[] = "Name: ";
424   char cclass[] = "Class: ";
425   char cstate[] = "State: ";
426   char cstatus[] = "Status: ";
427   char cprior[] = "Priority: ";
428   char line[] = "\n";
429   PCB* pcb = FindPCB(ProcessName);
430   strcpy(name,pcb->Process_Name);
431   int class = pcb->Process_Class;
432   int state = pcb->ReadyState;
433   int status = pcb->SuspendedState;
434   int prior = pcb->Priority;
435   if(name == NULL){
436     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
437   } else{
438     printf(cname);
439     printf(ProcessName);
440     printf(line);
441
442     printf(cclass);
443     sys_req(WRITE, COM1, itoa(class), &check);
444     printf(line);
445
446     printf(cstate);
447     sys_req(WRITE, COM1, itoa(state), &check);
448     printf(line);
449
450     printf(cstatus);
451     sys_req(WRITE, COM1, itoa(status), &check);
452     printf(line);
453
454     printf(cprior);
455     sys_req(WRITE, COM1, itoa(prior), &check);
456   }
457 }
```

4.23.1.18 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 553 of file userFunctions.c.

```
553                     {
554   int class, check, state, prior, status, i;
555   char name[10];
556   char ready[] = "Ready Queue:\n";
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   if(ReadyQueue.head != NULL)
568     PCB* pcb = ReadyQueue.head;
569
570   do {
571     if(pcb != ReadyQueue.head)
572       pcb = pcb.next;
573
574     class = pcb->Process_Class;
575     strcpy(name,pcb->Process_Name);
576     state = pcb->ReadyState;
577     status = pcb->SuspendedState;
578     prior = pcb->Priority;
579
580     printf(cname);
581     printf(name);
582     printf(line);
583
584     printf(cclass);
585     sys_req(WRITE, COM1, itoa(class), &check);
586     printf(line);
587
588     printf(cstate);
589     sys_req(WRITE, COM1, itoa(state), &check);
590     printf(line);
591
592     printf(cstatus);
593     sys_req(WRITE, COM1, itoa(status), &check);
594     printf(line);
595
596     printf(cprior);
597     sys_req(WRITE, COM1, itoa(prior), &check);
598   } while(pcb.next != NULL);
599 }
```

4.23.1.19 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 356 of file userFunctions.c.

```
356                                 {
357   // Name Error check
358   // Error check (Valid Name)
359   PCB* pcb = FindPCB(ProcessName);
360   if (pcb == NULL)  {
361     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
362   }
363   else {
364     if(pcb->SuspendedState == YES)  {
365         printf("\x1b[32m""\nThis Process is already SUSPENDED \n""\x1b[0m");
366     }
367     else    {
368         pcb->SuspendedState = YES;
369     }
370   }
371
372 }
```

4.23.1.20 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 263 of file userFunctions.c.

```
263                                 {
264         if((c >= 65) && (c <= 90))  {
265                 c = c + 32;
266             }
267         return c;
268         }
```

4.23.1.21 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 718 of file userFunctions.c.

```
718                                 {
719   PCB* pcb = FindPCB(ProcessName);
720   if (pcb == NULL)  {
721     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
722   }
723   else {
724     if(pcb->ReadyState == READY)    {
725         printf("\x1b[32m""\nThis Process is already in the READY state \n""\x1b[0m");
726     }
727     else    {
728         pcb->ReadyState = READY;
729     }
730   }
731 }
```

4.23.1.22 Version()

```
void Version ( )
```

Description: Simply returns a char containing "Version: R(module).

(the iteration that module is currently on).

No parameters.

Definition at line 256 of file userFunctions.c.

```
256                     {
257         printf("Version: R2.0 \n");
258     }
```

Index