RUNTIME TERROR OS

R5

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 Alarm Struct Reference

Public Attributes

- int **hour**
- int **minute**
- int **second**
- char **message** [85]
- struct Alarm * **next**
- struct Alarm * **prev**

3.1.1 Detailed Description

Definition at line 15 of file userFunctions.h.

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

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/R1/userFunctions.h

3.2 CMCB Struct Reference

Public Attributes

- u32int **size**
- struct CMCB * **prev**
- struct CMCB * **next**
- char **Process_name** [10]
- u32int **address**
- int **MEMState**

3.2.1 Detailed Description

Definition at line 4 of file MCB.h.

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

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/R5/MCB.h

3.3 context Struct Reference

Public Attributes

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

3.3.1 Detailed Description

Definition at line 34 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

3.4 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**

3.4.1 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

3.5 footer Struct Reference

Public Attributes

- header **head**

3.5.1 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

3.6 gdt_descriptor_struct Struct Reference

Public Attributes

- u16int **limit**
- u32int **base**

3.6.1 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

3.7 gdt_entry_struct Struct Reference

Public Attributes

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

3.7.1 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

3.8 header Struct Reference

Public Attributes

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

3.8.1 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

3.9 heap Struct Reference

Public Attributes

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

3.9.1 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

3.10 idt_entry_struct Struct Reference

Public Attributes

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

3.10.1 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

3.11 idt_struct Struct Reference

Public Attributes

- u16int **limit**
- u32int **base**

3.11.1 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

3.12 index_entry Struct Reference

Public Attributes

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

3.12.1 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

3.13 index_table Struct Reference

Public Attributes

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

3.13.1 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

3.14 List Struct Reference

Public Attributes

- Alarm * **head**
- Alarm * **tail**

3.14.1 Detailed Description

Definition at line 24 of file userFunctions.h.

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

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/R1/userFunctions.h

3.15 MemList Struct Reference

Public Attributes

- CMCB * **head**

3.15.1 Detailed Description

Definition at line 18 of file MCB.h.

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

- D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/R5/MCB.h

3.16 page_dir Struct Reference

Public Attributes

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

3.16.1 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

3.17 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

3.17.1 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

3.18 page_table Struct Reference

Public Attributes

- page_entry **pages** [1024]

3.18.1 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

3.19 param Struct Reference

Public Attributes

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

3.19.1 Detailed Description

Definition at line 34 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

3.20 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**

3.20.1 Detailed Description

Definition at line 15 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

3.21 Queue Struct Reference

Public Attributes

- int **count**
- PCB * **head**
- PCB * **tail**

3.21.1 Detailed Description

Definition at line 27 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)
- 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 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 **sys_call_isr** ()
- void **isr0** ()
- void **do_isr** ()
- void **init_irq** (void)
- void **init_pic** (void)
- void **do_divide_error** ()
- void **do_debug** ()
- void **do_nmi** ()
- void **do_breakpoint** ()
- void **do_overflow** ()
- void **do_bounds** ()
- void **do_invalid_op** ()
- void **do_device_not_available** ()
- void **do_double_fault** ()
- void **do_coprocessor_segment** ()
- void **do_invalid_tss** ()
- void **do_segment_not_present** ()
- void **do_stack_segment** ()
- void **do_general_protection** ()
- void **do_page_fault** ()
- void **do_reserved** ()
- void **do_coprocessor** ()

Variables

- idt_entry **idt_entries** [256]

4.11 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/sys_proc_loader.h"
#include "modules/R1/userFunctions.h"
#include "modules/R5/R5commands.h"
#include "modules/R5/MCB.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)
- 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 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** ()
- u32int * **sys_call** (context *registers)

Variables

- param **params**
- int **current_module** = -1
- u32int(* **student_malloc**)(u32int)
- int(* **student_free**)(void *)
- PCB * **cop**
- context * **initial**

4.19 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/mpx_↩ supt.h File Reference

```
#include <system.h>
#include "R2/PCB.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** ()
- u32int ∗ **sys_call** (context ∗registers)

4.20 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/R1/com↩ Hand.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 23 of file comHand.c.

```
23                      {
24
25          Help("\0");
26
27          char cmdBuffer[100];
28          int bufferSize = 99;
29          int quit = 0;
30          int shutdown = 0;
31
32          while(quit != 1)    {
33              memset(cmdBuffer, '\0', 100);
34            sys_req(READ, DEFAULT_DEVICE, cmdBuffer, &bufferSize);
35              char* FirstToken = strtok(cmdBuffer, "-");
36              char* SecondToken = strtok(NULL, "-");
37              char* ThirdToken = strtok(NULL, "-");
38              char* FourthToken = strtok(NULL, "-");
39              char* FifthToken = strtok(NULL, "-");
40              if(shutdown == 0)   {
41 /*****************************************************************************
42                  R1 comHand
43 *****************************************************************************/
44                      if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,NULL) == 0)     {
45                          Help("\0");
46                      }
47                      //R1 Commands
48                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"version") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
49                          Help("Version");
50                      }
51                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"getDate") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
52                          Help("GetDate");
53                      }
54                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"setDate") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
55                          Help("SetDate");
56                      }
57                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"getTime") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
58                          Help("GetTime");
59                      }
60                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"setTime") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
61                          Help("SetTime");
62                      }
63                      // R2 Commands
64                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"suspend") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
65                          Help("suspend");
66                      }
67                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"resume") == 0 &&
       strcmp(ThirdToken,NULL) == 0)  {
68                          Help("resume");
69                      }
70                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"setPriority") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
71                          Help("setPriority");
72                      }
73                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showPCB") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
74                          Help("showPCB");
75                      }
76                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showAll") == 0 &&
       strcmp(ThirdToken,NULL) == 0) {
77                          Help("showAll");
78                      }
79                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showReady") == 0 &&
       strcmp(ThirdToken,NULL) == 0)   {
```

```
80                          Help("showReady");
81                      }
82                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showBlocked") == 0 &&
    strcmp(ThirdToken,NULL) == 0) {
83                          Help("showBlocked");
84                      }
85                      // Temporary R2 commands
86                      // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"createPCB") == 0 &&
    strcmp(ThirdToken,NULL) == 0)  {
87                      //  Help("createPCB");
88                      // }
89                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"deletePCB") == 0 &&
    strcmp(ThirdToken,NULL) == 0)   {
90                          Help("deletePCB");
91                      }
92                      // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"block") == 0 &&
    strcmp(ThirdToken,NULL) == 0)  {
93                      //  Help("block");
94                      // }
95                      // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"unblock") == 0 &&
    strcmp(ThirdToken,NULL) == 0)    {
96                      //  Help("unblock");
97                      // }
98                      else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"shutdown") == 0 &&
    strcmp(ThirdToken,NULL) == 0)    {
99                          Help("shutdown");
100                     }
101                     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"infinite") == 0 &&
    strcmp(ThirdToken,NULL) == 0)    {
102                         Help("infinte");
103                     }
104                     // R4 Commands
105                     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"loadr3") == 0 &&
    strcmp(ThirdToken,NULL) == 0)  {
106                         Help("loadr3");
107                     }
108                     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"alarm") == 0 &&
    strcmp(ThirdToken,NULL) == 0)   {
109                         Help("alarm");
110                     }
111                     // Bonus Command
112                     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
    strcmp(ThirdToken,NULL) == 0)   {
113                         Help("clear");
114                     }
115                     // Temporary R5 Commands
116                     // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
    strcmp(ThirdToken,NULL) == 0)  {
117                     //  Help("heap");
118                     // }
119                     // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
    strcmp(ThirdToken,NULL) == 0)  {
120                     //  Help("alloc");
121                     // }
122                     // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
    strcmp(ThirdToken,NULL) == 0)  {
123                     //  Help("free");
124                     // }
125                     // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
    strcmp(ThirdToken,NULL) == 0)  {
126                     //  Help("empty");
127                     // }
128                     // R5 Commands
129                     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
    strcmp(ThirdToken,NULL) == 0)   {
130                         Help("showFree");
131                     }
132                     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
    strcmp(ThirdToken,NULL) == 0)   {
133                         Help("showAlloc");
134                     }
135
136
137
138
139
140
141                     else if(strcmp(FirstToken,"version") == 0 && strcmp(SecondToken,NULL) == 0)
142                         Version();
143                     else if(strcmp(FirstToken,"clear") == 0 && strcmp(SecondToken,NULL) == 0)
144                         clear();
145
146                     else if(strcmp(FirstToken,"getDate") == 0 && strcmp(SecondToken,NULL) == 0)
147                         GetDate();
148
149                     else if(strcmp(FirstToken,"setDate") == 0){
150                         if (EdgeCase(SecondToken) == 1 && EdgeCase(ThirdToken) == 1 &&
```

```
    EdgeCase(FourthToken) == 1 && EdgeCase(FifthToken) == 1)    {
151                         SetDate(atoi(SecondToken), atoi(ThirdToken), atoi(FourthToken),
    atoi(FifthToken));
152                     }
153                     else
154                         printf("\x1b[31m""\nERROR: Invalid parameters for setDate \n""\x1b[0m");
155                 }
156                 else if(strcmp(FirstToken,"getTime") == 0 && strcmp(SecondToken,NULL) == 0) //Return
    the current time held by the registers.
157                     GetTime();
158                 else if(strcmp(FirstToken,"setTime") == 0 && strcmp(FifthToken,NULL) == 0){
159                     if (EdgeCase(SecondToken) == 1 && EdgeCase(ThirdToken) == 1 &&
    EdgeCase(FourthToken) == 1)    {
160                         SetTime(atoi(SecondToken), atoi(ThirdToken), atoi(FourthToken));
    //input as Hour-Minute-Seconds
161                     }
162                     else
163                         printf("\x1b[31m""\nERROR: Invalid parameters for setTime \n""\x1b[0m");
164                 }
165
166
167
168
169
170
171  /*****************************************************************************
172                 R2 comHand
173  *****************************************************************************/
174                 else if(strcmp(FirstToken,"suspend") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
175                     Suspend(SecondToken);
176                 }
177                 else if(strcmp(FirstToken,"resume") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
178                     Resume(SecondToken);
179                 }
180                 else if(strcmp(FirstToken,"setPriority") == 0 && strcmp(FourthToken,NULL) == 0 &&
    strcmp(FifthToken,NULL) == 0) {
181                     if(EdgeCase(ThirdToken) == 1)    {
182                         Set_Priority(SecondToken, atoi(ThirdToken));    //input as
    setPriority-Process_Name-Priority
183                     }
184                     else
185                         printf("\x1b[31m""\nERROR: Invalid parameters for setPriority, priority must
    be entered as a integer. \n""\x1b[0m");
186                 }
187                 else if(strcmp(FirstToken,"showPCB") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
188                     Show_PCB(SecondToken);
189                     printf("\n");
190                 }
191                 else if(strcmp(FirstToken,"showAll") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
192                     Show_All();
193                     printf("\n");
194                 }
195                 else if(strcmp(FirstToken,"showReady") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
196                     Show_Ready();
197                     printf("\n");
198                 }
199                 else if(strcmp(FirstToken,"showBlocked") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
200                     Show_Blocked();
201                     printf("\n");
202                 }
203
204
205
206                 /********** R2 Temp Commands **********/
207                 //Removed from active for R3/R4
208                 /*
209                 else if(strcmp(FirstToken,"createPCB") == 0) {
210                     if( strlen(SecondToken) < 11)    {
211                         Create_PCB(SecondToken, atoi(ThirdToken), atoi(FourthToken));
    //input as Process_Name-Priority-Class
212                     }
213                     else
214                         printf("\x1b[31m""\nERROR: Invalid parameters for createPCB, Process_name
    must only contain 10 or fewer characters. \n""\x1b[0m");
215                 }
216                 */
217                 else if(strcmp(FirstToken,"deletePCB") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
218                     Delete_PCB(SecondToken);
219                 }
220
```

```
221
222
223                     //Removed from active for R3/R4
224                     /*
225                     else if(strcmp(FirstToken,"block") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
226                         Block(SecondToken);
227                     }
228                     else if(strcmp(FirstToken,"unblock") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
229                         Unblock(SecondToken);
230                     }
231                     */
232      /*************************************************************************************
233                     R3 comHand
234      *************************************************************************************/
235                     //Removed for R4
236                     /*
237                     else if(strcmp(FirstToken,"yield") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
238                         yield();
239                         printf("\n");
240                     }
241                     else if(strcmp(FirstToken,"loadr3") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
242                         loader();
243                         printf("\n");
244                     }
245                     */
246      /*************************************************************************************
247                     R4 comHand
248      *************************************************************************************/
249                     else if(strcmp(FirstToken,"alarm") == 0) {
250                         if (EdgeCase(ThirdToken) == 1 && EdgeCase(FourthToken) == 1 &&
    EdgeCase(FifthToken) == 1)    {
251                             if (atoi(ThirdToken) < 24 && atoi(FourthToken) < 60 && atoi(FifthToken) <
    60)    {
252                                 loaderalarm(SecondToken, atoi(ThirdToken), atoi(FourthToken),
    atoi(FifthToken));
253                                 printf("\n");   //input as Message-Hour-Minute-Seconds
254                             }
255                             else
256                             printf("\x1b[31m""\nERROR: Invalid parameters for alarm, must be a valid
    time \n""\x1b[0m");
257                         }
258                         else
259                             printf("\x1b[31m""\nERROR: Invalid parameters for alarm \n""\x1b[0m");
260
261                     }
262                     else if(strcmp(FirstToken,"loadr3") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
263                         loader();
264                         printf("\n");
265                     }
266                     else if(strcmp(FirstToken,"infinite") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
267                         loaderinfinite();
268                         printf("\n");
269                     }
270
271      /*************************************************************************************
272                     R5 comHand
273      *************************************************************************************/
274                     // else if(strcmp(FirstToken,"heap") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
275                     //  Init_Heap(atoi(SecondToken));
276                     // }
277                     // else if(strcmp(FirstToken,"alloc") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
278                     //  Alloc_Mem(atoi(SecondToken));
279                     // }
280                     // else if(strcmp(FirstToken,"free") == 0 && strcmp(ThirdToken,NULL) == 0 &&
    strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
281                     //  Free_Mem(atoi(SecondToken));
282                     // }
283                     // else if(strcmp(FirstToken,"empty") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
284                     //  IsEmpty();
285                     // }
286                     else if(strcmp(FirstToken,"showFree") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
287                         ShowFree();
288                     }
289                     else if(strcmp(FirstToken,"showAlloc") == 0 && strcmp(SecondToken,NULL) == 0 &&
    strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
290                         ShowAlloc();
291                     }
```

```
292
293     /*****************************************************************************
294                     shutdown comHand
295     *****************************************************************************/
296                     else if(strcmp(FirstToken,"shutdown") == 0 && strcmp(SecondToken,NULL) == 0){
297                         printf("\x1b[33m""\nAre you sure you want to shutdown? [yes/no]\n""\x1b[0m");
298                         shutdown = 1;
299                     }
300                     else    {
301                         printf("\x1b[31m""\nERROR: Not a valid command \n""\x1b[0m");
302                     }
303                 }
304                 else{
305                     if(strcmp(FirstToken,"yes") == 0 && shutdown == 1)    {
306                         quit = 1;
307                     }
308                     else if(strcmp(FirstToken,"no") == 0){
309                         printf("\x1b[33m""\nShutdown Cancelled\x1b[0m \n");
310                         shutdown = 0;
311                     }
312                     else
313                         printf("\x1b[31m""\nERROR: Please enter \"yes\" or \"no\" \n""\x1b[0m");
314                 }
315                 sys_req(IDLE, DEFAULT_DEVICE, NULL, NULL);
316         }
317         getReady() -> head = NULL;
318         sys_req(EXIT, DEFAULT_DEVICE, NULL, NULL);
319         return 0;   //shutdown procedure
320     }
```

4.21 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/R1/user↩ Functions.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"
#include "../procsr3.h"
#include "../sys_proc_loader.h"
```

Functions

- void **clear** ()
- 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)
- void **loader** ()
- void **loadr3** (char *name, u32int func)
- void **yield** ()
- void **loaderinfinite** ()
- List * **getList** ()
- void **loaderalarm** (char text[ ], int hours, int minutes, int seconds)

Variables

- List **AlarmList**

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 82 of file userFunctions.c.

```
82                              {
83     return (((BCD»4)*10) + (BCD & 0xF));
84 }
```

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 958 of file userFunctions.c.

```
958                                 {
959   PCB* pcb = FindPCB(ProcessName);
960   if (pcb == NULL)  {
961     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
962   }
963   else {
964     if(pcb->ReadyState == BLOCKED)  {
965         printf("\x1b[32m""\nThis Process is already BLOCKED \n""\x1b[0m");
966     }
967     else    {
968       RemovePCB(pcb);
969       pcb->ReadyState = BLOCKED;
970       InsertPCB(pcb);
971     }
972   }
973 }
```

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 901 of file userFunctions.c.

```
901                                                                 {
902   if (FindPCB(ProcessName) == NULL) {
903     if(Priority >= 0 && Priority < 10){
904       if(Class == 0 || Class == 1){
905         PCB* pcb = SetupPCB(ProcessName, Class, Priority);
906         InsertPCB(pcb);
907       } else{
908         printf("\x1b[31m""\nERROR: Not a valid Class \n""\x1b[0m");
909       }
910     } else{
911       printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
912     }
913   } else{
914     printf("\x1b[31m""\nERROR: This Process Name already exists \n""\x1b[0m");
915   }
916 }
```

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 89 of file userFunctions.c.

```
89                              {
90     return (((Decimal/10) « 4) | (Decimal % 10));
91 }
```

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 927 of file userFunctions.c.

```
927                                     {
928   PCB* pcb = FindPCB(ProcessName);
929   if (pcb == NULL)  {
930     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
931   }
932   else if(strcmp(pcb->Process_Name,"InfProc") == 0)   {
933         if(pcb->SuspendedState == YES)  {
934             RemovePCB(pcb);
935             FreePCB(pcb);
936         }
937         else
938             printf("\x1b[31m""\nERROR:This process cannot be deleted unless it is in the suspended
       state\n""\x1b[0m");
939   }
940   else if(pcb -> Process_Class == SYSTEM)   {
941       printf("\x1b[31m""\nERROR: System Processes cannot be deleted from the system. \n""\x1b[0m");
942   }
943     else {
944         RemovePCB(pcb);
945         FreePCB(pcb);
946     }
947 }
```

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 110 of file userFunctions.c.

```
110                                {
111   int valid = 0;
112   if (strcmp(pointer, "00") == 0) {
113     valid = 1;
114     return valid;
115   }
116   else if (strcmp(pointer, "0") == 0) {
117     valid = 1;
118     return valid;
119   }
120   else  {
121     int j;
122     valid = 0;
123     for(j = 0; j <= 99; j++)  {
124       if(strcmp(pointer,itoa(j)) == 0)
125         valid = 1;
126     }
127     if(valid == 0)  {
128       return valid;
129     }
130   }
131   return valid;
132 }
```

4.21.1.7 GetDate()

```
void GetDate ( )
```

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

No parameters.

Definition at line 272 of file userFunctions.c.

```
272                {
273
274   outb(0x70,0x07);
275     unsigned char day = BCDtoDec(inb(0x71));
276     outb(0x70,0x08);
277     unsigned char month = BCDtoDec(inb(0x71));
278     outb(0x70,0x32);
279     unsigned char millennium = BCDtoDec(inb(0x71));
280     char msg[2] = "-";
281     char msg3[10] = "Date: ";
282     printf(msg3);
283
284      printf(itoa(day));
285     //sys_req(WRITE, COM1, itoa(day), &check);
286     printf(msg);
287     printf(itoa(month));
288     //sys_req(WRITE, COM1, itoa(month), &check);
289     printf(msg);
290     printf(itoa(millennium));
291     //sys_req(WRITE, COM1, itoa(millennium), &check);
292   outb(0x70,0x09);
293   if(BCDtoDec(inb(0x71)) == 0){
294      printf("00");
295     //sys_req(WRITE, COM1, "00", &check);
296   }
297   else {
298         unsigned char year = BCDtoDec(inb(0x71));
299          printf(itoa(year));
300         //sys_req(WRITE, COM1, itoa(year), &check);
301   }
302     printf("\n");
303 }
```

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 191 of file userFunctions.c.

```
191                {
192
193     int hour;
194     int minute;
195     int second;
196     outb(0x70,0x04);
197     unsigned char hours = inb(0x71);
198     outb(0x70,0x02);
199     unsigned char minutes = inb(0x71);
200     outb(0x70,0x00);
201     unsigned char seconds = inb(0x71);
202     char msg1[2] = ":";
203     char msg2[10] = "Time: ";
204     printf(msg2);
205     hour = BCDtoDec(hours);
206     printf(itoa(hour));
207     //sys_req(WRITE, COM1, itoa(hour), &check);
208     printf(msg1);
209     minute = BCDtoDec(minutes);
210     printf(itoa(minute));
211     //sys_req(WRITE, COM1, itoa(minute), &check);
212     printf(msg1);
213     second = BCDtoDec(seconds);
214     printf(itoa(second));
215     //sys_req(WRITE, COM1, itoa(second), &check);
216   printf("\n");
217 }
```

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 332 of file userFunctions.c.

```
332                        {
333     if (request[0] == '\0') {
334         //removed for R3/R4 from active command list
335         //\n createPCB \n block \n unblock
336         //\n heap         alloc \n free         empty
337         printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
338         printf("\n getDate      setDate \n getTime      setTime \n version      suspend \n resume
       setPriority \n showPCB      showAll \n showReady    showBlocked  \n deletePCB    shutdown \n alarm
           clear \n loadr3       infinte \n showFree     showAlloc \n\n");
```

```
339     }
340     else if (strcmp(request, "GetDate") == 0) {
341         printf("\n getDate returns the current date that is loaded onto the operating system.\n");
342     }
343     else if (strcmp(request, "SetDate") == 0) {
344         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");
345     }
346     else if (strcmp(request, "GetTime") == 0) {
347         printf("\n getTime returns the current time as hours, minutes, seconds that is loaded onto the
      operating system.\n");
348     }
349     else if (strcmp(request, "SetTime") == 0) {
350         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");
351     }
352     else if (strcmp(request, "Version") == 0) {
353         printf("\n version returns the current operating software version that the system is
      running.\n");
354     }
355     else if (strcmp(request, "infinte") == 0) {
356         printf("\n infinite Loads the infinite process into the ready queue.\n");
357     }
358     else if (strcmp(request, "loadr3") == 0)  {
359         printf("\n loadr3 Loads in all five of the R3 test processes in a suspended state into the
      queue.\n");
360     }
361     else if (strcmp(request, "alarm") == 0)       {
362         printf("\n alarm creates a user specified alarm with a user set message and time
      alarm-MSG-hour-minute-second.\n");
363     }
364     else if (strcmp(request, "clear") == 0)       {
365         printf("\n clear erases the console of all typed commands and refreshes it with just the command
      list.\n");
366     }
367
368   else if(strcmp(request, "shutdown") == 0)   {
369     printf("\n shutdown shuts down the system.\n");
370   }
371
372
373
      /*****************************************************************************************************
374             R2 Commands
375
      *****************************************************************************************************/
376   else if(strcmp(request,"suspend") == 0) {
377         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");
378   }
379   else if(strcmp(request,"resume") == 0) {
380         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");
381   }
382   else if(strcmp(request,"setPriority") == 0) {
383         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");
384   }
385   else if(strcmp(request,"showPCB") == 0) {
386         printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
      user.\n");
387   }
388   else if(strcmp(request,"showAll") == 0) {
389         printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
      queues.\n");
390   }
391   else if(strcmp(request,"showReady") == 0) {
392         printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
      currently are in the ready state.\n");
393   }
394   else if(strcmp(request,"showBlocked") == 0) {
395         printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
      currently are in the blocked state.\n");
396   }
397
398 /******************************** R2 Temp Commands
      *******************************************************/
399   else if(strcmp(request,"deletePCB") == 0) {
400         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");
401   }
402   //removed for R3/R4 from active command list
403   /*
404   else if(strcmp(request,"createPCB") == 0) {
405         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");
406    }
407    else if(strcmp(request,"block") == 0) {
408          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");
409    }
410    else if(strcmp(request,"unblock") == 0) {
411          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");
412    }
413    */
414
415
416 /******************************** R5 Temp Commands
       *********************************************************/
417    // else if(strcmp(request,"heap") == 0) {
418      //  printf("\n heap initializes the memory heap for the entire system.\n");
419    // }
420    // else if(strcmp(request,"alloc") == 0) {
421      //  printf("\n alloc allocates the specified amount of memory to the specific process
       (alloc-process_name-size).\n");
422    // }
423    // else if(strcmp(request,"free") == 0) {
424      //  printf("\n free frees the specified memory at the address given (free-address).\n");
425    // }
426    // else if(strcmp(request,"empty") == 0) {
427      //  printf("\n isempty returns true or false depending on if the heap has allocated memory.\n");
428    // }
429 /******************************** R5 Commands
       *********************************************************/
430    else if(strcmp(request,"showFree") == 0) {
431          printf("\n showfree shows all the free blocks avaliable within the heap list.\n");
432    }
433    else if(strcmp(request,"showAlloc") == 0) {
434          printf("\n showAlloc shows all the allocated blocks within the heap list.\n");
435    }
436
437
438    else  {
439          printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
       full list of commands.\n""\x1b[0m");
440    }
441 }
```

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 51 of file userFunctions.c.

```
51                    {
52
53      int i,j,k,count;
54      i = num;
55      j = 0;
56      count = 0;
57      while(i){ // count number of digits
58          count++;
59          i /= 10;
60      }
61
62      char* arr1;
63      char arr2[count];
```

```
64      arr1 = (char*)sys_alloc_mem(count); //memory allocation
65
66      while(num){ // seperate last digit from number and add ASCII
67          arr2[++j] = num%10 + '0';
68          num /= 10;
69      }
70
71      for(k = 0; k < j; k++){ // reverse array results
72          arr1[k] = arr2[j-k];
73      }
74      arr1[k] = '\0';
75      sys_free_mem(arr1);
76      return(char*)arr1;
77  }
```

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 484 of file userFunctions.c.

```
484                                     {
485      PCB* pcb = FindPCB(ProcessName);
486      if (pcb == NULL)    {
487        printf(RED"\nERROR: Not a valid process name \n"RESET);
488      }
489      else {
490          if(pcb->SuspendedState == NO)   {
491              printf(GRN"\nThis Process is already in the NONSUSPENDED state \n"RESET);
492          }
493          else if(pcb -> Process_Class == APPLICATION)    {
494              pcb->SuspendedState = NO;
495          }
496          else
497              printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
498      }
499  }
```

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 511 of file userFunctions.c.

```
511                                                     {
512     PCB* pcb = FindPCB(ProcessName);
513     if (pcb == NULL)    {
514         printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
515     }
516     else if(Priority >= 10){
517         printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
518     }
519     else if(pcb -> Process_Class == APPLICATION) {
520         RemovePCB(pcb);
521         pcb->Priority = Priority;
522         InsertPCB(pcb);
523     }
524     else
525         printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
526 }
```

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 225 of file userFunctions.c.

```
225                                                     {
226   outb(0x70,0x07);
227   int tempDay = BCDtoDec(inb(0x71));
228   outb(0x70,0x08);
229   int tempMonth = BCDtoDec(inb(0x71));
230   outb(0x70,0x32);
231   int tempMillennium = BCDtoDec(inb(0x71));
232   outb(0x70,0x09);
233   int tempYear = BCDtoDec(inb(0x71));
234   cli();
235     outb(0x70,0x07);
236     outb(0x71,DectoBCD (day));
237     outb(0x70,0x08);
238     outb(0x71,DectoBCD (month));
239     outb(0x70,0x32);
240     outb(0x71,DectoBCD (millennium));
241     outb(0x70,0x09);
242     outb(0x71,DectoBCD (year));
243     sti();
244   outb(0x70,0x07);
245   unsigned char newDay = BCDtoDec(inb(0x71));
```

```
246   outb(0x70,0x08);
247   unsigned char newMonth = BCDtoDec(inb(0x71));
248   outb(0x70,0x32);
249   unsigned char newMillennium = BCDtoDec(inb(0x71));
250   outb(0x70,0x09);
251   unsigned char newYear = BCDtoDec(inb(0x71));
252   if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
253     printf("Your input was invalid\n");
254     cli();
255         outb(0x70,0x07);
256         outb(0x71,DectoBCD (tempDay));
257         outb(0x70,0x08);
258         outb(0x71,DectoBCD (tempMonth));
259         outb(0x70,0x32);
260         outb(0x71,DectoBCD (tempMillennium));
261         outb(0x70,0x09);
262         outb(0x71,DectoBCD (tempYear));
263         sti();
264   }
265   else
266     printf("Date Set\n");
267 }
```

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 152 of file userFunctions.c.

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

```
182         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
183   }
184   else
185     printf("Time Set\n");
186 }
```

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 611 of file userFunctions.c.

```
611                 {
612     Show_Ready();
613     Show_Blocked();
614 }
```

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 760 of file userFunctions.c.

```
760                     {
761     if(getBlocked()->head == NULL)  {
762          printf("\x1b[32m""\n The Blocked Queue is empty \n""\x1b[0m");
763     }
764     else    {
765           int class, state, prior, status;
766           char name[20];
767           char block[] = "\x1B[34m""Blocked Queue: \n""\x1b[0m";
768           char cname[] = "Name: ";
769           char cclass[] = "Class: ";
770           char cstate[] = "State: ";
771           char cstatus[] = "Status: ";
772           char cprior[] = "Priority: ";
773           char line[] = "\n";
774
775           printf(block);
776           //sys_req(WRITE, COM1, block, &check );
777
778           PCB* pcb = getBlocked()->head;
779
780           if(pcb->next == NULL) {
781             class = pcb->Process_Class;
782                 strcpy(name,pcb->Process_Name);
783                 state = pcb->ReadyState;
784                 status = pcb->SuspendedState;
785                 prior = pcb->Priority;
786
787                 printf(cname);
788                 printf(name);
789                 printf(line);
790
791                 printf(cclass);
792                 if(pcb->Process_Class == 0)  {
```

```
793                 printf("0");
794               }
795               else  {
796                 printf(itoa(class));
797                 //sys_req(WRITE, COM1, itoa(class), &check);
798               }
799               printf(line);
800 
801               printf(cstate);
802               if(pcb->ReadyState == 0)  {
803                 printf("0");
804               }
805               else  {
806                 printf(itoa(state));
807                 //sys_req(WRITE, COM1, itoa(state), &check);
808               }
809               printf(line);
810 
811               printf(cstatus);
812               if(pcb->SuspendedState == 0)  {
813                 printf("0");
814               }
815               else  {
816                 printf(itoa(status));
817                 //sys_req(WRITE, COM1, itoa(status), &check);
818               }
819               printf(line);
820 
821               printf(cprior);
822               if(pcb->Priority == 0)  {
823                 printf("0");
824                 printf("\n\n");
825               }
826               else  {
827                 printf(itoa(prior));
828                 //sys_req(WRITE, COM1, itoa(prior), &check);
829                 printf("\n\n");
830               }
831           }
832           else  {
833             while(pcb != NULL)  {
834               class = pcb->Process_Class;
835                   strcpy(name,pcb->Process_Name);
836                   state = pcb->ReadyState;
837                   status = pcb->SuspendedState;
838                   prior = pcb->Priority;
839 
840                   printf(cname);
841                   printf(name);
842                   printf(line);
843 
844                   printf(cclass);
845                   if(pcb->Process_Class == 0)  {
846                     printf("0");
847                   }
848                   else  {
849                      printf(itoa(class));
850                     //sys_req(WRITE, COM1, itoa(class), &check);
851                   }
852                   printf(line);
853 
854                   printf(cstate);
855                   if(pcb->ReadyState == 0)  {
856                     printf("0");
857                   }
858                   else  {
859                         printf(itoa(state));
860                     //sys_req(WRITE, COM1, itoa(state), &check);
861                   }
862                   printf(line);
863 
864                   printf(cstatus);
865                   if(pcb->SuspendedState == 0)  {
866                     printf("0");
867                   }
868                   else  {
869                     printf(itoa(status));
870                     //sys_req(WRITE, COM1, itoa(status), &check);
871                   }
872                   printf(line);
873 
874                   printf(cprior);
875                   if(pcb->Priority == 0)  {
876                     printf("0");
877                     printf("\n\n");
878                   }
879                   else  {
```

```
880                         printf(itoa(prior));
881                         //sys_req(WRITE, COM1, itoa(prior), &check);
882                         printf("\n\n");
883                     }
884                 pcb = pcb->next;
885             }
886         }
887     }
888 }
```

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 536 of file userFunctions.c.

```
536                                         {
537     if (FindPCB(ProcessName) == NULL)   {
538         printf("\x1b[31m""\nERROR: PCB does not exist \n""\x1b[0m");
539     }
540     else    {
541 
542         char name[10];
543         char cname[] = "Name: ";
544         char cclass[] = "Class: ";
545         char cstate[] = "State: ";
546         char cstatus[] = "Status: ";
547         char cprior[] = "Priority: ";
548         char line[] = "\n";
549         PCB* pcb = FindPCB(ProcessName);
550         strcpy(name,pcb->Process_Name);
551         int class = pcb->Process_Class;
552         int state = pcb->ReadyState;
553         int status = pcb->SuspendedState;
554         int prior = pcb->Priority;
555 
556         if(name == NULL){
557             printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
558         }
559         else    {
560             printf(cname);
561             printf(ProcessName);
562             printf(line);
563             printf(cclass);
564             if(pcb->Process_Class == 0)  {
565                 printf("0");
566             }
567             else  {
568                 printf(itoa(class));
569                 //sys_req(WRITE, COM1, itoa(class), &check);
570             }
571             printf(line);
572             printf(cstate);
573             if(pcb->ReadyState == 0)  {
574                 printf("0");
575             }
576             else  {
577                 printf(itoa(state));
578                 //sys_req(WRITE, COM1, itoa(state), &check);
579             }
580             printf(line);
581             printf(cstatus);
```

```
582             if(pcb->SuspendedState == 0)  {
583                 printf("0");
584             }
585             else  {
586                 printf(itoa(status));
587                 //sys_req(WRITE, COM1, itoa(status), &check);
588             }
589             printf(line);
590             printf(cprior);
591             if(pcb->Priority == 0)  {
592                 printf("0");
593                 printf("\n\n");
594             }
595             else  {
596                 printf(itoa(prior));
597                 //sys_req(WRITE, COM1, itoa(prior), &check);
598                 printf("\n\n");
599             }
600         }
601     }
602 }
```

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 623 of file userFunctions.c.

```
623                 {
624     if(getReady()->head == NULL)    {
625          printf("\x1b[32m""\n The Ready Queue is empty \n""\x1b[0m");
626     }
627     else    {
628         int class, state, prior, status;
629       char name[10];
630       char ready[] = "\x1B[34m""\nReady Queue:\n""\x1B[0m";
631       char cname[] = "Name: ";
632       char cclass[] = "Class: ";
633       char cstate[] = "State: ";
634       char cstatus[] = "Status: ";
635       char cprior[] = "Priority: ";
636       char line[] = "\n";
637
638       printf(ready);
639       //sys_req(WRITE, COM1, ready, &check );
640
641       PCB* pcb = getReady()->head;
642
643         if(pcb->next == NULL)   {
644             class = pcb->Process_Class;
645             strcpy(name,pcb->Process_Name);
646             state = pcb->ReadyState;
647             status = pcb->SuspendedState;
648             prior = pcb->Priority;
649
650             printf(cname);
651             printf(name);
652             printf(line);
653
654             printf(cclass);
655             if(pcb->Process_Class == 0)  {
656               printf("0");
657             }
658             else  {
659               printf(itoa(class));
660               //sys_req(WRITE, COM1, itoa(class), &check);
661             }
662             printf(line);
663
664             printf(cstate);
665             if(pcb->ReadyState == 0)  {
666               printf("0");
```

```
667             }
668             else  {
669               printf(itoa(state));
670               //sys_req(WRITE, COM1, itoa(state), &check);
671             }
672             printf(line);
673 
674             printf(cstatus);
675             if(pcb->SuspendedState == 0)  {
676               printf("0");
677             }
678             else  {
679               printf(itoa(status));
680               //sys_req(WRITE, COM1, itoa(status), &check);
681             }
682             printf(line);
683 
684             printf(cprior);
685             if(pcb->Priority == 0)  {
686               printf("0");
687               printf("\n\n");
688             }
689             else  {
690               printf(itoa(prior));
691               //sys_req(WRITE, COM1, itoa(prior), &check);
692               printf("\n\n");
693             }
694       }
695       else  {
696         while(pcb != NULL)  {
697                 class = pcb->Process_Class;
698                 strcpy(name,pcb->Process_Name);
699                 state = pcb->ReadyState;
700                 status = pcb->SuspendedState;
701                 prior = pcb->Priority;
702 
703                 printf(cname);
704                 printf(name);
705                 printf(line);
706 
707                 printf(cclass);
708                 if(pcb->Process_Class == 0)  {
709                   printf("0");
710                 }
711                 else  {
712                   printf(itoa(class));
713                   //sys_req(WRITE, COM1, itoa(class), &check);
714                 }
715                 printf(line);
716 
717                 printf(cstate);
718                 if(pcb->ReadyState == 0)  {
719                   printf("0");
720                 }
721                 else  {
722                   printf(itoa(state));
723                   //sys_req(WRITE, COM1, itoa(state), &check);
724                 }
725                 printf(line);
726 
727                 printf(cstatus);
728                 if(pcb->SuspendedState == 0)  {
729                   printf("0");
730                 }
731                 else  {
732                   printf(itoa(status));
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                   printf(itoa(prior));
744                   //sys_req(WRITE, COM1, itoa(prior), &check);
745                   printf("\n\n");
746                 }
747                 pcb = pcb->next;
748             }
749         }
750     }
751 }
```

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 458 of file userFunctions.c.

```
458                                 {
459     PCB* pcb = FindPCB(ProcessName);
460     if (pcb == NULL)    {
461       printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
462     }
463     else {
464         if(pcb->SuspendedState == YES)  {
465             printf("\x1b[32m""\nThis Process is already SUSPENDED \n""\x1b[0m");
466         }
467         else if(pcb -> Process_Class == APPLICATION)    {
468             pcb->SuspendedState = YES;
469         }
470         else
471             printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
472     }
473 }
```

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 315 of file userFunctions.c.

```
315                                 {
316     if((c >= 65) && (c <= 90))  {
317         c = c + 32;
318     }
319     return c;
320 }
```

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 984 of file userFunctions.c.

```
984                                 {
985   PCB* pcb = FindPCB(ProcessName);
986   if (pcb == NULL)  {
987     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
988   }
989   else {
990     if(pcb->ReadyState == READY)    {
991         printf("\x1b[32m""\nThis Process is already in the READY state \n""\x1b[0m");
992     }
993     else    {
994       RemovePCB(pcb);
995       pcb->ReadyState = READY;
996       InsertPCB(pcb);
997     }
998   }
999 }
```

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 308 of file userFunctions.c.

```
308                 {
309     printf("Version: R5.2 \n");
310 }
```

4.21.2 Variable Documentation

4.21.2.1 AlarmList

```
List AlarmList
```

Initial value:

```
={
  .head = NULL,
  .tail = NULL
}
```

Definition at line 1046 of file userFunctions.c.

4.22 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/R1/user↩ Functions.h File Reference

Classes

- struct Alarm
- struct List

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"

Typedefs

- typedef struct Alarm **Alarm**
- typedef struct List **List**

Functions

- void SetTime (int hours, int minutes, int seconds)
- void GetTime ()
- int DectoBCD (int Decimal)
- void **clear** ()
- 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)
- void **loader** ()
- void **loadr3** (char *name, u32int func)
- void **yield** ()
- void **loaderinfinite** ()
- List * **getList** ()
- void **loaderalarm** ()

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 82 of file userFunctions.c.

```
82                          {
83     return (((BCD»4)*10) + (BCD & 0xF));
84 }
```

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 958 of file userFunctions.c.

```
958                                 {
959   PCB* pcb = FindPCB(ProcessName);
960   if (pcb == NULL)  {
961     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
962   }
963   else {
964     if(pcb->ReadyState == BLOCKED)  {
965         printf("\x1b[32m""\nThis Process is already BLOCKED \n""\x1b[0m");
966     }
967     else    {
968       RemovePCB(pcb);
969       pcb->ReadyState = BLOCKED;
970       InsertPCB(pcb);
971     }
972   }
973 }
```

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 901 of file userFunctions.c.

```
901                                                                    {
902   if (FindPCB(ProcessName) == NULL) {
903     if(Priority >= 0 && Priority < 10){
904       if(Class == 0 || Class == 1){
905         PCB* pcb = SetupPCB(ProcessName, Class, Priority);
906         InsertPCB(pcb);
907       } else{
908         printf("\x1b[31m""\nERROR: Not a valid Class \n""\x1b[0m");
909       }
910     } else{
911       printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
912     }
913   } else{
914     printf("\x1b[31m""\nERROR: This Process Name already exists \n""\x1b[0m");
915   }
916 }
```

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 89 of file userFunctions.c.

```
89                              {
90     return (((Decimal/10) << 4) | (Decimal % 10));
91 }
```

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 927 of file userFunctions.c.

```
927                                     {
928   PCB* pcb = FindPCB(ProcessName);
929   if (pcb == NULL)  {
930     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
931   }
932   else if(strcmp(pcb->Process_Name,"InfProc") == 0)   {
933         if(pcb->SuspendedState == YES)  {
934             RemovePCB(pcb);
935             FreePCB(pcb);
936         }
937         else
938             printf("\x1b[31m""\nERROR:This process cannot be deleted unless it is in the suspended
       state\n""\x1b[0m");
939   }
940   else if(pcb -> Process_Class == SYSTEM)   {
941       printf("\x1b[31m""\nERROR: System Processes cannot be deleted from the system. \n""\x1b[0m");
942   }
943     else {
944         RemovePCB(pcb);
945         FreePCB(pcb);
946     }
947 }
```

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 110 of file userFunctions.c.

```
110                                 {
111   int valid = 0;
112   if (strcmp(pointer, "00") == 0) {
113     valid = 1;
114     return valid;
115   }
116   else if (strcmp(pointer, "0") == 0) {
117     valid = 1;
118     return valid;
119   }
120   else  {
121     int j;
```

```
122     valid = 0;
123     for(j = 0; j <= 99; j++)  {
124       if(strcmp(pointer,itoa(j)) == 0)
125         valid = 1;
126     }
127     if(valid == 0)  {
128       return valid;
129     }
130   }
131   return valid;
132 }
```

4.22.1.7 GetDate()

```
void GetDate ( )
```

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

No parameters.

Definition at line 272 of file userFunctions.c.

```
272                 {
273
274   outb(0x70,0x07);
275     unsigned char day = BCDtoDec(inb(0x71));
276     outb(0x70,0x08);
277     unsigned char month = BCDtoDec(inb(0x71));
278     outb(0x70,0x32);
279     unsigned char millennium = BCDtoDec(inb(0x71));
280     char msg[2] = "-";
281     char msg3[10] = "Date: ";
282     printf(msg3);
283
284      printf(itoa(day));
285     //sys_req(WRITE, COM1, itoa(day), &check);
286     printf(msg);
287     printf(itoa(month));
288     //sys_req(WRITE, COM1, itoa(month), &check);
289     printf(msg);
290     printf(itoa(millennium));
291     //sys_req(WRITE, COM1, itoa(millennium), &check);
292   outb(0x70,0x09);
293   if(BCDtoDec(inb(0x71)) == 0){
294      printf("00");
295     //sys_req(WRITE, COM1, "00", &check);
296   }
297   else {
298         unsigned char year = BCDtoDec(inb(0x71));
299          printf(itoa(year));
300         //sys_req(WRITE, COM1, itoa(year), &check);
301   }
302     printf("\n");
303 }
```

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 191 of file userFunctions.c.

```
191                 {
192
```

```
193     int hour;
194     int minute;
195     int second;
196     outb(0x70,0x04);
197     unsigned char hours = inb(0x71);
198     outb(0x70,0x02);
199     unsigned char minutes = inb(0x71);
200     outb(0x70,0x00);
201     unsigned char seconds = inb(0x71);
202     char msg1[2] = ":";
203     char msg2[10] = "Time: ";
204     printf(msg2);
205     hour = BCDtoDec(hours);
206     printf(itoa(hour));
207     //sys_req(WRITE, COM1, itoa(hour), &check);
208     printf(msg1);
209     minute = BCDtoDec(minutes);
210     printf(itoa(minute));
211     //sys_req(WRITE, COM1, itoa(minute), &check);
212     printf(msg1);
213     second = BCDtoDec(seconds);
214     printf(itoa(second));
215     //sys_req(WRITE, COM1, itoa(second), &check);
216   printf("\n");
217 }
```

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 332 of file userFunctions.c.

```
332                          {
333     if (request[0] == '\0') {
334         //removed for R3/R4 from active command list
335         //\n createPCB \n block \n unblock
336         //\n heap         alloc \n free         empty
337         printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
338         printf("\n getDate       setDate \n getTime       setTime \n version       suspend \n resume
       setPriority \n showPCB       showAll \n showReady     showBlocked  \n deletePCB     shutdown \n alarm
           clear \n loadr3       infinte \n showFree     showAlloc \n\n");
339     }
340     else if (strcmp(request, "GetDate") == 0) {
341         printf("\n getDate returns the current date that is loaded onto the operating system.\n");
342     }
343     else if (strcmp(request, "SetDate") == 0) {
344         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");
345     }
346     else if (strcmp(request, "GetTime") == 0) {
347         printf("\n getTime returns the current time as hours, minutes, seconds that is loaded onto the
       operating system.\n");
348     }
349     else if (strcmp(request, "SetTime") == 0) {
350         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");
351     }
```

```
352     else if (strcmp(request, "Version") == 0) {
353         printf("\n version returns the current operating software version that the system is
        running.\n");
354     }
355     else if (strcmp(request, "infinte") == 0) {
356         printf("\n infinite Loads the infinite process into the ready queue.\n");
357     }
358     else if (strcmp(request, "loadr3") == 0)  {
359         printf("\n loadr3 Loads in all five of the R3 test processes in a suspended state into the
        queue.\n");
360     }
361     else if (strcmp(request, "alarm") == 0)       {
362         printf("\n alarm creates a user specified alarm with a user set message and time
        alarm-MSG-hour-minute-second.\n");
363     }
364     else if (strcmp(request, "clear") == 0)       {
365         printf("\n clear erases the console of all typed commands and refreshes it with just the command
        list.\n");
366     }
367
368   else if(strcmp(request, "shutdown") == 0)   {
369     printf("\n shutdown shuts down the system.\n");
370   }
371
372
373
      /*****************************************************************************************************
374           R2 Commands
375
      *****************************************************************************************************/
376   else if(strcmp(request,"suspend") == 0) {
377         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");
378   }
379   else if(strcmp(request,"resume") == 0) {
380         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");
381   }
382   else if(strcmp(request,"setPriority") == 0) {
383         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");
384   }
385   else if(strcmp(request,"showPCB") == 0) {
386         printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
        user.\n");
387   }
388   else if(strcmp(request,"showAll") == 0) {
389         printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
        queues.\n");
390   }
391   else if(strcmp(request,"showReady") == 0) {
392         printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
        currently are in the ready state.\n");
393   }
394   else if(strcmp(request,"showBlocked") == 0) {
395         printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
        currently are in the blocked state.\n");
396   }
397
398 /******************************** R2 Temp Commands
      *******************************************************/
399   else if(strcmp(request,"deletePCB") == 0) {
400         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");
401   }
402   //removed for R3/R4 from active command list
403   /*
404   else if(strcmp(request,"createPCB") == 0) {
405         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");
406   }
407   else if(strcmp(request,"block") == 0) {
408         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");
409   }
410   else if(strcmp(request,"unblock") == 0) {
411         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");
412   }
413   */
414
415
416 /******************************** R5 Temp Commands
      *******************************************************/
417   // else if(strcmp(request,"heap") == 0) {
418     //  printf("\n heap initializes the memory heap for the entire system.\n");
```

```
419   // }
420   // else if(strcmp(request,"alloc") == 0) {
421     //  printf("\n alloc allocates the specified amount of memory to the specific process
       (alloc-process_name-size).\n");
422   // }
423   // else if(strcmp(request,"free") == 0) {
424     //  printf("\n free frees the specified memory at the address given (free-address).\n");
425   // }
426   // else if(strcmp(request,"empty") == 0) {
427     //  printf("\n isempty returns true or false depending on if the heap has allocated memory.\n");
428   // }
429 /******************************** R5 Commands
       *******************************************************/
430   else if(strcmp(request,"showFree") == 0) {
431         printf("\n showfree shows all the free blocks avaliable within the heap list.\n");
432   }
433   else if(strcmp(request,"showAlloc") == 0) {
434         printf("\n showAlloc shows all the allocated blocks within the heap list.\n");
435   }
436
437
438   else  {
439         printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
       full list of commands.\n""\x1b[0m");
440   }
441 }
```

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 51 of file userFunctions.c.

```
51                    {
52
53     int i,j,k,count;
54     i = num;
55     j = 0;
56     count = 0;
57     while(i){ // count number of digits
58         count++;
59         i /= 10;
60     }
61
62     char* arr1;
63     char arr2[count];
64     arr1 = (char*)sys_alloc_mem(count); //memory allocation
65
66     while(num){ // seperate last digit from number and add ASCII
67         arr2[++j] = num%10 + '0';
68         num /= 10;
69     }
70
71     for(k = 0; k < j; k++){ // reverse array results
72         arr1[k] = arr2[j-k];
73     }
74     arr1[k] = '\0';
75     sys_free_mem(arr1);
76     return(char*)arr1;
77 }
```

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 484 of file userFunctions.c.

```
484                                 {
485     PCB* pcb = FindPCB(ProcessName);
486     if (pcb == NULL)    {
487       printf(RED"\nERROR: Not a valid process name \n"RESET);
488     }
489     else {
490         if(pcb->SuspendedState == NO)   {
491             printf(GRN"\nThis Process is already in the NONSUSPENDED state \n"RESET);
492         }
493         else if(pcb -> Process_Class == APPLICATION)    {
494             pcb->SuspendedState = NO;
495         }
496         else
497             printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
498     }
499 }
```

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 511 of file userFunctions.c.

```
511                                                 {
512     PCB* pcb = FindPCB(ProcessName);
513     if (pcb == NULL)    {
514         printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
515     }
516     else if(Priority >= 10){
517         printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
518     }
519     else if(pcb -> Process_Class == APPLICATION) {
```

```
520         RemovePCB(pcb);
521         pcb->Priority = Priority;
522         InsertPCB(pcb);
523     }
524     else
525         printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
526 }
```

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 225 of file userFunctions.c.

```
225                                                                 {
226   outb(0x70,0x07);
227   int tempDay = BCDtoDec(inb(0x71));
228   outb(0x70,0x08);
229   int tempMonth = BCDtoDec(inb(0x71));
230   outb(0x70,0x32);
231   int tempMillennium = BCDtoDec(inb(0x71));
232   outb(0x70,0x09);
233   int tempYear = BCDtoDec(inb(0x71));
234   cli();
235     outb(0x70,0x07);
236     outb(0x71,DectoBCD (day));
237     outb(0x70,0x08);
238     outb(0x71,DectoBCD (month));
239     outb(0x70,0x32);
240     outb(0x71,DectoBCD (millennium));
241     outb(0x70,0x09);
242     outb(0x71,DectoBCD (year));
243     sti();
244   outb(0x70,0x07);
245   unsigned char newDay = BCDtoDec(inb(0x71));
246   outb(0x70,0x08);
247   unsigned char newMonth = BCDtoDec(inb(0x71));
248   outb(0x70,0x32);
249   unsigned char newMillennium = BCDtoDec(inb(0x71));
250   outb(0x70,0x09);
251   unsigned char newYear = BCDtoDec(inb(0x71));
252   if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
253     printf("Your input was invalid\n");
254     cli();
255         outb(0x70,0x07);
256         outb(0x71,DectoBCD (tempDay));
257         outb(0x70,0x08);
258         outb(0x71,DectoBCD (tempMonth));
259         outb(0x70,0x32);
260         outb(0x71,DectoBCD (tempMillennium));
261         outb(0x70,0x09);
262         outb(0x71,DectoBCD (tempYear));
263         sti();
264   }
265   else
```

```
266     printf("Date Set\n");
267 }
```

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 152 of file userFunctions.c.

```
152                                                 {
153   outb(0x70,0x04);
154   unsigned char tempHours = BCDtoDec(inb(0x71));
155   outb(0x70,0x02);
156   unsigned char tempMinutes = BCDtoDec(inb(0x71));
157   outb(0x70,0x00);
158   unsigned char tempSeconds = BCDtoDec(inb(0x71));
159     cli(); //outb(device + 1, 0x00); //disable interrupts
160     outb(0x70,0x04);
161     outb(0x71, DectoBCD(hours));// change to bcd
162     outb(0x70,0x02);
163     outb(0x71, DectoBCD(minutes));
164     outb(0x70,0x00);
165     outb(0x71, DectoBCD(seconds));
166     sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
167   outb(0x70,0x04);
168   unsigned char newHours = BCDtoDec(inb(0x71));
169   outb(0x70,0x02);
170   unsigned char newMinutes = BCDtoDec(inb(0x71));
171   outb(0x70,0x00);
172   unsigned char newSeconds = BCDtoDec(inb(0x71));
173   if(newHours != hours || newMinutes != minutes || newSeconds != seconds){
174     printf("Your input was invalid\n");
175     cli(); //outb(device + 1, 0x00); //disable interrupts
176         outb(0x70,0x04);
177         outb(0x71, DectoBCD(tempHours));// change to bcd
178         outb(0x70,0x02);
179         outb(0x71, DectoBCD(tempMinutes));
180         outb(0x70,0x00);
181         outb(0x71, DectoBCD(tempSeconds));
182         sti();  //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
183   }
184   else
185     printf("Time Set\n");
186 }
```

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 611 of file userFunctions.c.

```
611                 {
612     Show_Ready();
613     Show_Blocked();
614 }
```

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 760 of file userFunctions.c.

```
760                     {
761     if(getBlocked()->head == NULL)  {
762          printf("\x1b[32m""\n The Blocked Queue is empty \n""\x1b[0m");
763     }
764     else    {
765          int class, state, prior, status;
766          char name[20];
767          char block[] = "\x1B[34m""Blocked Queue: \n""\x1b[0m";
768          char cname[] = "Name: ";
769          char cclass[] = "Class: ";
770          char cstate[] = "State: ";
771          char cstatus[] = "Status: ";
772          char cprior[] = "Priority: ";
773          char line[] = "\n";
774
775          printf(block);
776          //sys_req(WRITE, COM1, block, &check );
777
778          PCB* pcb = getBlocked()->head;
779
780          if(pcb->next == NULL) {
781            class = pcb->Process_Class;
782               strcpy(name,pcb->Process_Name);
783               state = pcb->ReadyState;
784               status = pcb->SuspendedState;
785               prior = pcb->Priority;
786
787               printf(cname);
788               printf(name);
789               printf(line);
790
791               printf(cclass);
792               if(pcb->Process_Class == 0)  {
793                 printf("0");
794               }
795               else  {
796                 printf(itoa(class));
797                 //sys_req(WRITE, COM1, itoa(class), &check);
798               }
799               printf(line);
800
801               printf(cstate);
802               if(pcb->ReadyState == 0)  {
803                 printf("0");
804               }
805               else  {
806                 printf(itoa(state));
807                 //sys_req(WRITE, COM1, itoa(state), &check);
808               }
809               printf(line);
810
811               printf(cstatus);
812               if(pcb->SuspendedState == 0)  {
813                 printf("0");
814               }
815               else  {
```

```
816                   printf(itoa(status));
817                   //sys_req(WRITE, COM1, itoa(status), &check);
818                 }
819                 printf(line);
820 
821                 printf(cprior);
822                 if(pcb->Priority == 0)  {
823                   printf("0");
824                   printf("\n\n");
825                 }
826                 else  {
827                   printf(itoa(prior));
828                   //sys_req(WRITE, COM1, itoa(prior), &check);
829                   printf("\n\n");
830                 }
831           }
832           else  {
833             while(pcb != NULL)  {
834                 class = pcb->Process_Class;
835                     strcpy(name,pcb->Process_Name);
836                     state = pcb->ReadyState;
837                     status = pcb->SuspendedState;
838                     prior = pcb->Priority;
839 
840                     printf(cname);
841                     printf(name);
842                     printf(line);
843 
844                     printf(cclass);
845                     if(pcb->Process_Class == 0)  {
846                       printf("0");
847                     }
848                     else  {
849                        printf(itoa(class));
850                       //sys_req(WRITE, COM1, itoa(class), &check);
851                     }
852                     printf(line);
853 
854                     printf(cstate);
855                     if(pcb->ReadyState == 0)  {
856                       printf("0");
857                     }
858                     else  {
859                           printf(itoa(state));
860                       //sys_req(WRITE, COM1, itoa(state), &check);
861                     }
862                     printf(line);
863 
864                     printf(cstatus);
865                     if(pcb->SuspendedState == 0)  {
866                       printf("0");
867                     }
868                     else  {
869                       printf(itoa(status));
870                       //sys_req(WRITE, COM1, itoa(status), &check);
871                     }
872                     printf(line);
873 
874                     printf(cprior);
875                     if(pcb->Priority == 0)  {
876                       printf("0");
877                       printf("\n\n");
878                     }
879                     else  {
880                       printf(itoa(prior));
881                       //sys_req(WRITE, COM1, itoa(prior), &check);
882                       printf("\n\n");
883                     }
884                 pcb = pcb->next;
885               }
886           }
887     }
888 }
```

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 536 of file userFunctions.c.

```
536                                 {
537     if (FindPCB(ProcessName) == NULL)   {
538         printf("\x1b[31m""\nERROR: PCB does not exist \n""\x1b[0m");
539     }
540     else    {
541 
542         char name[10];
543         char cname[] = "Name: ";
544         char cclass[] = "Class: ";
545         char cstate[] = "State: ";
546         char cstatus[] = "Status: ";
547         char cprior[] = "Priority: ";
548         char line[] = "\n";
549         PCB* pcb = FindPCB(ProcessName);
550         strcpy(name,pcb->Process_Name);
551         int class = pcb->Process_Class;
552         int state = pcb->ReadyState;
553         int status = pcb->SuspendedState;
554         int prior = pcb->Priority;
555 
556         if(name == NULL){
557             printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
558         }
559         else    {
560             printf(cname);
561             printf(ProcessName);
562             printf(line);
563             printf(cclass);
564             if(pcb->Process_Class == 0)  {
565                 printf("0");
566             }
567             else  {
568                 printf(itoa(class));
569                 //sys_req(WRITE, COM1, itoa(class), &check);
570             }
571             printf(line);
572             printf(cstate);
573             if(pcb->ReadyState == 0)  {
574                 printf("0");
575             }
576             else  {
577                 printf(itoa(state));
578                 //sys_req(WRITE, COM1, itoa(state), &check);
579             }
580             printf(line);
581             printf(cstatus);
582             if(pcb->SuspendedState == 0)  {
583                 printf("0");
584             }
585             else  {
586                 printf(itoa(status));
587                 //sys_req(WRITE, COM1, itoa(status), &check);
588             }
589             printf(line);
590             printf(cprior);
591             if(pcb->Priority == 0)  {
592                 printf("0");
593                 printf("\n\n");
594             }
595             else  {
596                 printf(itoa(prior));
597                 //sys_req(WRITE, COM1, itoa(prior), &check);
598                 printf("\n\n");
599             }
600         }
601     }
602 }
```

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 623 of file userFunctions.c.

```
623                     {
624     if(getReady()->head == NULL)    {
625          printf("\x1b[32m""\n The Ready Queue is empty \n""\x1b[0m");
626     }
627     else    {
628         int class, state, prior, status;
629       char name[10];
630       char ready[] = "\x1B[34m""\nReady Queue:\n""\x1B[0m";
631       char cname[] = "Name: ";
632       char cclass[] = "Class: ";
633       char cstate[] = "State: ";
634       char cstatus[] = "Status: ";
635       char cprior[] = "Priority: ";
636       char line[] = "\n";
637
638       printf(ready);
639       //sys_req(WRITE, COM1, ready, &check );
640
641       PCB* pcb = getReady()->head;
642
643         if(pcb->next == NULL)   {
644             class = pcb->Process_Class;
645             strcpy(name,pcb->Process_Name);
646             state = pcb->ReadyState;
647             status = pcb->SuspendedState;
648             prior = pcb->Priority;
649
650             printf(cname);
651             printf(name);
652             printf(line);
653
654             printf(cclass);
655             if(pcb->Process_Class == 0)  {
656               printf("0");
657             }
658             else  {
659               printf(itoa(class));
660               //sys_req(WRITE, COM1, itoa(class), &check);
661             }
662             printf(line);
663
664             printf(cstate);
665             if(pcb->ReadyState == 0)  {
666               printf("0");
667             }
668             else  {
669               printf(itoa(state));
670               //sys_req(WRITE, COM1, itoa(state), &check);
671             }
672             printf(line);
673
674             printf(cstatus);
675             if(pcb->SuspendedState == 0)  {
676               printf("0");
677             }
678             else  {
679               printf(itoa(status));
680               //sys_req(WRITE, COM1, itoa(status), &check);
681             }
682             printf(line);
683
684             printf(cprior);
685             if(pcb->Priority == 0)  {
686               printf("0");
687               printf("\n\n");
688             }
689             else  {
690               printf(itoa(prior));
691               //sys_req(WRITE, COM1, itoa(prior), &check);
692               printf("\n\n");
693             }
694       }
```

```
695       else  {
696         while(pcb != NULL)  {
697                 class = pcb->Process_Class;
698                 strcpy(name,pcb->Process_Name);
699                 state = pcb->ReadyState;
700                 status = pcb->SuspendedState;
701                 prior = pcb->Priority;
702 
703                 printf(cname);
704                 printf(name);
705                 printf(line);
706 
707                 printf(cclass);
708                 if(pcb->Process_Class == 0)  {
709                   printf("0");
710                 }
711                 else  {
712                   printf(itoa(class));
713                   //sys_req(WRITE, COM1, itoa(class), &check);
714                 }
715                 printf(line);
716 
717                 printf(cstate);
718                 if(pcb->ReadyState == 0)  {
719                   printf("0");
720                 }
721                 else  {
722                   printf(itoa(state));
723                   //sys_req(WRITE, COM1, itoa(state), &check);
724                 }
725                 printf(line);
726 
727                 printf(cstatus);
728                 if(pcb->SuspendedState == 0)  {
729                   printf("0");
730                 }
731                 else  {
732                   printf(itoa(status));
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                   printf(itoa(prior));
744                   //sys_req(WRITE, COM1, itoa(prior), &check);
745                   printf("\n\n");
746                 }
747                 pcb = pcb->next;
748             }
749         }
750     }
751 }
```

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 458 of file userFunctions.c.

```
458                                 {
459     PCB* pcb = FindPCB(ProcessName);
460     if (pcb == NULL)    {
461       printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
462     }
463     else {
464         if(pcb->SuspendedState == YES)  {
465             printf("\x1b[32m""\nThis Process is already SUSPENDED \n""\x1b[0m");
466         }
467         else if(pcb -> Process_Class == APPLICATION)    {
468             pcb->SuspendedState = YES;
469         }
470         else
471             printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
472     }
473 }
```

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 315 of file userFunctions.c.

```
315                                 {
316     if((c >= 65) && (c <= 90))  {
317         c = c + 32;
318     }
319     return c;
320 }
```

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 984 of file userFunctions.c.

```
984                                 {
985   PCB* pcb = FindPCB(ProcessName);
986   if (pcb == NULL)  {
987     printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
```

```
988   }
989   else {
990     if(pcb->ReadyState == READY)    {
991         printf("\x1b[32m""\nThis Process is already in the READY state \n""\x1b[0m");
992     }
993     else    {
994       RemovePCB(pcb);
995       pcb->ReadyState = READY;
996       InsertPCB(pcb);
997     }
998   }
999 }
```

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 308 of file userFunctions.c.

```
308                 {
309     printf("Version: R5.2 \n");
310 }
```

4.23 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/sys_proc_loader.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 "R1/userFunctions.h"
#include "procsr3.h"
#include "R1/comHand.h"
#include "sys_proc_loader.h"
```

Functions

- void **sysLoader** ()
- void **loadSysProc** (char *name, u32int func, int priority)
- void **InfiniteProc** ()
- void **AlarmProc** ()

4.24 D:/GITHUB/CS_450_RunTime_Terror/mpx_core/modules/sys_proc_loader.h File Reference

Functions

- void **sysLoader** ()
- void **loadSysProc** (char *name, u32int func, int priority)
- void **InfiniteProc** ()
- void **AlarmProc** ()

Index