# Runtime Terror OS R3/R4

Generated by Doxygen 1.9.1

1	Class Index	1
	1.1 Class List	1
2	File Index	3
	2.1 File List	3
3	Class Documentation	5
	3.1 Alarm Struct Reference	5
	3.1.1 Detailed Description	5
	3.2 context Struct Reference	5
	3.2.1 Detailed Description	6
	3.3 date_time Struct Reference	6
	3.3.1 Detailed Description	6
	3.4 footer Struct Reference	6
	3.4.1 Detailed Description	6
	3.5 gdt_descriptor_struct Struct Reference	7
	3.5.1 Detailed Description	7
	3.6 gdt_entry_struct Struct Reference	7
	3.6.1 Detailed Description	7
	3.7 header Struct Reference	7
	3.7.1 Detailed Description	8
	3.8 heap Struct Reference	8
	3.8.1 Detailed Description	8
	3.9 idt_entry_struct Struct Reference	8
	3.9.1 Detailed Description	8
	3.10 idt_struct Struct Reference	9
	3.10.1 Detailed Description	9
	3.11 index_entry Struct Reference	9
	3.11.1 Detailed Description	9
	3.12 index_table Struct Reference	9
	3.12.1 Detailed Description	9
	3.13 List Struct Reference	10
	3.13.1 Detailed Description	10
	3.14 page_dir Struct Reference	10
	3.14.1 Detailed Description	10
	3.15 page_entry Struct Reference	10
	3.15.1 Detailed Description	11
	3.16 page_table Struct Reference	11
	3.16.1 Detailed Description	11
	3.17 param Struct Reference	11
	3.17.1 Detailed Description	11
	3.18 PCB Struct Reference	12
	3.18.1 Detailed Description	12

3.19 Queue Struct Reference	. 12
3.19.1 Detailed Description	. 12
4 File Documentation	13
4.1 include/core/asm.h File Reference	. 13
4.2 include/core/interrupts.h File Reference	. 13
4.3 include/core/io.h File Reference	. 13
4.3.1 Macro Definition Documentation	. 13
4.3.1.1 inb	. 13
4.4 include/core/serial.h File Reference	. 14
4.5 include/core/tables.h File Reference	. 14
4.6 include/mem/heap.h File Reference	. 15
4.7 include/mem/paging.h File Reference	. 15
4.8 include/string.h File Reference	. 16
4.8.1 Function Documentation	. 16
4.8.1.1 atoi()	. 16
4.8.1.2 isspace()	. 17
4.8.1.3 memset()	. 17
4.8.1.4 strcat()	. 17
4.8.1.5 strcmp()	. 18
4.8.1.6 strcpy()	. 18
4.8.1.7 strlen()	. 19
4.8.1.8 strtok()	. 19
4.9 include/system.h File Reference	. 20
4.10 kernel/core/interrupts.c File Reference	. 21
4.11 kernel/core/kmain.c File Reference	. 22
4.12 kernel/core/serial.c File Reference	. 22
4.13 kernel/core/system.c File Reference	. 23
4.14 kernel/core/tables.c File Reference	. 23
4.15 kernel/mem/heap.c File Reference	. 24
4.16 kernel/mem/paging.c File Reference	. 24
4.17 lib/string.c File Reference	. 25
4.17.1 Function Documentation	. 25
4.17.1.1 atoi()	. 25
4.17.1.2 isspace()	. 26
4.17.1.3 memset()	. 26
4.17.1.4 strcat()	. 27
4.17.1.5 strcmp()	. 27
4.17.1.6 strcpy()	. 28
4.17.1.7 strlen()	. 28
4.17.1.8 strtok()	. 28
4.18 modules/mpx_supt.c File Reference	. 29

4.19 modules/mpx_supt.h File Reference	30
4.20 modules/R1/comHand.h File Reference	31
4.20.1 Function Documentation	31
4.20.1.1 comHand()	31
4.21 modules/R1/userFunctions.c File Reference	35
4.21.1 Function Documentation	36
4.21.1.1 BCDtoDec()	36
4.21.1.2 Block()	36
4.21.1.3 Create_PCB()	37
4.21.1.4 DectoBCD()	37
4.21.1.5 Delete_PCB()	38
4.21.1.6 EdgeCase()	38
4.21.1.7 GetDate()	39
4.21.1.8 GetTime()	39
4.21.1.9 Help()	40
4.21.1.10 itoa()	42
4.21.1.11 Resume()	42
4.21.1.12 Set_Priority()	43
4.21.1.13 SetDate()	43
4.21.1.14 SetTime()	44
4.21.1.15 Show_All()	45
4.21.1.16 Show_Blocked()	45
4.21.1.17 Show_PCB()	47
4.21.1.18 Show_Ready()	48
4.21.1.19 Suspend()	50
4.21.1.20 toLowercase()	50
4.21.1.21 Unblock()	51
4.21.1.22 Version()	51
4.21.2 Variable Documentation	51
4.21.2.1 AlarmList	52
4.22 modules/R1/userFunctions.h File Reference	52
4.22.1 Function Documentation	53
4.22.1.1 BCDtoDec()	53
4.22.1.2 Block()	53
4.22.1.3 Create_PCB()	54
4.22.1.4 DectoBCD()	55
4.22.1.5 Delete_PCB()	55
4.22.1.6 EdgeCase()	55
4.22.1.7 GetDate()	56
4.22.1.8 GetTime()	57
4.22.1.9 Help()	57
4.22.1.10 itoa()	59

4.22.1.11 Resume()	 	 59
4.22.1.12 Set_Priority()	 	 60
4.22.1.13 SetDate()	 	 60
4.22.1.14 SetTime()	 	 61
4.22.1.15 Show_All()	 	 62
4.22.1.16 Show_Blocked()	 	 63
4.22.1.17 Show_PCB()	 	 64
4.22.1.18 Show_Ready()	 	 65
4.22.1.19 Suspend()	 	 67
4.22.1.20 toLowercase()	 	 67
4.22.1.21 Unblock()	 	 68
4.22.1.22 Version()	 	 68
4.23 modules/sys_proc_loader.c File Reference .	 	 69
4.24 modules/sys_proc_loader.h File Reference .	 	 69
Index		71

# **Chapter 1**

# **Class Index**

# 1.1 Class List

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

Alarm	5
context	5
date_time	6
footer	6
gdt_descriptor_struct	7
gdt_entry_struct	7
header	7
heap	
idt_entry_struct	8
idt_struct	
index_entry	9
index_table	9
List	
page_dir	
page_entry	10
page_table	
param	11
PCB	12
Queue	12

2 Class Index

# Chapter 2

# File Index

# 2.1 File List

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

include/string.h	16
include/system.h	20
include/core/asm.h	13
include/core/interrupts.h	13
include/core/io.h	13
include/core/serial.h	14
	14
include/mem/heap.h	15
include/mem/paging.h	15
kernel/core/interrupts.c	21
kernel/core/kmain.c	22
kernel/core/serial.c	22
<b>,</b>	23
kernel/core/tables.c	23
	24
	24
lib/string.c	25
	29
modules/mpx_supt.h	30
	??
<b></b>	??
modules/sys_proc_loader.c	69
	69
modules/R1/comHand.c	??
modules/R1/comHand.h	31
modules/R1/userFunctions.c	35
	52
modules/R2/ <b>PCB.c</b>	??
modules/R2/ <b>PCB.h</b>	??

File Index

# **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:

• modules/R1/userFunctions.h

# 3.2 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

6 Class Documentation

# 3.2.1 Detailed Description

Definition at line 34 of file PCB.h.

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

· modules/R2/PCB.h

# 3.3 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

## 3.3.1 Detailed Description

Definition at line 32 of file system.h.

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

· include/system.h

## 3.4 footer Struct Reference

## **Public Attributes**

· header head

# 3.4.1 Detailed Description

Definition at line 18 of file heap.h.

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

· include/mem/heap.h

# 3.5 gdt\_descriptor\_struct Struct Reference

## **Public Attributes**

- u16int limit
- u32int base

# 3.5.1 Detailed Description

Definition at line 25 of file tables.h.

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

• include/core/tables.h

# 3.6 gdt\_entry\_struct Struct Reference

## **Public Attributes**

- u16int limit\_low
- u16int base\_low
- u8int base\_mid
- u8int access
- u8int flags
- u8int base\_high

# 3.6.1 Detailed Description

Definition at line 32 of file tables.h.

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

• include/core/tables.h

# 3.7 header Struct Reference

#### **Public Attributes**

- int size
- int index\_id

8 Class Documentation

# 3.7.1 Detailed Description

Definition at line 13 of file heap.h.

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

• include/mem/heap.h

# 3.8 heap Struct Reference

#### **Public Attributes**

- index\_table index
- u32int base
- u32int max size
- u32int min\_size

# 3.8.1 Detailed Description

Definition at line 35 of file heap.h.

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

• include/mem/heap.h

# 3.9 idt\_entry\_struct Struct Reference

#### **Public Attributes**

- u16int base\_low
- u16int sselect
- u8int zero
- u8int flags
- u16int base\_high

# 3.9.1 Detailed Description

Definition at line 8 of file tables.h.

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

• include/core/tables.h

# 3.10 idt struct Struct Reference

#### **Public Attributes**

- u16int limit
- u32int base

# 3.10.1 Detailed Description

Definition at line 18 of file tables.h.

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

· include/core/tables.h

# 3.11 index\_entry Struct Reference

#### **Public Attributes**

- int size
- · int empty
- u32int block

# 3.11.1 Detailed Description

Definition at line 22 of file heap.h.

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

· include/mem/heap.h

# 3.12 index\_table Struct Reference

## **Public Attributes**

- index entry table [TABLE SIZE]
- int id

## 3.12.1 Detailed Description

Definition at line 29 of file heap.h.

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

include/mem/heap.h

10 Class Documentation

## 3.13 List Struct Reference

## **Public Attributes**

- Alarm \* head
- Alarm \* tail

# 3.13.1 Detailed Description

Definition at line 24 of file userFunctions.h.

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

• modules/R1/userFunctions.h

# 3.14 page\_dir Struct Reference

#### **Public Attributes**

- page\_table \* tables [1024]
- u32int tables\_phys [1024]

# 3.14.1 Detailed Description

Definition at line 36 of file paging.h.

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

· include/mem/paging.h

# 3.15 page\_entry Struct Reference

#### **Public Attributes**

u32int present: 1
u32int writeable: 1
u32int usermode: 1
u32int accessed: 1
u32int dirty: 1

u32int reserved: 7u32int frameaddr: 20

# 3.15.1 Detailed Description

Definition at line 14 of file paging.h.

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

• include/mem/paging.h

# 3.16 page table Struct Reference

## **Public Attributes**

• page\_entry pages [1024]

## 3.16.1 Detailed Description

Definition at line 28 of file paging.h.

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

• include/mem/paging.h

# 3.17 param Struct Reference

## **Public Attributes**

- int op\_code
- · int device id
- char \* buffer\_ptr
- int \* count\_ptr

## 3.17.1 Detailed Description

Definition at line 34 of file mpx\_supt.h.

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

• modules/mpx\_supt.h

12 Class Documentation

# 3.18 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

# 3.18.1 Detailed Description

Definition at line 15 of file PCB.h.

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

· modules/R2/PCB.h

## 3.19 Queue Struct Reference

## **Public Attributes**

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

# 3.19.1 Detailed Description

Definition at line 27 of file PCB.h.

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

· modules/R2/PCB.h

# **Chapter 4**

# **File Documentation**

# 4.1 include/core/asm.h File Reference

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

# 4.2 include/core/interrupts.h File Reference

#### **Functions**

- void init\_irq (void)
- void init\_pic (void)

# 4.3 include/core/io.h File Reference

## **Macros**

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

#### 4.3.1 Macro Definition Documentation

#### 4.3.1.1 inb

Definition at line 17 of file io.h.

# 4.4 include/core/serial.h File Reference

#### **Macros**

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

#### **Functions**

- int init\_serial (int device)
- int serial\_println (const char \*msg)
- int serial\_print (const char \*msg)
- int set\_serial\_out (int device)
- int **set\_serial\_in** (int device)
- int \* polling (char \*buffer, int \*count)

## 4.5 include/core/tables.h File Reference

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

#### Classes

- struct idt\_entry\_struct
- struct idt struct
- struct gdt\_descriptor\_struct
- struct gdt\_entry\_struct

#### **Functions**

- struct idt\_entry\_struct \_\_attribute\_\_ ((packed)) idt\_entry
- void idt\_set\_gate (u8int idx, u32int base, u16int sel, u8int flags)
- void gdt\_init\_entry (int idx, u32int base, u32int limit, u8int access, u8int flags)
- void init\_idt ()
- void init\_gdt ()

#### **Variables**

- u16int base\_low
- u16int sselect
- u8int zero
- u8int flags
- u16int base\_high
- u16int limit
- u32int base
- u16int limit\_low
- u8int base mid
- · u8int access

# 4.6 include/mem/heap.h File Reference

#### **Classes**

- struct header
- struct footer
- · struct index\_entry
- struct index\_table
- struct heap

#### **Macros**

- #define TABLE SIZE 0x1000
- #define KHEAP\_BASE 0xD000000
- #define KHEAP\_MIN 0x10000
- #define KHEAP\_SIZE 0x1000000

#### **Functions**

- u32int \_kmalloc (u32int size, int align, u32int \*phys\_addr)
- u32int kmalloc (u32int size)
- u32int kfree ()
- void init\_kheap ()
- u32int alloc (u32int size, heap \*hp, int align)
- heap \* make\_heap (u32int base, u32int max, u32int min)

# 4.7 include/mem/paging.h File Reference

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

## Classes

- struct page\_entry
- · struct page table
- struct page\_dir

#### **Macros**

• #define PAGE\_SIZE 0x1000

#### **Functions**

- void set\_bit (u32int addr)
- void clear\_bit (u32int addr)
- u32int get\_bit (u32int addr)
- u32int first\_free ()
- void init\_paging ()
- void load\_page\_dir (page\_dir \*new\_page\_dir)
- page\_entry \* get\_page (u32int addr, page\_dir \*dir, int make\_table)
- void new\_frame (page\_entry \*page)

# 4.8 include/string.h File Reference

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

#### **Functions**

```
• int isspace (const char *c)
```

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

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

## 4.8.1 Function Documentation

#### 4.8.1.1 atoi()

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

Description: Convert an ASCII string to an integer

#### **Parameters**

```
s String
```

Definition at line 50 of file string.c.

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

### 4.8.1.2 isspace()

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

Description: Determine if a character is whitespace.

#### **Parameters**

```
c character to check
```

Definition at line 121 of file string.c.

#### 4.8.1.3 memset()

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

Description: Set a region of memory.

#### **Parameters**

s	destination
С	byte to write
n	count

Definition at line 139 of file string.c.

#### 4.8.1.4 strcat()

Description: Concatenate the contents of one string onto another.

#### **Parameters**

s1	destination
s2	source

Definition at line 108 of file string.c.

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

## 4.8.1.5 strcmp()

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

Description: String comparison

#### **Parameters**

s1	string 1
s2	string 2

Definition at line 81 of file string.c.

#### 4.8.1.6 strcpy()

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

Description: Copy one string to another.

#### **Parameters**

s1	destination
s2	source

Definition at line 38 of file string.c.

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

#### 4.8.1.7 strlen()

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

Description: Returns the length of a string.

#### **Parameters**

```
s input string
```

#### Definition at line 26 of file string.c.

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

## 4.8.1.8 strtok()

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

Description: Split string into tokens

### Parameters

s1	String
s2	delimiter

#### Definition at line 153 of file string.c.

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

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

# 4.9 include/system.h File Reference

#### **Classes**

· struct date\_time

#### **Macros**

- #define NULL 0
- #define **no\_warn**(p) if (p) while (1) break
- #define asm \_\_asm\_\_
- #define volatile \_\_volatile\_
- #define sti() asm volatile ("sti"::)
- #define cli() asm volatile ("cli"::)
- #define **nop**() asm volatile ("nop"::)
- #define **hlt**() asm volatile ("hlt"::)
- #define iret() asm volatile ("iret"::)
- #define GDT\_CS\_ID 0x01
- #define GDT\_DS\_ID 0x02

## **Typedefs**

- typedef unsigned int size\_t
- typedef unsigned char u8int
- · typedef unsigned short u16int
- typedef unsigned long u32int

## **Functions**

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

# 4.10 kernel/core/interrupts.c File Reference

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

#### **Macros**

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

## **Functions**

- void divide\_error ()
- void debug ()
- void **nmi** ()
- void breakpoint ()
- · void overflow ()
- void bounds ()
- void invalid\_op ()
- void device\_not\_available ()
- void double\_fault ()
- void coprocessor\_segment ()
- void invalid\_tss ()
- void segment\_not\_present ()
- void stack\_segment()
- void general\_protection ()
- void page\_fault ()
- · void reserved ()
- void coprocessor ()
- void rtc\_isr ()
- void 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 kernel/core/kmain.c File Reference

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

#### **Functions**

· void kmain (void)

## 4.12 kernel/core/serial.c File Reference

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

#### **Macros**

• #define NO\_ERROR 0

#### **Functions**

- int init\_serial (int device)
- int serial\_println (const char \*msg)
- int serial\_print (const char \*msg)
- int set\_serial\_out (int device)
- int set\_serial\_in (int device)
- int \* polling (char \*cmdBuffer, int \*count)

#### **Variables**

- int serial\_port\_out = 0
- int serial\_port\_in = 0

# 4.13 kernel/core/system.c File Reference

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

#### **Functions**

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

# 4.14 kernel/core/tables.c File Reference

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

#### **Functions**

- void write\_gdt\_ptr (u32int, size\_t)
- void write\_idt\_ptr (u32int)
- void idt\_set\_gate (u8int idx, u32int base, u16int sel, u8int flags)
- void init\_idt ()
- void **gdt\_init\_entry** (int idx, u32int base, u32int limit, u8int access, u8int flags)
- void init\_gdt ()

#### **Variables**

- gdt\_descriptor gdt\_ptr
- gdt\_entry gdt\_entries [5]
- idt\_descriptor idt\_ptr
- idt\_entry idt\_entries [256]

# 4.15 kernel/mem/heap.c File Reference

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

#### **Functions**

- u32int \_kmalloc (u32int size, int page\_align, u32int \*phys\_addr)
- u32int kmalloc (u32int size)
- u32int alloc (u32int size, heap \*h, int align)
- heap \* make\_heap (u32int base, u32int max, u32int min)

#### **Variables**

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

# 4.16 kernel/mem/paging.c File Reference

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

#### **Functions**

```
    void set_bit (u32int addr)
```

- void clear\_bit (u32int addr)
- u32int **get\_bit** (u32int addr)
- u32int find\_free ()
- page\_entry \* get\_page (u32int addr, page\_dir \*dir, int make\_table)
- void init\_paging ()
- void load\_page\_dir (page\_dir \*new\_dir)
- void **new\_frame** (page\_entry \*page)

#### **Variables**

```
• u32int mem_size = 0x4000000
```

- u32int **page\_size** = 0x1000
- · u32int nframes
- u32int \* frames
- page\_dir \* **kdir** = 0
- page\_dir \* cdir = 0
- u32int phys\_alloc\_addr
- heap \* kheap

# 4.17 lib/string.c File Reference

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

## **Functions**

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

# 4.17.1 Function Documentation

#### 4.17.1.1 atoi()

```
int atoi ( \label{eq:const_char} \mbox{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
      while(isspace(&c)){ ++s; c = *s;} // advance past whitespace
58
59
60
       if (*s == '-' \mid | *s == '+') sign = *(s++); // save the sign
61
62
      while(*s != '\0') {
    charVal = *s - 48;
res = res * 10 + charVal;
64
65
66
67
       s++;
68
69
70
71
72
     if ( sign == '-') res=res * -1;
73
    return res; // return integer
```

#### 4.17.1.2 isspace()

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

Description: Determine if a character is whitespace.

#### **Parameters**

```
character to check
```

Definition at line 121 of file string.c.

```
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
С	byte to write
n	count

Definition at line 139 of file string.c.

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

#### 4.17.1.4 strcat()

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

Description: String comparison

#### **Parameters**

s1	string 1
s2	string 2

Definition at line 81 of file string.c.

## 4.17.1.6 strcpy()

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

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

Description: Split string into tokens

#### **Parameters**

s1	String
s2	delimiter

#### Definition at line 153 of file string.c.

```
154 {
      static char *tok_tmp = NULL;
156
      const char *p = s2;
157
      //new string
if (s1!=NULL) {
158
159
160
       tok\_tmp = s1;
161
      //old string cont'd
163
      if (tok_tmp==NULL) {
164
        return NULL;

165
166
167
        s1 = tok_tmp;
168
169
170
      //skip leading s2 characters
      while ( *p && *s1 ) {
  if (*s1==*p) {
171
172
        ++s1;
p = s2;
173
174
175
          continue;
176
177
        ++p;
      }
178
179
      //no more to parse
181
      if (!*s1){
     return (tok_tmp = NULL);
}
182
183
184
      //skip non-s2 characters
185
186
      tok\_tmp = s1;
      while (*tok_tmp) {
      p = s2;
188
       while (*p) {
   if (*tok_tmp==*p++) {
   *tok_tmp++ = '\0';
189
190
191
192
       return s1;
193
         }
194
195
        ++tok_tmp;
196
197
198
      //end of string
199
     tok_tmp = NULL;
200 return s1;
201 }
```

# 4.18 modules/mpx\_supt.c File Reference

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

#### **Functions**

- int sys\_req (int op\_code, int device\_id, char \*buffer\_ptr, int \*count\_ptr)
- void mpx\_init (int cur\_mod)
- void sys\_set\_malloc (u32int(\*func)(u32int))
- void sys\_set\_free (int(\*func)(void \*))

- void \* sys\_alloc\_mem (u32int size)
- int sys\_free\_mem (void \*ptr)
- void idle ()
- 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 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 modules/R1/comHand.h File Reference

## **Functions**

• int comHand ()

#### 4.20.1 Function Documentation

#### 4.20.1.1 comHand()

```
int comHand ( )
```

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

Definition at line 22 of file comHand.c.

```
23
            Help("\0");
            char cmdBuffer[100];
            int bufferSize = 99;
28
            int quit = 0;
29
            int shutdown = 0;
            while(quit != 1) {
    memset(cmdBuffer, '\0', 100);
    sys_req(READ, DEFAULT_DEVICE, cmdBuffer, &bufferSize);
    char* FirstToken = strtok(cmdBuffer, "-");
    char* SecondToken = strtok(NULL, "-");
            while(quit != 1)
33
34
35
                char* ThirdToken = strtok(NULL, "-");
36
                 char* FourthToken = strtok(NULL, "-");
char* FifthToken = strtok(NULL, "-");
38
39
                  if(shutdown == 0) {
40 /******************************
41
                     R1 comHand
42
                           if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, NULL) == 0)
43
                                Help("\setminus 0");
45
                           //R1 Commands
46
        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "version") == 0 &&
strcmp(ThirdToken, NULL) == 0) {
47
48
                                Help("Version");
50
                            else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "getDate") == 0 &&
        strcmp(ThirdToken, NULL) == 0) {
                               Help("GetDate");
51
53
                           else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "setDate") == 0 &&
        strcmp(ThirdToken, NULL) == 0) {
```

```
54
                            Help("SetDate");
55
                         else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "getTime") == 0 &&
56
       strcmp(ThirdToken, NULL) == 0) {
                            Help("GetTime");
57
58
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "setTime") == 0 &&
59
       strcmp(ThirdToken, NULL) == 0)
                            Help("SetTime");
60
61
                        // R2 Commands
62
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "suspend") == 0 &&
63
       strcmp(ThirdToken, NULL) == 0) {
64
                            Help("suspend");
65
66
                         else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "resume") == 0 &&
       strcmp(ThirdToken, NULL) == 0)
67
                            Help("resume");
68
69
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "setPriority") == 0 &&
       strcmp(ThirdToken, NULL) == 0) {
                            Help("setPriority");
70
71
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "showPCB") == 0 &&
72
       strcmp(ThirdToken, NULL) == 0) {
73
                            Help("showPCB");
74
75
                         else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "showAll") == 0 &&
       strcmp(ThirdToken, NULL) == 0)
                            Help("showAll");
76
78
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "showReady") == 0 &&
       strcmp(ThirdToken, NULL) == 0)
79
                            Help("showReady");
80
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "showBlocked") == 0 &&
81
       strcmp(ThirdToken, NULL) == 0) {
                            Help("showBlocked");
82
83
84
                         // Temporary R2 commands
       else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "createPCB") == 0 && strcmp(ThirdToken, NULL) == 0) {
85
                            Help("createPCB");
86
87
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "deletePCB") == 0 &&
88
       strcmp(ThirdToken, NULL) == 0)
89
                            Help("deletePCB");
90
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "block") == 0 &&
91
       strcmp(ThirdToken, NULL) == 0)
92
                            Help("block");
93
94
                         else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "unblock") == 0 &&
       strcmp(ThirdToken, NULL) == 0) {
    Help("unblock");
95
96
                        }
                        else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "shutdown") == 0 &&
97
       strcmp(ThirdToken, NULL) == 0)
98
                            Help("shutdown");
99
100
                         else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "infinite") == 0 &&
       strcmp(ThirdToken,NULL) == 0)
101
                             Help("infinte");
102
103
                         else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "loadr3") == 0 &&
       strcmp(ThirdToken, NULL) == 0) {
    Help("loadr3");
104
105
106
                         else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "alarm") == 0 &&
       strcmp(ThirdToken, NULL) == 0)
107
                             Help("alarm");
108
109
                         else if(strcmp(FirstToken, "help") == 0 && strcmp(SecondToken, "clear") == 0 &&
       strcmp(ThirdToken, NULL) == 0)
                             Help("clear");
110
111
112
113
114
115
116
117
                         else if(strcmp(FirstToken, "version") == 0 && strcmp(SecondToken, NULL) == 0)
118
119
                              Version();
120
                         else if(strcmp(FirstToken, "clear") == 0 && strcmp(SecondToken, NULL) == 0)
121
                              clear();
122
```

```
123
                        else if(strcmp(FirstToken, "getDate") == 0 && strcmp(SecondToken, NULL) == 0)
124
125
126
                        else if (strcmp(FirstToken, "setDate") == 0) {
       if (EdgeCase(SecondToken) == 1 && EdgeCase(ThirdToken) == 1 &&
EdgeCase(FourthToken) == 1 && EdgeCase(FifthToken) == 1) {
127
128
                                SetDate(atoi(SecondToken), atoi(ThirdToken), atoi(FourthToken),
       atoi(FifthToken));
129
130
                            else
                               printf("\x1b[31m""\nERROR: Invalid parameters for setDate \n""\x1b[0m");
131
132
                        else if (strcmp(FirstToken, "getTime") == 0 && strcmp(SecondToken, NULL) == 0) //Return
133
       the current time held by the registers.
134
                            GetTime();
                        else if(strcmp(FirstToken, "setTime") == 0 && strcmp(FifthToken, NULL) == 0){
135
                            if (EdgeCase(SecondToken) == 1 && EdgeCase(ThirdToken) == 1 &&
136
       EdgeCase(FourthToken) == 1)
                                    SetTime(atoi(SecondToken), atoi(ThirdToken), atoi(FourthToken));
137
       //input as Hour-Minute-Seconds
138
139
                                printf("\times1b[31m""\setminusnERROR: Invalid parameters for setTime \setminusn""\setminusx1b[0m");
140
141
142
143
144
145
146
147
148
149
                       R2 comHand
150
           151
                        else if(strcmp(FirstToken, "suspend") == 0 && strcmp(ThirdToken, NULL) == 0 &&
       strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
152
                            Suspend (SecondToken);
153
154
                        else if(strcmp(FirstToken, "resume") == 0 && strcmp(ThirdToken, NULL) == 0 &&
       strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
155
                           Resume (SecondToken);
156
                        else if(strcmp(FirstToken, "setPriority") == 0 && strcmp(FourthToken, NULL) == 0 &&
157
       strcmp(FifthToken,NULL) == 0) {
158
                            if (EdgeCase (ThirdToken) == 1)
                                Set_Priority(SecondToken, atoi(ThirdToken)); //input as
159
       setPriority-Process_Name-Priority
160
161
                            else
                                printf("\x1b[31m""\nERROR: Invalid parameters for setPriority, priority must
162
       be entered as a integer. \n""\x1b[0m");
163
                        else if(strcmp(FirstToken, "showPCB") == 0 && strcmp(ThirdToken, NULL) == 0 &&
164
       165
166
                            printf("\n");
167
                        else if(strcmp(FirstToken, "showAll") == 0 && strcmp(SecondToken, NULL) == 0 &&
168
       strcmp (ThirdToken, NULL) == 0 && strcmp (FourthToken, NULL) == 0 && strcmp (FifthToken, NULL) == 0) {
169
                            Show_All();
170
                            printf("\n");
171
                        else if(strcmp(FirstToken, "showReady") == 0 && strcmp(SecondToken, NULL) == 0 &&
172
       strcmp (ThirdToken, NULL) == 0 && strcmp (FourthToken, NULL) == 0 && strcmp (FifthToken, NULL) == 0) {
173
                            Show_Ready();
174
                            printf("\n");
175
                        else if(strcmp(FirstToken, "showBlocked") == 0 && strcmp(SecondToken, NULL) == 0 &&
176
       strcmp(ThirdToken, NULL) == 0 && strcmp(FourthToken, NULL) == 0 && strcmp(FifthToken, NULL) == 0) {
177
                            Show_Blocked();
178
                            printf("\n");
179
180
181
182
                        /****** R2 Temp Commands *******/
183
                        //Removed from active for R3/R4
184
185
186
                        else if(strcmp(FirstToken, "createPCB") == 0) {
187
                            if( strlen(SecondToken) < 11)</pre>
                                    Create_PCB(SecondToken, atoi(ThirdToken), atoi(FourthToken));
188
       //input as Process_Name-Priority-Class
189
190
                                printf("\x1b[31m""\nERROR: Invalid parameters for createPCB, Process_name
191
       must only contain 10 or fewer characters. \n""\x1b[0m");
192
                        */
193
```

```
else if(strcmp(FirstToken, "deletePCB") == 0 && strcmp(ThirdToken, NULL) == 0 &&
194
       strcmp(FourthToken, NULL) == 0 && strcmp(FifthToken, NULL) == 0) {
195
                           Delete_PCB(SecondToken);
196
197
198
199
200
                       //Removed from active for R3/R4
201
                       else if(strcmp(FirstToken, "block") == 0 && strcmp(ThirdToken, NULL) == 0 &&
202
       strcmp(FourthToken, NULL) == 0 && strcmp(FifthToken, NULL) == 0) {
203
                           Block (SecondToken):
204
205
                       else if(strcmp(FirstToken, "unblock") == 0 && strcmp(ThirdToken, NULL) == 0 &&
       strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
206
                           Unblock (SecondToken);
207
208
                       */
209
210
                       R3 comHand
211
212
                       //Removed for R4
213
                       else if(strcmp(FirstToken, "yield") == 0 && strcmp(SecondToken, NULL) == 0 &&
214
       strcmp (ThirdToken, NULL) == 0 && strcmp (FourthToken, NULL) == 0) {
215
                         yield();
216
                           printf("\n");
217
                       else if(strcmp(FirstToken, "loadr3") == 0 && strcmp(SecondToken, NULL) == 0 &&
218
       strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
219
                           loader();
220
                           printf("\n");
221
222
223
                  224
                       R4 comHand
225
                       else if(strcmp(FirstToken, "alarm") == 0) {
226
227
                           if (EdgeCase(ThirdToken) == 1 && EdgeCase(FourthToken) == 1 &&
       EdgeCase(FifthToken) == 1)
                               if (atoi(ThirdToken) < 23 && atoi(FourthToken) < 59 && atoi(FifthToken) <</pre>
228
      59)
             {
229
                                   loaderalarm (SecondToken, atoi (ThirdToken), atoi (FourthToken),
       atoi(FifthToken));
230
                                   printf("\n"); //input as Message-Hour-Minute-Seconds
231
                               else
232
                               \label{lem:printf("x1b[31m""} nERROR: Invalid parameters for alarm, must be a valid
233
      time \n""\x1b[0m");
234
235
                           else
236
                              printf("\x1b[31m""\nERROR: Invalid parameters for alarm \n""\x1b[0m");
237
238
                       else if(strcmp(FirstToken, "loadr3") == 0 && strcmp(SecondToken, NULL) == 0 &&
239
       strcmp(ThirdToken, NULL) == 0 && strcmp(FourthToken, NULL) == 0 && strcmp(FifthToken, NULL) == 0) {
240
                           loader();
241
                           printf("\n");
242
                       else if(strcmp(FirstToken, "infinite") == 0 && strcmp(SecondToken, NULL) == 0 &&
243
       strcmp (ThirdToken, NULL) == 0 && strcmp (FourthToken, NULL) == 0 && strcmp (FifthToken, NULL) == 0) {
                           loaderinfinite();
244
245
                           printf("\n");
246
247
248
                       shutdown comHand
249
                    ******
                       else if(strcmp(FirstToken, "shutdown") == 0 && strcmp(SecondToken, NULL) == 0){
250
                           printf("\x1b[33m""\nAre you sure you want to shutdown? [yes/no]\n""\x1b[0m");
251
252
                           shutdown = 1;
253
254
                       else
                           printf("\x1b[31m""\nERROR: Not a valid command \n""\x1b[0m");
255
256
257
258
259
                       if(strcmp(FirstToken, "yes") == 0 && shutdown == 1)
260
                           quit = 1;
261
                       else if(strcmp(FirstToken, "no") == 0){
2.62
                           printf("\x1b[33m""\nShutdown Cancelled\x1b[0m \n");
263
264
                           shutdown = 0;
265
266
267
                           2.68
                   sys_reg(IDLE, DEFAULT_DEVICE, NULL, NULL);
269
```

```
270     }
271     getReady() -> head = NULL;
272     sys_req(EXIT, DEFAULT_DEVICE, NULL, NULL);
273     return 0; //shutdown procedure
274 }
```

## 4.21 modules/R1/userFunctions.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <system.h>
#include <core/serial.h>
#include <core/io.h>
#include "../mpx_supt.h"
#include "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()

Description: Changes binary number to decimal numbers.

#### **Parameters**

value Binary number to be changed to decim	ıal
--	-----

#### Definition at line 79 of file userFunctions.c.

```
79 {
80     return (((BCD»4)*10) + (BCD & 0xF));
81 }
```

#### 4.21.1.2 Block()

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

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

## **Parameters**

Process\_Name | Character pointer that matches the name of process.

## Definition at line 899 of file userFunctions.c.

```
900
       PCB* pcb = FindPCB(ProcessName);
      if (pcb == NULL) {
   printf("\xlb[3lm""\nERROR: Not a valid process name \n""\xlb[0m");
901
902
903
      else {
904
        if(pcb->ReadyState == BLOCKED) {
    printf("\xlb[32m""\nThis Process is already BLOCKED \n""\xlb[0m");
905
906
907
908
         else
         RemovePCB(pcb);
909
           pcb->ReadyState = BLOCKED;
910
911
           InsertPCB(pcb);
```

```
913 }
914 }
```

## 4.21.1.3 Create\_PCB()

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

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

#### **Parameters**

Process_Name	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 842 of file userFunctions.c.

```
842
      if (FindPCB(ProcessName) == NULL)
844
       if(Priority >= 0 && Priority < 10){</pre>
845
         if(Class == 0 || Class == 1){
            PCB* pcb = SetupPCB(ProcessName, Class, Priority);
846
847
            InsertPCB(pcb);
848
         } else{
           printf("\x1b[31m""\nERROR: Not a valid Class \n""\x1b[0m");
851
       } else{
         printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
852
853
     } else{
854
       printf("\x1b[31m""\nERROR: This Process Name already exists \n""\x1b[0m");
855
857 }
```

#### 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 86 of file userFunctions.c.

# 4.21.1.5 Delete\_PCB()

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

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

#### **Parameters**

*Process\_Name* Character pointer that matches the name of process.

Definition at line 868 of file userFunctions.c.

```
PCB* pcb = FindPCB(ProcessName);
      if (pcb == NULL)
870
       printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
871
872
873
      else if(strcmp(pcb->Process_Name, "InfProc") == 0) {
874
            if (pcb->SuspendedState == YES) {
875
                RemovePCB (pcb);
876
                FreePCB(pcb);
877
       \label{limit} printf("\x1b[31m""\nERROR:This process cannot be deleted unless it is in the suspended state\n""\x1b[0m");
878
879
880
881
      else if(pcb -> Process_Class == SYSTEM)
882
        printf("\x1b[31m""\nERROR: System Processes cannot be deleted from the system. \n""\x1b[0m");
883
        else {
884
           RemovePCB (pcb);
885
886
            FreePCB (pcb);
887
888 }
```

## 4.21.1.6 EdgeCase()

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

## **Parameters**

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

Definition at line 107 of file userFunctions.c.

```
107 {
108 int valid = 0;
109 if (strcmp(pointer, "00") == 0) {
```

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

#### 4.21.1.7 GetDate()

```
void GetDate ( )
```

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

No parameters.

Definition at line 264 of file userFunctions.c.

```
264
265
      int check = 2;
      outb (0x70, 0x07);
266
        unsigned char day = BCDtoDec(inb(0x71));
267
        outb(0x70,0x08);
268
269
        unsigned char month = BCDtoDec(inb(0x71));
270
        outb(0x70,0x32);
        unsigned char millennium = BCDtoDec(inb(0x71));
char msg[2] = "-";
char msg3[10] = "Date: ";
271
272
273
274
        printf(msg3);
275
        sys_req(WRITE, COM1, itoa(day), &check);
        printf(msg);
277
         sys_req(WRITE, COM1, itoa(month), &check);
278
        printf(msg);
        sys_req(WRITE, COM1, itoa(millennium), &check);
279
280
      outb(0x70,0x09);
if(BCDtoDec(inb(0x71)) == 0){
281
282
        sys_req(WRITE, COM1, "00", &check);
283
284
      else {
            unsigned char year = BCDtoDec(inb(0x71));
sys_req(WRITE, COM1, itoa(year), &check);
285
286
287
      }
288
        printf("\n");
289 }
```

## 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 186 of file userFunctions.c.

```
int check = 2;
187
188
         int hour;
189
         int minute;
190
         int second:
191
        outb (0x70, 0x04);
192
        unsigned char hours = inb(0x71);
193
        outb (0x70, 0x02);
194
        unsigned char minutes = inb(0x71);
        outb(0x70,0x00);
195
        unsigned char seconds = inb(0x71); char msg1[2] = ":"; char msg2[10] = "Time: ";
196
197
198
199
        printf(msg2);
200
         hour = BCDtoDec(hours);
201
         sys_req(WRITE, COM1, itoa(hour), &check);
202
         printf(msg1);
        minute = BCDtoDec(minutes);
203
204
        sys_req(WRITE, COM1, itoa(minute), &check);
205
        printf(msg1);
206
         second = BCDtoDec(seconds);
        sys_req(WRITE, COM1, itoa(second), &check);
2.07
      printf("\n");
208
209 }
```

## 4.21.1.9 Help()

Brief Description: Gives helpful information for one of the functions

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

**Parameters** 

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

Definition at line 318 of file userFunctions.c.

```
318
319
          if (request[0] == ' \setminus 0')
320
               //removed for R3/R4 from active command list
321
               //\n createPCB \n block \n unblock
               printf("\n to chain commands and parameters, please use \n");
322
        printf("\n getDate \n setDate \n getTime \n setTime \n version \n suspend \n resume \n setPriority \n showPCB \n showAll \n showReady \n showBlocked \n deletePCB \n shutdown \n alarm \n
323
        clear \n loadr3 \n infinte \n\n");
324
         else if (strcmp(request, "GetDate") == 0) {
325
326
              printf("\n getDate returns the current date that is loaded onto the operating system.\n");
327
328
         else if (strcmp(request, "SetDate") == 0) {
        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");
330
331
         else if (strcmp(request, "GetTime") == 0) {
332
              printf("\n getTime returns the current time as hours, minutes, seconds that is loaded onto the
        operating system.\n");
333
         else if (strcmp(request, "SetTime") == 0) {
334
        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
335
        number, Example 02 or 00");
336
         else if (strcmp(request, "Version") == 0) {
337
```

```
338
            printf("\n version returns the current operating software version that the system is
       running.\n");
339
        else if (strcmp(request, "infinte") == 0) {
340
           printf("\n infinite Loads the infinite process into the ready queue.\n");
341
342
343
        else if (strcmp(request, "loadr3") == 0)
344
           printf("\n loadr3 Loads in all five of the R3 test processes in a suspended state into the
       queue.\n");
345
        else if (strcmp(request, "alarm") == 0)
346
           printf("\n alarm creates a user specified alarm with a user set message and time
347
       alarm-MSG-hour-minute-second.\n");
348
349
        else if (strcmp(request, "clear") == 0)
           printf("\n clear erases the console of all typed commands and refreshes it with just the command
350
       list.\n");
351
        }
352
353
      else if(strcmp(request, "shutdown") == 0)
       printf("\n shutdown shuts down the system.\n");
354
355
356
357
358
359
                R2 Commands
360
       361
      else if(strcmp(request, "suspend") == 0) {
           printf("\n Suspend takes in the name of a PCB (suspend-NAME) then places it into the suspended
362
      state and reinserts it into the correct queue.\n");
363
      else if(strcmp(request, "resume") == 0) { printf("\n Resume takes in the name of a PCB (resume-NAME) then removes it from the suspended
364
365
       state and adds it to the correct queue. \n");
366
367
      else if(strcmp(request, "setPriority") == 0) {
368
           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^n;
369
      else if(strcmp(request, "showPCB") == 0) { printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
370
371
      user.\n");
372
373
      else if(strcmp(request, "showAll") == 0) {
374
           printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
      queues.\n");
375
      }
376
      else if(strcmp(request, "showReady") == 0) {
           printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
377
       currently are in the ready state.\n");
378
      else if(strcmp(request, "showBlocked") == 0) {
    printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
379
380
      currently are in the blocked state.\n");
381
382
383 /******** R2 Temp Commands
       ***********************
      else if(strcmp(request, "deletePCB") == 0) {
384
            printf("\n DeletePCB takes in the process_name (deletePCB-NAME) then deletes it from the queue
385
       and free's all the memory that was previously allocated to the specified PCB.\n");
386
387
      //removed for R3/R4 from active command list
388
389
      else if(strcmp(request, "createPCB") == 0) {
           printf("\n CreatePCB takes in the process_name, process_class, and
390
       process_priority.(createPCB-NAME-PRIORITY-CLASS) Then assigns this new process into the correct
       queue.\n");
391
392
      else if(strcmp(request,"block") == 0) {
           \texttt{printf("} \  \, \texttt{N} \  \, \texttt{Block takes in the process\_name (block-NAME)} \  \, \texttt{then sets it's state to blocked and}
393
       reinserts it back into the correct queue.\n");
394
395
      else if(strcmp(request, "unblock") == 0) {
           printf("\n Unblock takes in the process_name (unblock-NAME) then sets it's state to ready and
396
      reinserts it back into the correct queue. \n");
397
      }
398
399
      else {
           printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
400
       full list of commands.\n""\x1b[0m");
401
402 }
```

## 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 49 of file userFunctions.c.

```
50
        int i, j, k, count;
51
       i = num;
        j = 0;
        count = 0;
        while(i){ // count number of digits
54
            count++;
i /= 10;
55
56
        }
58
59
        char* arr1;
60
        char arr2[count];
       arr1 = (char*)sys_alloc_mem(count); //memory allocation
61
62
        while(num){ // seperate last digit from number and add ASCII
63
        arr2[++j] = num%10 + '0';
num /= 10;
65
66
67
       for (k = 0; k < j; k++) \{ // \text{ reverse array results } 
 arr1[k] = arr2[j-k];
68
69
70
        arr1[k] = ' \setminus 0';
72
        return(char*)arr1;
73
74 }
```

#### 4.21.1.11 Resume()

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

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

#### **Parameters**

*Process\_Name* Character pointer that matches the name of process.

Definition at line 445 of file userFunctions.c.

```
445 {
446 PCB* pcb = FindPCB(ProcessName);
447 if (pcb == NULL) {
448 printf(RED"\nERROR: Not a valid process name \n"RESET);
449 }
```

```
450
451
            if (pcb->SuspendedState == NO)
               printf(GRN"\nThis Process is already in the NONSUSPENDED state \n"RESET);
452
453
454
            else if(pcb -> Process Class == APPLICATION)
              pcb->SuspendedState = NO;
455
456
457
458
               printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
459
460 }
```

#### 4.21.1.12 Set\_Priority()

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

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

#### **Parameters**

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

## Definition at line 472 of file userFunctions.c.

```
472
                                                            {
473
        PCB* pcb = FindPCB(ProcessName);
474
        if (pcb == NULL)
475
            printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
476
        else if(Priority >= 10){
    printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
477
478
479
480
        else if(pcb -> Process_Class == APPLICATION) {
            RemovePCB(pcb);
481
482
            pcb->Priority = Priority;
483
             InsertPCB(pcb);
484
485
        else
            printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
486
487 }
```

#### 4.21.1.13 SetDate()

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

Dime(day, month, millenial, year).

#### **Parameters**

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

Definition at line 217 of file userFunctions.c.

```
217
218
      outb (0x70, 0x07);
      int tempDay = BCDtoDec(inb(0x71));
219
      outb (0x70, 0x08);
221
      int tempMonth = BCDtoDec(inb(0x71));
222
      outb (0x70, 0x32);
223
      int tempMillennium = BCDtoDec(inb(0x71));
      outb (0x70, 0x09);
224
225
      int tempYear = BCDtoDec(inb(0x71));
226
      cli();
        outb (0x70, 0x07);
228
        outb(0x71,DectoBCD (day));
229
        outb (0x70, 0x08);
230
        outb(0x71,DectoBCD (month));
       outb(0x70,0x32);
outb(0x71,DectoBCD (millennium));
231
232
233
        outb(0x70,0x09);
234
        outb(0x71,DectoBCD (year));
      sti();
outb(0x70,0x07);
235
236
237
      unsigned char newDay = BCDtoDec(inb(0x71));
238
      outb(0x70,0x08);
      unsigned char newMonth = BCDtoDec(inb(0x71));
240
      outb (0x70, 0x32);
241
      unsigned char newMillennium = BCDtoDec(inb(0x71));
242
      outb (0x70, 0x09);
      unsigned char newYear = BCDtoDec(inb(0x71));
243
      if (newDay != day || newMonth != month || newMillennium != millennium || newYear != year) {
244
245
        printf("Your input was invalid\n");
        cli();
247
            outb (0x70, 0x07);
            outb(0x71,DectoBCD (tempDay));
248
            outb(0x70,0x08);
outb(0x71,DectoBCD (tempMonth));
249
250
            outb(0x70,0x32);
251
252
            outb(0x71,DectoBCD (tempMillennium));
253
            outb (0x70, 0x09);
2.54
            outb(0x71,DectoBCD (tempYear));
255
            sti();
256
     }
     else
258
        printf("Date Set\n");
259 }
```

## 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 147 of file userFunctions.c.

```
148
      outb (0x70, 0x04);
149
      unsigned char tempHours = BCDtoDec(inb(0x71));
150
      outb (0x70, 0x02);
151
      unsigned char tempMinutes = BCDtoDec(inb(0x71));
152
      outb (0x70, 0x00);
153
      unsigned char tempSeconds = BCDtoDec(inb(0x71));
154
       cli(); //outb(device + 1, 0x00); //disable interrupts
155
        outb (0x70, 0x04);
        outb(0x71, DectoBCD(hours));// change to bcd
156
157
        outb (0x70, 0x02);
        outb(0x71, DectoBCD(minutes));
158
159
        outb(0x70,0x00);
160
        outb(0x71, DectoBCD(seconds));
      sti(); //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set outb(0x70,0x04);
161
162
      unsigned char newHours = BCDtoDec(inb(0x71));
163
164
      outb (0x70, 0x02);
165
      unsigned char newMinutes = BCDtoDec(inb(0x71));
166
      outb (0x70, 0x00);
167
      unsigned char newSeconds = BCDtoDec(inb(0x71));
      if (newHours != hours || newMinutes != minutes || newSeconds != seconds) {
  printf("Your input was invalid\n");
  cli(); //outb(device + 1, 0x00); //disable interrupts
168
169
170
171
            outb(0x70,0x04);
172
             outb(0x71, DectoBCD(tempHours));// change to bcd
173
             outb (0x70, 0x02);
            outb(0x71, DectoBCD(tempMinutes));
174
             outb(0x70,0x00);
175
             outb(0x71, DectoBCD(tempSeconds));
176
177
             sti(); //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
178
179
180
        printf("Time Set\n");
181 }
```

#### 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 568 of file userFunctions.c.

#### 4.21.1.16 Show\_Blocked()

```
void Show_Blocked ( )
```

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

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

Definition at line 709 of file userFunctions.c.

```
709 {
710 if(getBlocked()->head == NULL) {
```

```
711
               printf("\x1b[32m""\n The Blocked Queue is empty \n""\x1b[0m");
712
713
         else
                int class, check, state, prior, status;
char name[20];
char block[] = "\x1B[34m""Blocked Queue: \n""\x1b[0m";
char cname[] = "Name: ";
char cclass[] = "Class: ";
714
715
716
717
718
                char cstate[] = "State: ";
char cstatus[] = "Status: ";
719
720
                char cstatus[] = "status. ,
char cprior[] = "Priority: ";
char line[] = "\n";
721
722
723
                check = 15;
724
725
                sys_req(WRITE, COM1, block, &check);
726
727
                PCB* pcb = getBlocked()->head;
728
                if(pcb->next == NULL) {
729
730
                  class = pcb->Process_Class;
731
                        strcpy(name,pcb->Process_Name);
                        state = pcb->ReadyState;
status = pcb->SuspendedState;
732
733
                       prior = pcb->Priority;
734
735
736
                       printf(cname);
                       printf(name);
737
738
                       printf(line);
739
740
                        printf(cclass);
741
                        if(pcb->Process_Class == 0) {
742
                         printf("0");
743
744
                        else {
745
                          sys_req(WRITE, COM1, itoa(class), &check);
746
747
                        printf(line);
748
749
                        printf(cstate);
750
                        if(pcb->ReadyState == 0) {
751
                          printf("0");
752
753
                        else
754
                          sys_req(WRITE, COM1, itoa(state), &check);
755
756
                        printf(line);
757
758
                        printf(cstatus);
                        if(pcb->SuspendedState == 0) {
  printf("0");
759
760
761
762
763
                          sys_req(WRITE, COM1, itoa(status), &check);
764
765
                        printf(line);
766
767
                        printf(cprior);
768
                        if (pcb->Priority == 0) {
                          printf("0");
printf("\n\n");
769
770
771
772
                        else {
773
                          sys_req(WRITE, COM1, itoa(prior), &check);
774
                          printf("\n\n");
775
776
777
                else {
                   while (pcb != NULL) {
778
779
                        class = pcb->Process_Class;
780
                            strcpy(name,pcb->Process_Name);
                            state = pcb->ReadyState;
status = pcb->SuspendedState;
781
782
                            prior = pcb->Priority;
783
784
785
                            printf(cname);
786
                            printf(name);
787
                            printf(line);
788
789
                             printf(cclass);
790
                             if (pcb->Process_Class == 0) {
791
                               printf("0");
792
793
                               sys_req(WRITE, COM1, itoa(class), &check);
794
795
796
                             printf(line);
797
```

```
798
                         printf(cstate);
799
                         if(pcb->ReadyState == 0) {
800
                           printf("0");
801
                         else (
802
                           sys_req(WRITE, COM1, itoa(state), &check);
803
804
805
                         printf(line);
806
807
                         printf(cstatus);
                         if(pcb->SuspendedState == 0) {
  printf("0");
808
809
810
811
812
                           sys_req(WRITE, COM1, itoa(status), &check);
813
814
                         printf(line);
815
816
                         printf(cprior);
                         if(pcb->Priority == 0) {
818
                           printf("0");
819
                           printf("\n\n");
820
821
                         else {
822
                           sys_reg(WRITE, COM1, itoa(prior), &check);
                           printf("\n\n");
823
824
825
                    pcb = pcb->next;
826
                }
827
             }
828
        }
829 }
```

#### 4.21.1.17 Show PCB()

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

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

#### **Parameters**

Process\_Name | Character pointer that matches the name of process

## Definition at line 497 of file userFunctions.c.

```
497
          if (FindPCB(ProcessName) == NULL)
498
499
               printf("\x1b[31m""\nERROR: PCB does not exist \n""\x1b[0m");
500
501
               int check = 5;
502
               char name[10];
503
               char cname[] = "Name: ";
504
               char cclass[] = "Class: ";
               char cstatus[] = "Status: ";
char cstatus[] = "Status: ";
506
507
               char cprior[] = "Priority: ";
char line[] = "\n";
PCB* pcb = FindPCB(ProcessName);
508
509
510
511
               strcpy(name,pcb->Process_Name);
               int class = pcb->Process_Class;
int state = pcb->ReadyState;
int status = pcb->SuspendedState;
513
514
               int prior = pcb->Priority;
515
516
517
               if (name == NULL) {
518
                    printf("\x1b[31m""\\nERROR: Not a valid process name \n""\x1b[0m");
519
```

```
else
521
             printf(cname);
522
               printf(ProcessName);
523
               printf(line);
524
               printf(cclass);
               if (pcb->Process_Class == 0) {
525
                  printf("0");
526
527
528
               else {
                  sys_req(WRITE, COM1, itoa(class), &check);
529
530
               printf(line);
531
               printf(cstate);
532
533
               if(pcb->ReadyState == 0) {
534
                  printf("0");
535
536
               else {
                  sys_req(WRITE, COM1, itoa(state), &check);
537
538
539
               printf(line);
540
               printf(cstatus);
541
               if (pcb->SuspendedState == 0) {
                  printf("0");
542
543
544
               else
                  sys_req(WRITE, COM1, itoa(status), &check);
545
546
               printf(line);
547
548
               printf(cprior);
549
               if(pcb->Priority == 0) {
                  printf("0");
550
551
                  printf("\n\n");
552
553
                  554
555
556
557
           }
558
       }
559 }
```

## 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 580 of file userFunctions.c.

```
581
582
583
584
         else
             int class, check, state, prior, status;
585
           char name[10];
char ready[] = "\x1B[34m""\nReady Queue:\n""\x1B[0m";
char cname[] = "Name: ";
char cclass[] = "Class: ";
586
587
588
589
           char cstate[] = "State: ";
590
           char cstatus[] = "Status: ";
char cprior[] = "Priority: ";
char line[] = "\n";
591
592
593
594
           check = 5:
595
596
           sys_req(WRITE, COM1, ready, &check);
598
           PCB* pcb = getReady()->head;
599
600
              if(pcb->next == NULL)
                  class = pcb->Process_Class;
601
                  strcpy(name,pcb->Process_Name);
state = pcb->ReadyState;
602
603
                  status = pcb->SuspendedState;
```

```
605
                prior = pcb->Priority;
606
607
                printf(cname);
608
                printf(name);
609
                printf(line);
610
                printf(cclass);
611
612
                 if(pcb->Process_Class == 0) {
                  printf("0");
613
614
615
                else {
                  sys_req(WRITE, COM1, itoa(class), &check);
616
617
618
                printf(line);
619
620
                printf(cstate);
                if(pcb->ReadyState == 0) {
621
                  printf("0");
622
623
624
                else {
625
                  sys_req(WRITE, COM1, itoa(state), &check);
626
                printf(line);
62.7
62.8
629
                printf(cstatus);
630
                if(pcb->SuspendedState == 0) {
                  printf("0");
631
632
633
                else {
                  sys_req(WRITE, COM1, itoa(status), &check);
634
635
636
                printf(line);
637
638
                printf(cprior);
                if(pcb->Priority == 0) {
  printf("0");
639
640
                  printf("\n\n");
641
642
643
                else {
644
                  sys_req(WRITE, COM1, itoa(prior), &check);
645
                  printf("\n\n");
646
647
648
          else {
            while (pcb != NULL) {
650
                    class = pcb->Process_Class;
651
                     strcpy(name,pcb->Process_Name);
                    state = pcb->ReadyState;
status = pcb->SuspendedState;
652
653
654
                    prior = pcb->Priority;
655
656
                    printf(cname);
657
                     printf(name);
658
                     printf(line);
659
660
                     printf(cclass);
661
                     if (pcb->Process_Class == 0) {
662
                       printf("0");
663
664
                     else
                       sys_req(WRITE, COM1, itoa(class), &check);
665
666
667
                    printf(line);
668
669
                     printf(cstate);
670
                     if(pcb->ReadyState == 0) {
671
                       printf("0");
672
673
                     else {
                      sys_req(WRITE, COM1, itoa(state), &check);
675
676
                     printf(line);
677
                     printf(cstatus);
678
679
                     if (pcb->SuspendedState == 0) {
680
                       printf("0");
681
682
                     else {
683
                       sys_req(WRITE, COM1, itoa(status), &check);
684
685
                     printf(line);
686
687
                     printf(cprior);
688
                     if (pcb->Priority == 0) {
689
                       printf("0");
                       printf("\n\n");
690
691
```

#### 4.21.1.19 Suspend()

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

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

#### **Parameters**

Process\_Name

Character pointer that matches the name of process.

Definition at line 419 of file userFunctions.c.

```
419
        PCB* pcb = FindPCB(ProcessName);
420
        if (pcb == NULL)
421
         printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
422
423
424
       else {
425
            if (pcb->SuspendedState == YES) {
               printf("\x1b[32m""\nThis Process is already SUSPENDED \n""\x1b[0m");
426
427
           else if(pcb -> Process_Class == APPLICATION)
428
429
               pcb->SuspendedState = YES;
430
431
           else
432
               printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
433
       }
434 }
```

## 4.21.1.20 toLowercase()

```
char toLowercase (
```

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 301 of file userFunctions.c.

#### 4.21.1.21 Unblock()

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

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

#### **Parameters**

*Process\_Name* Character pointer that matches the name of process.

Definition at line 925 of file userFunctions.c.

```
926
      PCB* pcb = FindPCB(ProcessName);
927
      if (pcb == NULL)
928
        printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
929
      else {
       if(pcb->ReadyState == READY) {
    printf("\xlb[32m""\nThis Process is already in the READY state \n""\xlb[0m");
931
932
933
       else {
  RemovePCB(pcb);
  pcb->ReadyState = READY;
934
935
936
937
           InsertPCB(pcb);
938
939
      }
940 }
```

#### 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 294 of file userFunctions.c.

```
294 {
295    printf("Version: R4.6 \n");
296 }
```

#### 4.21.2 Variable Documentation

## 4.21.2.1 AlarmList

```
List AlarmList
```

#### Initial value:

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

Definition at line 987 of file userFunctions.c.

# 4.22 modules/R1/userFunctions.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 ( \label{eq:bcd} \text{int } \textit{BCD} \ )
```

Description: Changes binary number to decimal numbers.

#### **Parameters**

value Binary number to be changed to decimal

Definition at line 79 of file userFunctions.c.

```
79 {
80         return (((BCD»4)*10) + (BCD & 0xF));
81 }
```

# 4.22.1.2 Block()

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

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

#### **Parameters**

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

Definition at line 899 of file userFunctions.c.

```
900
      PCB* pcb = FindPCB(ProcessName);
901
      if (pcb == NULL)
       printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
902
903
904
     else {
       if (pcb->ReadyState == BLOCKED) {
905
           printf("\x1b[32m""\nThis Process is already BLOCKED \n""\x1b[0m");
906
907
908
       else
         RemovePCB (pcb);
909
         pcb->ReadyState = BLOCKED;
910
911
          InsertPCB(pcb);
912
     }
914 }
```

#### 4.22.1.3 Create PCB()

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

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

## Parameters

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

## Definition at line 842 of file userFunctions.c.

```
843
       if (FindPCB(ProcessName) == NULL)
         if(Priority >= 0 && Priority < 10){
  if(Class == 0 || Class == 1){
    PCB* pcb = SetupPCB(ProcessName, Class, Priority);</pre>
844
845
846
847
             InsertPCB (pcb);
848
           } else{
             printf("\x1b[31m""\nERROR: Not a valid Class \n""\\x1b[0m");
850
851
        } else{
           printf("\x1b[31m""\\nERROR: Not a valid Priority \n""\\x1b[0m");
852
853
854
      } else{
         printf("\x1b[31m""\nERROR: This Process Name already exists \n""\x1b[0m");
855
856 }
857 }
```

#### 4.22.1.4 DectoBCD()

Description: Changes decimal numbers to binary numbers.

#### **Parameters**

Decimal Decimal number to be changed to binary

Definition at line 86 of file userFunctions.c.

#### 4.22.1.5 Delete PCB()

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

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

#### **Parameters**

*Process\_Name* Character pointer that matches the name of process.

Definition at line 868 of file userFunctions.c.

```
869
      PCB* pcb = FindPCB(ProcessName);
870
     if (pcb == NULL)
      printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
871
872
873
     else if(strcmp(pcb->Process_Name, "InfProc") == 0) {
874
           if (pcb->SuspendedState == YES) {
875
               RemovePCB (pcb);
876
               FreePCB(pcb);
877
878
           else
               printf("\xlb[3lm""\nERROR:This process cannot be deleted unless it is in the suspended
879
      state\n""\x1b[0m");
880
881
     else if(pcb -> Process_Class == SYSTEM) {
        printf("\x1b[31m""\nERROR: System Processes cannot be deleted from the system. \n""\x1b[0m");
882
883
884
       else {
          RemovePCB (pcb);
885
886
           FreePCB(pcb);
887
888 }
```

## 4.22.1.6 EdgeCase()

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

#### **Parameters**

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

Definition at line 107 of file userFunctions.c.

```
107
        int valid = 0;
108
       if (strcmp(pointer, "00") == 0) {
  valid = 1;
109
110
111
          return valid;
112
       else if (strcmp(pointer, "0") == 0) {
113
        valid = 1;
return valid;
114
115
116
117
       else
          int j;
118
         int j;
valid = 0;
for(j = 0; j <= 99; j++) {
   if(strcmp(pointer,itoa(j)) == 0)
   valid = 1;</pre>
119
120
121
122
123
124
         if (valid == 0)
125
            return valid;
         }
126
127
128
       return valid;
```

#### 4.22.1.7 GetDate()

```
void GetDate ( )
```

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

No parameters.

Definition at line 264 of file userFunctions.c.

```
int check = 2;
265
266
      outb (0x70, 0x07);
        unsigned char day = BCDtoDec(inb(0x71));
267
268
        outb(0x70,0x08);
269
        unsigned char month = BCDtoDec(inb(0x71));
270
        outb(0x70,0x32);
        unsigned char millennium = BCDtoDec(inb(0x71)); char msg[2] = "-";
271
272
        char msg3[10] = "Date: ";
273
274
        printf(msg3);
275
         sys_req(WRITE, COM1, itoa(day), &check);
276
        printf(msg);
277
         sys_req(WRITE, COM1, itoa(month), &check);
278
         printf(msg);
         sys_req(WRITE, COM1, itoa(millennium), &check);
279
280
      outb(0x70,0x09);
if(BCDtoDec(inb(0x71)) == 0){
281
        sys_req(WRITE, COM1, "00", &check);
282
283
284
             unsigned char year = BCDtoDec(inb(0x71));
sys_req(WRITE, COM1, itoa(year), &check);
285
286
287
      }
288
        printf("\n");
289 }
```

#### 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 186 of file userFunctions.c.

```
187
        int check = 2;
188
        int hour;
189
        int minute;
190
        int second;
191
        outb (0x70, 0x04);
192
        unsigned char hours = inb(0x71);
193
        outb (0x70, 0x02);
194
        unsigned char minutes = inb(0x71);
195
        outb (0x70, 0x00);
        unsigned char seconds = inb(0x71);
char msg1[2] = ":";
196
197
        char msg2[10] = "Time: ";
198
        printf(msg2);
        hour = BCDtoDec(hours);
200
201
        sys_req(WRITE, COM1, itoa(hour), &check);
202
        printf(msg1);
203
        minute = BCDtoDec(minutes);
204
        sys_req(WRITE, COM1, itoa(minute), &check);
205
        printf(msg1);
        second = BCDtoDec(seconds);
206
     sys_req(WRITE, COM1, itoa(second), &check);
printf("\n");
207
208
209 }
```

## 4.22.1.9 Help()

Brief Description: Gives helpful information for one of the functions

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

#### **Parameters**

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

Definition at line 318 of file userFunctions.c.

```
319
320
              //removed for R3/R4 from active command list
321
              //\n createPCB \n block \n unblock
              printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
322
       printf("\n getDate \n setDate \n getTime \n setTime \n version \n suspend \n resume \n setPriority \n showPCB \n showAll \n showReady \n showBlocked \n deletePCB \n shutdown \n alarm \n
323
        clear \n loadr3 \n infinte \n\n");
324
325
         else if (strcmp(request, "GetDate") == 0) {
              printf("\n getDate returns the current date that is loaded onto the operating system.\n");
326
327
```

```
else if (strcmp(request, "SetDate") == 0) {
           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");
330
            else if (strcmp(request, "GetTime") == 0) {
331
                  printf("\n getTime returns the current time as hours, minutes, seconds that is loaded onto the
332
           operating system.\n");
333
334
             else if (strcmp(request, "SetTime") == 0) {
           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
335
           number, Example 02 or 00");
336
337
             else if (strcmp(request, "Version") == 0) {
                  printf("\n version returns the current operating software version that the system is
338
           running.\n");
339
340
            else if (strcmp(request, "infinte") == 0) {
341
                printf("\n infinite Loads the infinite process into the ready queue.\n");
342
343
             else if (strcmp(request, "loadr3") == 0) {
                  printf("\n loadr3 Loads in all five of the R3 test processes in a suspended state into the
344
           queue.\n");
345
346
             else if (strcmp(request, "alarm") == 0)
347
                  printf("\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n^{-}\n
           alarm-MSG-hour-minute-second.\n");
348
            else if (strcmp(request, "clear") == 0)
349
                  printf("\n clear erases the console of all typed commands and refreshes it with just the command \\
350
           list.\n");
351
352
         else if(strcmp(request, "shutdown") == 0) {
   printf("\n shutdown shuts down the system.\n");
353
354
355
356
357
358
           359
                        R2 Commands
360
         else if(strcmp(request,"suspend") == 0) {
    printf("\n Suspend takes in the name of a PCB (suspend-NAME) then places it into the suspended
362
           state and reinserts it into the correct queue.\n");
363
         else if(strcmp(request, "resume") == 0) {
364
                  printf("\n Resume takes in the name of a PCB (resume-NAME) then removes it from the suspended
365
           state and adds it to the correct queue. \n");
366
367
          else if(strcmp(request, "setPriority") == 0) {
368
                  \texttt{printf("} \  \, \texttt{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");
369
370
         else if(strcmp(request, "showPCB") == 0) {
                  printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
371
           user.\n");
372
373
         else if(strcmp(request, "showAll") == 0) {
                  printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
374
           queues.\n");
375
          else if(strcmp(request, "showReady") == 0) {
376
                    \texttt{printf("} \  \, \texttt{NowReady takes in no parameters but returns all PCB's and there attributes that } \\
377
           currently are in the ready state.\n");
378
379
         else if(strcmp(request, "showBlocked") == 0) {
                  printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
380
           currently are in the blocked state.\n");
381
382
383 /******* R2 Temp Commands
           else if(strcmp(request, "deletePCB") == 0) {
                   printf("\n DeletePCB takes in the process_name (deletePCB-NAME) then deletes it from the queue
385
           and free's all the memory that was previously allocated to the specified PCB.\n");
386
         //removed for R3/R4 from active command list
387
388
         else if(strcmp(request, "createPCB") == 0) {
389
                  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
391
         else if(strcmp(request, "block") == 0) {
392
```

```
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");

394  }

395  else if(strcmp(request, "unblock") == 0) {
        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");

397  }

398  */

399  else {

400     printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
        full list of commands.\n""\x1b[0m");

401  }

402 }
```

#### 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 49 of file userFunctions.c.

```
int i, j, k, count;
50
51
       i = num;
       j = 0;
       count = 0;
       while(i){ // count number of digits
55
           count++;
           i /= 10;
56
59
       char* arr1;
60
       char arr2[count];
61
       arr1 = (char*)sys_alloc_mem(count); //memory allocation
62
       while(num){ // seperate last digit from number and add ASCII
63
        arr2[++j] = num%10 + '0';
           num /= 10;
66
67
       for(k = 0; k < j; k++){ // reverse array results
    arr1[k] = arr2[j-k];</pre>
68
69
70
71
       arr1[k] = ' \setminus 0';
72
73
       return(char*)arr1;
74 }
```

## 4.22.1.11 Resume()

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

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

#### **Parameters**

Definition at line 445 of file userFunctions.c.

```
446
        PCB* pcb = FindPCB(ProcessName);
447
        if (pcb == NULL)
         printf(RED"\next{nERROR}: Not a valid process name \next{n"RESET});
448
449
450
        else {
451
            if(pcb->SuspendedState == NO)
452
                printf(GRN"\nThis Process is already in the NONSUSPENDED state \n"RESET);
453
            else if(pcb -> Process_Class == APPLICATION)
454
455
               pcb->SuspendedState = NO;
456
457
            else
458
                printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
459
        }
460 }
```

#### 4.22.1.12 Set Priority()

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

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

#### **Parameters**

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

Definition at line 472 of file userFunctions.c.

```
472
473
       PCB* pcb = FindPCB(ProcessName);
       if (pcb == NULL)
474
           printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
475
476
477
       else if(Priority >= 10){
           printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m");
478
479
480
       else if(pcb -> Process_Class == APPLICATION) {
           RemovePCB(pcb);
481
482
           pcb->Priority = Priority;
483
            InsertPCB(pcb);
484
485
       else
           printf("\x1b[31m""\nERROR: Cannot Alter System Process \n""\x1b[0m");
486
487 }
```

## 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 217 of file userFunctions.c.

```
217
      outb(0x70,0x07);
218
219
      int tempDay = BCDtoDec(inb(0x71));
220
      outb (0x70, 0x08);
      int tempMonth = BCDtoDec(inb(0x71));
221
222
      outb (0x70, 0x32);
223
      int tempMillennium = BCDtoDec(inb(0x71));
     outb (0x70, 0x09);
225
      int tempYear = BCDtoDec(inb(0x71));
226
      cli();
227
       outb (0x70, 0x07);
       outb(0x71, DectoBCD (day));
228
229
       outb(0x70,0x08);
230
       outb(0x71,DectoBCD (month));
231
       outb(0x70,0x32);
232
       outb(0x71,DectoBCD (millennium));
233
       outb(0x70,0x09);
234
       outb(0x71,DectoBCD (year));
235
       sti();
     outb(0x70,0x07);
236
237
      unsigned char newDay = BCDtoDec(inb(0x71));
238
      outb (0x70, 0x08);
239
      unsigned char newMonth = BCDtoDec(inb(0x71));
240
      outb (0x70, 0x32);
241
     unsigned char newMillennium = BCDtoDec(inb(0x71));
      outb (0x70, 0x09);
242
243
      unsigned char newYear = BCDtoDec(inb(0x71));
      if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
   printf("Your input was invalid\n");
244
245
246
        cli();
            outb(0x70,0x07);
2.47
            outb(0x71,DectoBCD (tempDay));
248
249
            outb(0x70,0x08);
250
           outb(0x71,DectoBCD (tempMonth));
251
           outb(0x70,0x32);
           outb(0x71,DectoBCD (tempMillennium));
252
           outb(0x70,0x09);
253
            outb(0x71,DectoBCD (tempYear));
254
255
            sti();
256
     else
257
258
       printf("Date Set\n");
259 }
```

#### 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	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 147 of file userFunctions.c.

```
147
148
      outb (0x70, 0x04);
      unsigned char tempHours = BCDtoDec(inb(0x71));
149
150
      outb(0x70,0x02);
151
      unsigned char tempMinutes = BCDtoDec(inb(0x71));
152
      outb (0x70, 0x00);
      unsigned char tempSeconds = BCDtoDec(inb(0x71));
cli(); //outb(device + 1, 0x00); //disable interrupts
153
154
155
        outb (0x70, 0x04);
156
        outb(0x71, DectoBCD(hours));// change to bcd
157
        outb (0x70, 0x02);
        outb(0x71, DectoBCD(minutes));
158
159
        outb (0x70, 0x00);
       outb(0x71, DectoBCD(seconds));
sti(); //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
160
161
      outb(0x70,0x04);
162
163
      unsigned char newHours = BCDtoDec(inb(0x71));
164
      outb (0x70, 0x02);
      unsigned char newMinutes = BCDtoDec(inb(0x71));
165
166
      outb (0x70, 0x00);
167
      unsigned char newSeconds = BCDtoDec(inb(0x71));
      if (newHours != hours || newMinutes != minutes || newSeconds != seconds) {
168
169
        printf("Your input was invalid\n");
170
        cli(); //outb(device + 1, 0x00); //disable interrupts
171
            outb(0x70,0x04);
            outb(0x71, DectoBCD(tempHours));// change to bcd
outb(0x70,0x02);
172
173
174
            outb(0x71, DectoBCD(tempMinutes));
175
            outb(0x70,0x00);
176
             outb(0x71, DectoBCD(tempSeconds));
177
            sti(); //outb(device + 4, 0x0B); //enable interrupts, rts/dsr set
178
      }
179
      else
180
        printf("Time Set\n");
181 }
```

#### 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 568 of file userFunctions.c.

```
568 {
569 Show_Ready();
570 Show_Blocked();
571 }
```

#### 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 709 of file userFunctions.c.

```
709
710
         if (getBlocked()->head == NULL)
711
              printf("\x1b[32m""\n The Blocked Queue is empty \n""\x1b[0m");
712
713
        else
               int class, check, state, prior, status;
char name[20];
char block[] = "\x1B[34m""Blocked Queue: \n""\x1b[0m";
char cname[] = "Name: ";
714
715
716
717
               char cclass[] = "Class: ";
               char cstate[] = "State: ";
719
               char cstatus[] = "Status: ";
char oprior[] = "Priority: ";
char line[] = "\n";
720
721
722
723
               check = 15;
724
725
               sys_req(WRITE, COM1, block, &check);
726
727
               PCB* pcb = getBlocked()->head;
728
729
               if(pcb->next == NULL) {
                 class = pcb->Process_Class;
731
                    strcpy(name,pcb->Process_Name);
732
                      state = pcb->ReadyState;
                      status = pcb->SuspendedState;
733
                     prior = pcb->Priority;
734
735
736
                     printf(cname);
737
                      printf(name);
738
                      printf(line);
739
                      printf(cclass);
740
741
                      if(pcb->Process_Class == 0) {
742
                        printf("0");
743
744
                      else
745
                        sys_req(WRITE, COM1, itoa(class), &check);
746
                      printf(line);
747
748
749
                      printf(cstate);
750
                      if(pcb->ReadyState == 0) {
751
                        printf("0");
752
753
                      else {
754
                        sys_req(WRITE, COM1, itoa(state), &check);
755
756
                      printf(line);
757
758
                      printf(cstatus);
                      if (pcb->SuspendedState == 0) {
759
760
                        printf("0");
761
762
                      else {
                        sys_req(WRITE, COM1, itoa(status), &check);
763
764
                      printf(line);
765
766
767
                      printf(cprior);
768
                      if (pcb->Priority == 0)
769
                        printf("0");
770
                        printf("\n\n");
771
772
                      else {
                        sys_req(WRITE, COM1, itoa(prior), &check);
774
                        printf("\n\n");
775
776
777
               else {
778
                 while (pcb != NULL) {
                     class = pcb->Process_Class;
780
                          strcpy(name,pcb->Process_Name);
```

```
781
                       state = pcb->ReadyState;
782
                       status = pcb->SuspendedState;
783
                       prior = pcb->Priority;
784
785
                       printf(cname);
786
                       printf(name);
787
                       printf(line);
788
789
                       printf(cclass);
                        if (pcb->Process_Class == 0) {
790
                         printf("0");
791
792
793
                       else {
794
                        sys_req(WRITE, COM1, itoa(class), &check);
795
796
                       printf(line);
797
798
                       printf(cstate);
                       if (pcb->ReadyState == 0) {
800
                        printf("0");
801
                       else {
802
803
                         sys_req(WRITE, COM1, itoa(state), &check);
804
805
                       printf(line);
806
807
                       printf(cstatus);
808
                       if(pcb->SuspendedState == 0) {
809
                         printf("0");
810
811
                       else {
812
                         sys_req(WRITE, COM1, itoa(status), &check);
813
814
                       printf(line);
815
                       printf(cprior);
816
                       if(pcb->Priority == 0) {
817
818
                         printf("0");
819
                         printf("\n\n");
820
821
                       else {
                         822
823
824
                   pcb = pcb->next;
826
               }
827
828
       }
829 }
```

## 4.22.1.17 Show\_PCB()

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

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

## **Parameters**

Process\_Name | Character pointer that matches the name of process

## Definition at line 497 of file userFunctions.c.

```
503
              char name[10];
             char chame[] = "Name: ";
char cclass[] = "Class: ";
char cstate[] = "State: ";
char cstatus[] = "Status: ";
504
505
506
507
             char cstatus[] = Status. ,
char cprior[] = "Priority: ";
char line[] = "\n";
508
509
510
              PCB* pcb = FindPCB(ProcessName);
511
             strcpy(name,pcb->Process_Name);
             int class = pcb->Process_Class;
int state = pcb->ReadyState;
int status = pcb->SuspendedState;
512
513
514
              int prior = pcb->Priority;
515
516
517
              if (name == NULL) {
                  printf("\x1b[31m""\\nERROR: Not a valid process name \n""\\x1b[0m");
518
519
520
              else
                  printf(cname);
521
                   printf(ProcessName);
523
                   printf(line);
524
                   printf(cclass);
                   if(pcb->Process_Class == 0) {
525
                        printf("0");
526
527
528
                   else {
529
                        sys_req(WRITE, COM1, itoa(class), &check);
530
531
                   printf(line);
                   printf(cstate);
if(pcb->ReadyState == 0) {
532
533
534
                        printf("0");
535
536
537
                        sys_req(WRITE, COM1, itoa(state), &check);
538
539
                   printf(line);
540
                   printf(cstatus);
541
                   if(pcb->SuspendedState == 0)
542
                       printf("0");
543
544
                   else
                       sys_req(WRITE, COM1, itoa(status), &check);
545
546
547
                   printf(line);
548
                   printf(cprior);
549
                   if(pcb->Priority == 0) {
550
                        printf("0");
                        printf("\n\n");
551
552
553
                   else
554
                        sys_req(WRITE, COM1, itoa(prior), &check);
555
                        printf("\n\n");
556
              }
557
         }
558
```

## 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 580 of file userFunctions.c.

```
588
           char cname[] = "Name: ";
           char cname[] = "Name: ";
char cclass[] = "Class: ";
char cstate[] = "State: ";
char cstatus[] = "Status: ";
char cprior[] = "Priority: ";
char line[] = "\n";
589
590
591
592
593
594
           check = 5;
595
596
           sys_req(WRITE, COM1, ready, &check);
597
598
           PCB* pcb = getReady()->head;
599
600
              if(pcb->next == NULL)
601
                  class = pcb->Process_Class;
602
                  strcpy(name,pcb->Process_Name);
                  state = pcb->ReadyState;
status = pcb->SuspendedState;
603
604
                  prior = pcb->Priority;
605
606
607
                  printf(cname);
608
                  printf(name);
609
                  printf(line);
610
                  printf(cclass);
611
                  if (pcb->Process_Class == 0) {
612
                    printf("0");
613
614
615
                    sys_req(WRITE, COM1, itoa(class), &check);
616
617
618
                  printf(line);
619
620
                  printf(cstate);
621
                  if (pcb->ReadyState == 0) {
622
                    printf("0");
623
624
                  else {
                    sys_req(WRITE, COM1, itoa(state), &check);
625
626
627
                  printf(line);
628
62.9
                  printf(cstatus);
                  if(pcb->SuspendedState == 0) {
630
631
                    printf("0");
632
633
                  else
634
                    sys_req(WRITE, COM1, itoa(status), &check);
635
                  printf(line);
636
637
638
                  printf(cprior);
639
                  if(pcb->Priority == 0) {
640
                    printf("0");
                    printf("\n\n");
641
642
643
                  else {
644
                    sys_req(WRITE, COM1, itoa(prior), &check);
645
                    printf("\n\n");
646
647
648
           else (
             while (pcb != NULL) {
649
650
                      class = pcb->Process_Class;
651
                       strcpy(name,pcb->Process_Name);
652
                       state = pcb->ReadyState;
                      status = pcb->SuspendedState;
prior = pcb->Priority;
653
654
655
656
                      printf(cname);
657
                       printf(name);
658
                       printf(line);
659
660
                       printf(cclass);
                       if (pcb->Process_Class == 0) {
661
                         printf("0");
662
663
664
665
                        sys_req(WRITE, COM1, itoa(class), &check);
666
                       printf(line);
667
668
669
                       printf(cstate);
670
                       if(pcb->ReadyState == 0) {
671
                         printf("0");
672
673
                       else {
674
                         sys_reg(WRITE, COM1, itoa(state), &check);
```

```
676
                    printf(line);
677
678
                    printf(cstatus);
                    if(pcb->SuspendedState == 0) {
679
                     printf("0");
680
681
682
683
                      sys_req(WRITE, COM1, itoa(status), &check);
684
                    printf(line);
685
686
687
                    printf(cprior);
688
                    if (pcb->Priority == 0) {
689
                      printf("0");
                      printf("\n\n");
690
691
                    else {
692
                     sys_req(WRITE, COM1, itoa(prior), &check);
693
694
                      printf("\n\n");
695
696
                    pcb = pcb->next;
               }
697
698
           }
699
       }
700 }
```

#### 4.22.1.19 Suspend()

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

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

## Parameters

*Process Name* Character pointer that matches the name of process.

Definition at line 419 of file userFunctions.c.

```
419
420
       PCB* pcb = FindPCB(ProcessName);
421
       if (pcb == NULL)
        printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
422
423
424
          if(pcb->SuspendedState == YES) {
    printf("\xlb[32m""\nThis Process is already SUSPENDED \n""\xlb[0m");
425
42.6
427
428
          else if(pcb -> Process_Class == APPLICATION)
             pcb->SuspendedState = YES;
430
431
          else
              432
433
       }
434 }
```

#### 4.22.1.20 toLowercase()

```
char toLowercase (
```

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 301 of file userFunctions.c.

#### 4.22.1.21 Unblock()

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

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

#### **Parameters**

Process\_Name Character pointer that matches the name of process.

Definition at line 925 of file userFunctions.c.

```
925
926
      PCB* pcb = FindPCB(ProcessName);
927
      if (pcb == NULL)
       printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
928
929
930
       if(pcb->ReadyState == READY) {
    printf("\x1b[32m""\nThis Process is already in the READY state \n""\x1b[0m");
931
932
933
       else
934
935
        RemovePCB(pcb);
          pcb->ReadyState = READY;
936
937
          InsertPCB(pcb);
938
        }
939 }
940 }
```

## 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 294 of file userFunctions.c.

```
295 printf("Version: R4.6 \n");
296 }
```

# 4.23 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 modules/sys\_proc\_loader.h File Reference

## **Functions**

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

# Index

Alarm, 5	idt_entry_struct, 8
AlarmList	idt_struct, 9
userFunctions.c, 51	inb
atoi	io.h, 13
string.c, 25	include/core/asm.h, 13
string.h, 16	include/core/interrupts.h, 13
	include/core/io.h, 13
BCDtoDec	include/core/serial.h, 14
userFunctions.c, 36	include/core/tables.h, 14
userFunctions.h, 53	include/mem/heap.h, 15
Block	include/mem/paging.h, 15
userFunctions.c, 36	include/string.h, 16
userFunctions.h, 53	include/system.h, 20
	index_entry, 9
comHand	index_table, 9
comHand.h, 31	io.h
comHand.h	inb, 13
comHand, 31	
context, 5	isspace
Create PCB	string.c, 26
userFunctions.c, 37	string.h, 16
userFunctions.h, 54	itoa
	userFunctions.c, 41
date time, 6	userFunctions.h, 59
DectoBCD	1 1/ /
userFunctions.c, 37	kernel/core/interrupts.c, 21
userFunctions.h, 54	kernel/core/kmain.c, 22
Delete PCB	kernel/core/serial.c, 22
userFunctions.c, 38	kernel/core/system.c, 23
userFunctions.h, 55	kernel/core/tables.c, 23
useri unctions.ii, 55	kernel/mem/heap.c, 24
EdgeCase	kernel/mem/paging.c, 24
userFunctions.c, 38	
userFunctions.h, 55	lib/string.c, 25
useri unctions.ii, 55	List, 10
footer, 6	
,	memset
gdt descriptor struct, 7	string.c, 26
gdt_entry_struct, 7	string.h, 17
GetDate	modules/mpx_supt.c, 29
userFunctions.c, 39	modules/mpx_supt.h, 30
userFunctions.h, 56	modules/R1/comHand.h, 31
GetTime	modules/R1/userFunctions.c, 35
userFunctions.c, 39	modules/R1/userFunctions.h, 52
userFunctions.h, 56	modules/sys_proc_loader.c, 69
useri unctions.ii, so	modules/sys_proc_loader.h, 69
header, 7	
heap, 8	page_dir, 10
Help	page_entry, 10
userFunctions.c, 40	page_table, 11
userFunctions.h, 57	param, 11
ason unouonsin, or	

72 INDEX

DCD 10	atutal
PCB, 12	strtok string.c, 28
Queue, 12	string.h, 19
	Suspend
Resume	userFunctions.c, 50
userFunctions.c, 42	userFunctions.h, 67
userFunctions.h, 59	
Set_Priority	toLowercase
userFunctions.c, 43	userFunctions.c, 50 userFunctions.h, 67
userFunctions.h, 60	userrunctions.n, 67
SetDate	Unblock
userFunctions.c, 43	userFunctions.c, 51
userFunctions.h, 60	userFunctions.h, 68
SetTime	userFunctions.c
userFunctions.c, 44	AlarmList, 51
userFunctions.h, 61	BCDtoDec, 36
Show_All	Block, 36
userFunctions.c, 45 userFunctions.h, 62	Create_PCB, 37
Show Blocked	DectoBCD, 37
userFunctions.c, 45	Delete_PCB, 38
userFunctions.h, 62	EdgeCase, 38 GetDate, 39
Show PCB	GetTime, 39
userFunctions.c, 47	Help, 40
userFunctions.h, 64	itoa, 41
Show_Ready	Resume, 42
userFunctions.c, 48	Set_Priority, 43
userFunctions.h, 65	SetDate, 43
strcat	SetTime, 44
string.c, 27	Show_All, 45
string.h, 17	Show_Blocked, 45
strcmp	Show_PCB, 47
string.c, 27	Show_Ready, 48
string.h, 18	Suspend, 50
strcpy string.c, 28	toLowercase, 50
string.h, 18	Unblock, 51
string.c	Version, 51 userFunctions.h
atoi, 25	BCDtoDec, 53
isspace, 26	Block, 53
memset, 26	Create PCB, 54
strcat, 27	DectoBCD, 54
strcmp, 27	Delete_PCB, 55
strcpy, 28	EdgeCase, 55
strlen, 28	GetDate, 56
strtok, 28	GetTime, 56
string.h	Help, 57
atoi, 16	itoa, 59
isspace, 16 memset, 17	Resume, 59
streat, 17	Set_Priority, 60
strcmp, 18	SetDate, 60 SetTime, 61
stropy, 18	Show All, 62
strlen, 19	Show Blocked, 62
strtok, 19	Show PCB, 64
strlen	Show Ready, 65
string.c, 28	Suspend, 67
string.h, 19	toLowercase, 67
	,

INDEX 73

Unblock, 68 Version, 68

Version

userFunctions.c, 51 userFunctions.h, 68