## Runtime Terror

Generated by Doxygen 1.9.1

1	CS_450_RunTime_Terror	1
2	Class Index	3
	2.1 Class List	3
3	File Index	5
	3.1 File List	5
4	Class Documentation	7
	4.1 date_time Struct Reference	7
	4.1.1 Detailed Description	7
	4.2 footer Struct Reference	7
	4.2.1 Detailed Description	7
	4.3 gdt_descriptor_struct Struct Reference	8
	4.3.1 Detailed Description	8
	4.4 gdt_entry_struct Struct Reference	8
	4.4.1 Detailed Description	8
	4.5 header Struct Reference	8
	4.5.1 Detailed Description	9
	4.6 heap Struct Reference	9
	4.6.1 Detailed Description	9
	4.7 idt_entry_struct Struct Reference	9
	4.7.1 Detailed Description	9
	4.8 idt_struct Struct Reference	10
	4.8.1 Detailed Description	10
	4.9 index_entry Struct Reference	10
	4.9.1 Detailed Description	10
	4.10 index_table Struct Reference	10
	4.10.1 Detailed Description	10
	4.11 page_dir Struct Reference	11
	4.11.1 Detailed Description	11
	4.12 page_entry Struct Reference	11
	4.12.1 Detailed Description	11
	4.13 page_table Struct Reference	11
	4.13.1 Detailed Description	12
	4.14 param Struct Reference	12
	4.14.1 Detailed Description	12
	4.15 PCB Struct Reference	12
	4.15.1 Detailed Description	12
	4.16 Queue Struct Reference	13
	4.16.1 Detailed Description	13
	4.17 struct Struct Reference	13
	4.17.1 Detailed Description	13
		-

5 File Documentation	15
5.1 mpx_core/include/core/asm.h File Reference	15
5.2 mpx_core/include/core/interrupts.h File Reference	15
5.3 mpx_core/include/core/io.h File Reference	15
5.3.1 Macro Definition Documentation	15
5.3.1.1 inb	15
5.4 mpx_core/include/core/serial.h File Reference	16
5.5 mpx_core/include/core/tables.h File Reference	16
5.6 mpx_core/include/mem/heap.h File Reference	17
5.7 mpx_core/include/mem/paging.h File Reference	17
5.8 mpx_core/include/string.h File Reference	18
5.8.1 Function Documentation	18
5.8.1.1 atoi()	18
5.8.1.2 isspace()	19
5.8.1.3 memset()	19
5.8.1.4 strcat()	20
5.8.1.5 strcmp()	20
5.8.1.6 strcpy()	21
5.8.1.7 strlen()	21
5.8.1.8 strtok()	21
5.9 mpx_core/include/system.h File Reference	22
5.10 mpx_core/kernel/core/interrupts.c File Reference	23
5.11 mpx_core/kernel/core/kmain.c File Reference	24
5.12 mpx_core/kernel/core/serial.c File Reference	25
5.13 mpx_core/kernel/core/system.c File Reference	25
5.14 mpx_core/kernel/core/tables.c File Reference	25
5.15 mpx_core/kernel/mem/heap.c File Reference	26
5.16 mpx_core/kernel/mem/paging.c File Reference	26
5.17 mpx_core/lib/string.c File Reference	27
5.17.1 Function Documentation	28
5.17.1.1 atoi()	28
5.17.1.2 isspace()	28
5.17.1.3 memset()	29
5.17.1.4 strcat()	29
5.17.1.5 strcmp()	30
5.17.1.6 strcpy()	30
5.17.1.7 strlen()	30
5.17.1.8 strtok()	31
5.18 mpx_core/modules/mpx_supt.c File Reference	32
5.19 mpx_core/modules/mpx_supt.h File Reference	32
5.20 mpx_core/modules/R1/comHand.c File Reference	33
5.21 mpx_core/modules/R1/comHand.h File Reference	34

59

5.22 mpx_core/modules/R1/userFunctions.c File Reference	34
5.22.1 Function Documentation	35
5.22.1.1 BCDtoDec()	35
5.22.1.2 DectoBCD()	35
5.22.1.3 EdgeCase()	36
5.22.1.4 GetDate()	36
5.22.1.5 GetTime()	37
5.22.1.6 Help()	37
5.22.1.7 itoa()	39
5.22.1.8 Resume()	39
5.22.1.9 Set_Priority()	40
5.22.1.10 SetDate()	40
5.22.1.11 SetTime()	41
5.22.1.12 Show_All()	42
5.22.1.13 Show_Blocked()	43
5.22.1.14 Show_PCB()	44
5.22.1.15 Show_Ready()	44
5.22.1.16 Suspend()	45
5.22.1.17 toLowercase()	45
5 00 4 40 Newsiam ()	45
5.22.1.18 Version()	45
5.22.1.18 Version()	
	46
5.23 mpx_core/modules/R1/userFunctions.h File Reference	46 47
5.23 mpx_core/modules/R1/userFunctions.h File Reference	46 47 47
5.23 mpx_core/modules/R1/userFunctions.h File Reference	46 47 47
5.23 mpx_core/modules/R1/userFunctions.h File Reference	46 47 47 48
5.23 mpx_core/modules/R1/userFunctions.h File Reference  5.23.1 Function Documentation	46 47 47 47 48
5.23 mpx_core/modules/R1/userFunctions.h File Reference	46 47 47 47 48 48
5.23 mpx_core/modules/R1/userFunctions.h File Reference	46 47 47 48 48 49
5.23 mpx_core/modules/R1/userFunctions.h File Reference         5.23.1 Function Documentation         5.23.1.1 BCDtoDec()         5.23.1.2 DectoBCD()         5.23.1.3 EdgeCase()         5.23.1.4 GetDate()         5.23.1.5 GetTime()         5.23.1.6 Help()	46 47 47 48 48 49 51
5.23 mpx_core/modules/R1/userFunctions.h File Reference         5.23.1 Function Documentation         5.23.1.1 BCDtoDec()         5.23.1.2 DectoBCD()         5.23.1.3 EdgeCase()         5.23.1.4 GetDate()         5.23.1.5 GetTime()         5.23.1.6 Help()         5.23.1.7 itoa()	46 47 47 48 49 49 51
5.23 mpx_core/modules/R1/userFunctions.h File Reference         5.23.1 Function Documentation         5.23.1.1 BCDtoDec()         5.23.1.2 DectoBCD()         5.23.1.3 EdgeCase()         5.23.1.4 GetDate()         5.23.1.5 GetTime()         5.23.1.6 Help()         5.23.1.7 itoa()         5.23.1.8 Resume()	46 47 47 48 48 49 51 51 52
5.23 mpx_core/modules/R1/userFunctions.h File Reference         5.23.1 Function Documentation         5.23.1.1 BCDtoDec()         5.23.1.2 DectoBCD()         5.23.1.3 EdgeCase()         5.23.1.4 GetDate()         5.23.1.5 GetTime()         5.23.1.6 Help()         5.23.1.7 itoa()         5.23.1.8 Resume()         5.23.1.9 Set_Priority()	46 47 47 48 49 51 52 52
5.23 mpx_core/modules/R1/userFunctions.h File Reference  5.23.1 Function Documentation  5.23.1.1 BCDtoDec()  5.23.1.2 DectoBCD()  5.23.1.3 EdgeCase()  5.23.1.4 GetDate()  5.23.1.5 GetTime()  5.23.1.6 Help()  5.23.1.7 itoa()  5.23.1.8 Resume()  5.23.1.9 Set_Priority()  5.23.1.10 SetDate()	46 47 47 48 49 51 52 53
5.23 mpx_core/modules/R1/userFunctions.h File Reference 5.23.1 Function Documentation 5.23.1.1 BCDtoDec() 5.23.1.2 DectoBCD() 5.23.1.3 EdgeCase() 5.23.1.4 GetDate() 5.23.1.5 GetTime() 5.23.1.6 Help() 5.23.1.7 itoa() 5.23.1.8 Resume() 5.23.1.9 Set_Priority() 5.23.1.10 SetDate() 5.23.1.11 SetTime()	46 47 48 48 49 51 52 52 53 54
5.23 mpx_core/modules/R1/userFunctions.h File Reference  5.23.1 Function Documentation  5.23.1.1 BCDtoDec()  5.23.1.2 DectoBCD()  5.23.1.3 EdgeCase()  5.23.1.4 GetDate()  5.23.1.5 GetTime()  5.23.1.6 Help()  5.23.1.7 itoa()  5.23.1.8 Resume()  5.23.1.9 Set_Priority()  5.23.1.10 SetDate()  5.23.1.11 SetTime()  5.23.1.11 SetTime()  5.23.1.12 Show_All()	46 47 48 49 51 52 52 53 54 55
5.23 mpx_core/modules/R1/userFunctions.h File Reference  5.23.1 Function Documentation  5.23.1.1 BCDtoDec()  5.23.1.2 DectoBCD()  5.23.1.3 EdgeCase()  5.23.1.4 GetDate()  5.23.1.5 GetTime()  5.23.1.6 Help()  5.23.1.7 itoa()  5.23.1.8 Resume()  5.23.1.9 Set_Priority()  5.23.1.10 SetDate()  5.23.1.11 SetTime()  5.23.1.12 Show_All()  5.23.1.13 Show_Blocked()	46 47 47 48 49 51 52 52 53 54 55 56
5.23 mpx_core/modules/R1/userFunctions.h File Reference  5.23.1 Function Documentation  5.23.1.1 BCDtoDec()  5.23.1.2 DectoBCD()  5.23.1.3 EdgeCase()  5.23.1.4 GetDate()  5.23.1.5 GetTime()  5.23.1.6 Help()  5.23.1.7 itoa()  5.23.1.8 Resume()  5.23.1.9 Set_Priority()  5.23.1.10 SetDate()  5.23.1.11 SetTime()  5.23.1.12 Show_All()  5.23.1.13 Show_Blocked()  5.23.1.14 Show_Ready()	46 47 48 49 51 52 52 53 54 56 56
5.23 mpx_core/modules/R1/userFunctions.h File Reference  5.23.1 Function Documentation  5.23.1.1 BCDtoDec()  5.23.1.2 DectoBCD()  5.23.1.3 EdgeCase()  5.23.1.4 GetDate()  5.23.1.5 GetTime()  5.23.1.6 Help()  5.23.1.7 itoa()  5.23.1.8 Resume()  5.23.1.9 Set_Priority()  5.23.1.10 SetDate()  5.23.1.11 SetTime()  5.23.1.13 Show_All()  5.23.1.13 Show_Blocked()  5.23.1.14 Show_Ready()  5.23.1.15 Suspend()	46 47 47 48 49 51 52 52 53 54 55 56 57

Index

# **Chapter 1**

# CS\_450\_RunTime\_Terror

R1 Implementation

Bonus assignments included are -Variable Text Color -itoa funciton

# Chapter 2

# **Class Index**

### 2.1 Class List

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

date_time	7
	7
2	8
9 · <u>_</u> · · / · · ·	8
	8
	9
······································	9
	10
	10
	10
page	11
pago_o)	11
	11
	12
• • • • • • • • • • • • • • • • • • • •	12
Queue	13
struct	13

4 Class Index

# **Chapter 3**

# File Index

### 3.1 File List

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

mpx_core/include/string.h	. 18
mpx_core/include/system.h	. 22
mpx_core/include/core/asm.h	. 15
mpx_core/include/core/interrupts.h	. 15
mpx_core/include/core/io.h	. 15
mpx_core/include/core/serial.h	. 16
mpx_core/include/core/tables.h	. 16
mpx_core/include/mem/heap.h	. 17
mpx_core/include/mem/paging.h	. 17
mpx_core/kernel/core/interrupts.c	. 23
mpx_core/kernel/core/kmain.c	. 24
mpx_core/kernel/core/serial.c	. 25
mpx_core/kernel/core/system.c	. 25
mpx_core/kernel/core/tables.c	. 25
mpx_core/kernel/mem/heap.c	. 26
mpx_core/kernel/mem/paging.c	. 26
mpx_core/lib/string.c	. 27
mpx_core/modules/mpx_supt.c	. 32
mpx_core/modules/mpx_supt.h	. 32
mpx_core/modules/R1/comHand.c	. 33
mpx_core/modules/R1/comHand.h	. 34
mpx_core/modules/R1/userFunctions.c	. 34
mpx_core/modules/R1/userFunctions.h	. 46
mpx_core/modules/R2/ <b>PCB.c</b>	. ??
mpx_core/modules/R2/ <b>PCB.h</b>	. ??

6 File Index

# **Chapter 4**

# **Class Documentation**

### 4.1 date\_time Struct Reference

### **Public Attributes**

- int sec
- int min
- int hour
- int day\_w
- int day\_m
- int day\_y
- int mon
- int year

### 4.1.1 Detailed Description

Definition at line 32 of file system.h.

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

• mpx\_core/include/system.h

### 4.2 footer Struct Reference

### **Public Attributes**

· header head

### 4.2.1 Detailed Description

Definition at line 18 of file heap.h.

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

• mpx\_core/include/mem/heap.h

8 Class Documentation

### 4.3 gdt\_descriptor\_struct Struct Reference

### **Public Attributes**

- u16int limit
- u32int base

### 4.3.1 Detailed Description

Definition at line 25 of file tables.h.

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

• mpx\_core/include/core/tables.h

### 4.4 gdt\_entry\_struct Struct Reference

### **Public Attributes**

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

### 4.4.1 Detailed Description

Definition at line 32 of file tables.h.

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

• mpx\_core/include/core/tables.h

### 4.5 header Struct Reference

### **Public Attributes**

- int size
- int index\_id

### 4.5.1 Detailed Description

Definition at line 13 of file heap.h.

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

• mpx\_core/include/mem/heap.h

### 4.6 heap Struct Reference

### **Public Attributes**

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

### 4.6.1 Detailed Description

Definition at line 35 of file heap.h.

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

• mpx\_core/include/mem/heap.h

### 4.7 idt\_entry\_struct Struct Reference

### **Public Attributes**

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

### 4.7.1 Detailed Description

Definition at line 8 of file tables.h.

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

• mpx core/include/core/tables.h

10 Class Documentation

### 4.8 idt struct Struct Reference

#### **Public Attributes**

- u16int limit
- u32int base

### 4.8.1 Detailed Description

Definition at line 18 of file tables.h.

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

• mpx\_core/include/core/tables.h

### 4.9 index\_entry Struct Reference

#### **Public Attributes**

- int size
- · int empty
- u32int block

### 4.9.1 Detailed Description

Definition at line 22 of file heap.h.

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

• mpx\_core/include/mem/heap.h

### 4.10 index\_table Struct Reference

### **Public Attributes**

- index entry table [TABLE SIZE]
- int id

### 4.10.1 Detailed Description

Definition at line 29 of file heap.h.

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

• mpx\_core/include/mem/heap.h

### 4.11 page\_dir Struct Reference

### **Public Attributes**

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

### 4.11.1 Detailed Description

Definition at line 36 of file paging.h.

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

• mpx\_core/include/mem/paging.h

### 4.12 page\_entry Struct Reference

### **Public Attributes**

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

### 4.12.1 Detailed Description

Definition at line 14 of file paging.h.

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

• mpx\_core/include/mem/paging.h

### 4.13 page\_table Struct Reference

### **Public Attributes**

• page\_entry pages [1024]

12 Class Documentation

### 4.13.1 Detailed Description

Definition at line 28 of file paging.h.

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

• mpx core/include/mem/paging.h

### 4.14 param Struct Reference

#### **Public Attributes**

- int op code
- · int device\_id
- char \* buffer ptr
- int \* count\_ptr

### 4.14.1 Detailed Description

Definition at line 33 of file mpx\_supt.h.

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

• mpx\_core/modules/mpx\_supt.h

### 4.15 PCB Struct Reference

### **Public Attributes**

- unsigned char stack [1KMEM]
- unsigned char \* stackTop
- struct PCB \* prev
- struct PCB \* next
- char Process\_Name [10]
- · int Process\_Class
- int **Priority**
- · int ReadyState
- · int SuspendedState

### 4.15.1 Detailed Description

Definition at line 27 of file PCB.c.

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

• mpx\_core/modules/R2/PCB.c

### 4.16 Queue Struct Reference

### **Public Attributes**

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

### 4.16.1 Detailed Description

Definition at line 7 of file PCB.c.

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

• mpx\_core/modules/R2/PCB.c

### 4.17 struct Struct Reference

### **Public Attributes**

· int count

### 4.17.1 Detailed Description

Definition at line 9 of file PCB.h.

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

• mpx\_core/modules/R2/PCB.h

14 Class Documentation

# **Chapter 5**

# **File Documentation**

### 5.1 mpx\_core/include/core/asm.h File Reference

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

## 5.2 mpx\_core/include/core/interrupts.h File Reference

### **Functions**

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

## 5.3 mpx\_core/include/core/io.h File Reference

### **Macros**

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

#### 5.3.1 Macro Definition Documentation

### 5.3.1.1 inb

Definition at line 17 of file io.h.

### 5.4 mpx\_core/include/core/serial.h File Reference

#### **Macros**

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

### **Functions**

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

### 5.5 mpx\_core/include/core/tables.h File Reference

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

#### Classes

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

### **Functions**

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

#### **Variables**

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

### 5.6 mpx core/include/mem/heap.h File Reference

#### **Classes**

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

### **Macros**

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

### **Functions**

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

### 5.7 mpx\_core/include/mem/paging.h File Reference

#include <system.h>

### **Classes**

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

### **Macros**

• #define PAGE\_SIZE 0x1000

### **Functions**

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

### 5.8 mpx\_core/include/string.h File Reference

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

### **Functions**

• int isspace (const char \*c)

Description: Determine if a character is whitespace.

void \* memset (void \*s, int c, size\_t n)

Description: Set a region of memory.

• char \* strcpy (char \*s1, const char \*s2)

Description: Copy one string to another.

• char \* strcat (char \*s1, const char \*s2)

Description: Concatenate the contents of one string onto another.

• int strlen (const char \*s)

Description: Returns the length of a string.

• int strcmp (const char \*s1, const char \*s2)

Description: String comparison.

• char \* strtok (char \*s1, const char \*s2)

Description: Split string into tokens.

• int atoi (const char \*s)

Description: Convert an ASCII string to an integer.

### 5.8.1 Function Documentation

#### 5.8.1.1 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.

```
51 {
52    int res=0;
53    int charVal=0;
54    char sign = ' ';
55    char c = *s;
56
57
58    while(isspace(&c)){ ++s; c = *s;} // advance past whitespace
59
60
61    if (*s == '-' || *s == '+') sign = *(s++); // save the sign
62
```

### 5.8.1.2 isspace()

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

Description: Determine if a character is whitespace.

#### **Parameters**

```
c character to check
```

Definition at line 121 of file string.c.

### 5.8.1.3 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.

```
140 {
141 unsigned char *p = (unsigned char *) s;
```

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

### 5.8.1.4 strcat()

```
char* strcat (  {\rm char} \ * \ s1, \\ {\rm const} \ {\rm 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 }
```

### 5.8.1.5 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.

```
83
       // Remarks:
84
      // Remarks: 

// 1) If we made it to the end of both strings (i. e. our pointer points to a 

// ^{\prime} (0' character), the function will return 0 

// 2) If we didn't make it to the end of both strings, the function will
85
86
            return the difference of the characters at the first index of indifference.
89
       while ( (*s1) && (*s1==*s2) ){
90
        ++s1;
91
92
         ++s2;
93
       return ( *(unsigned char *)s1 - *(unsigned char *)s2 );
95 }
```

### 5.8.1.6 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 }
```

### 5.8.1.7 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 }
```

### 5.8.1.8 strtok()

```
char* strtok ( \label{eq:char} \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.

```
static char *tok_tmp = NULL;
155
156
       const char *p = s2;
157
158
       //new string
159
       if (s1!=NULL) {
160
        tok\_tmp = s1;
161
       //old string cont'd
162
       else {
  if (tok_tmp==NULL) {
163
164
           return NULL;
165
166
167
         s1 = tok\_tmp;
168
169
       //skip leading s2 characters while ( *p && *s1 ) {
   if (*s1==*p) {
170
171
         ++s1;
p = s2;
continue;
173
174
175
176
177
         ++p;
178
179
180
       //{\rm no} more to parse
      return (tok_tmp = NULL);
}
       if (!*s1){
181
182
183
184
185
       //skip non-s2 characters
       tok_tmp = s1;
while (*tok_tmp) {
 p = s2;
186
187
188
        while (*p) {
   if (*tok_tmp==*p++) {
   *tok_tmp++ = '\0';
189
190
192
         return s1;
193
           }
194
195
         ++tok_tmp;
196
197
198
       //end of string
199
      tok_tmp = NULL;
200
       return s1;
201 }
```

## 5.9 mpx\_core/include/system.h File Reference

#### **Classes**

struct date\_time

### **Macros**

- #define NULL 0
- #define no\_warn(p) if (p) while (1) break
- #define asm \_\_asm\_\_
- #define volatile \_\_volatile\_\_
- #define sti() asm volatile ("sti"::)

- #define cli() asm volatile ("cli"::)
- #define nop() asm volatile ("nop"::)
- #define hlt() asm volatile ("hlt"::)
- #define iret() asm volatile ("iret"::)
- #define GDT CS ID 0x01
- #define GDT\_DS\_ID 0x02

### **Typedefs**

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

#### **Functions**

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

### 5.10 mpx\_core/kernel/core/interrupts.c File Reference

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

#### **Macros**

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

### **Functions**

- void divide\_error ()
- void debug ()
- void **nmi** ()
- · void breakpoint ()
- · void overflow ()
- void bounds ()
- void invalid\_op ()
- void device\_not\_available ()
- void double\_fault ()
- void coprocessor\_segment ()
- void invalid\_tss ()

```
    void segment_not_present ()
```

- void stack\_segment ()
- void general\_protection ()
- void page\_fault()
- void reserved ()
- void coprocessor ()
- · void rtc isr ()
- void isr0 ()
- void do\_isr ()
- void init irq (void)
- void init pic (void)
- void do divide error ()
- void do\_debug ()
- void do\_nmi ()
- void do\_breakpoint ()
- void do\_overflow ()
- void do bounds ()
- void do\_invalid\_op ()
- void do\_device\_not\_available ()
- void do\_double\_fault ()
- void do\_coprocessor\_segment ()
- void do\_invalid\_tss ()
- void do\_segment\_not\_present ()
- void do stack segment ()
- void do\_general\_protection ()
- void do\_page\_fault ()
- void do\_reserved ()
- void do\_coprocessor ()

### **Variables**

• idt\_entry idt\_entries [256]

## 5.11 mpx\_core/kernel/core/kmain.c File Reference

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

### **Functions**

· void kmain (void)

### 5.12 mpx\_core/kernel/core/serial.c File Reference

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

#### **Macros**

• #define NO\_ERROR 0

#### **Functions**

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

#### **Variables**

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

### 5.13 mpx\_core/kernel/core/system.c File Reference

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

### **Functions**

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

### 5.14 mpx\_core/kernel/core/tables.c File Reference

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

### **Functions**

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

### **Variables**

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

### 5.15 mpx core/kernel/mem/heap.c File Reference

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

#### **Functions**

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

### **Variables**

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

## 5.16 mpx\_core/kernel/mem/paging.c File Reference

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

#### **Functions**

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

### **Variables**

```
u32int mem_size = 0x4000000
```

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

### 5.17 mpx\_core/lib/string.c File Reference

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

### **Functions**

• int strlen (const char \*s)

Description: Returns the length of a string.

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

Description: Copy one string to another.

• int atoi (const char \*s)

Description: Convert an ASCII string to an integer.

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

Description: String comparison.

• char \* strcat (char \*s1, const char \*s2)

Description: Concatenate the contents of one string onto another.

• int isspace (const char \*c)

Description: Determine if a character is whitespace.

void \* memset (void \*s, int c, size\_t n)

Description: Set a region of memory.

• char \* strtok (char \*s1, const char \*s2)

Description: Split string into tokens.

### 5.17.1 Function Documentation

### 5.17.1.1 atoi()

```
int atoi ( const char * s )
```

Description: Convert an ASCII string to an integer.

**Parameters** 

s String

Definition at line 50 of file string.c.

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

### 5.17.1.2 isspace()

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

Description: Determine if a character is whitespace.

**Parameters** 

c character to check

Definition at line 121 of file string.c.

```
*C == '\t' ||
*C == '\v'){
128
```

#### 5.17.1.3 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.

```
unsigned char *p = (unsigned char *) s;

142 while(n--) {

143 *p++ = (unsigned char) c;
```

### 5.17.1.4 strcat()

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

Description: Concatenate the contents of one string onto another.

### **Parameters**

s1	destination
s2	source

### Definition at line 108 of file string.c.

```
109 {
110 char *rc = s1;

111 if (*s1) while(*++s1);

112 while((*s1++ = *s2++));

113 return rc;
114 }
```

#### 5.17.1.5 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.

### 5.17.1.6 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 }
```

#### 5.17.1.7 strlen()

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

Description: Returns the length of a string.

### **Parameters**

```
s input string
```

Definition at line 26 of file string.c.

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

### 5.17.1.8 strtok()

Description: Split string into tokens.

#### **Parameters**

s1	String
s2	delimiter

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

```
154 {
      static char *tok_tmp = NULL;
155
      const char *p = s2;
156
157
158
       //new string
159
      if (s1!=NULL) {
160
        tok\_tmp = s1;
161
162
      //old string cont'd
163
      else {
164
       if (tok_tmp==NULL) {
        return NULL;
165
166
      s1 = tok_tmp;
167
168
169
       //skip leading s2 characters
171
       while ( *p && *s1 ) {
       wnile ( *p && *s
if (*s1==*p) {
    ++s1;
    p = s2;
    continue;
}
172
173
174
175
176
177
         ++p;
178
179
      //no more to parse
if (!*s1) {
180
181
182
        return (tok_tmp = NULL);
183
184
       //skip non-s2 characters
185
      tok_tmp = s1;
while (*tok_tmp) {
   p = s2;
186
187
188
         while (*p) {
189
190
          if (*tok_tmp==*p++) {
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 }
```

# 5.18 mpx\_core/modules/mpx\_supt.c File Reference

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

### **Functions**

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

### **Variables**

- param params
- int current\_module = -1
- u32int(\* student\_malloc )(u32int)
- int(\* student\_free )(void \*)

# 5.19 mpx\_core/modules/mpx\_supt.h File Reference

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

### **Classes**

• struct param

### **Macros**

- #define EXIT 0
- #define IDLE 1
- #define READ 2
- #define WRITE 3
- #define INVALID OPERATION 4
- #define TRUE 1
- #define FALSE 0
- #define MODULE R1 0
- #define MODULE\_R2 1
- #define MODULE R3 2
- #define MODULE\_R4 4
- #define MODULE\_R5 8
- #define MODULE\_F 9
- #define IO MODULE 10
- #define MEM\_MODULE 11
- #define INVALID\_BUFFER 1000
- #define INVALID\_COUNT 2000
- #define DEFAULT\_DEVICE 111
- #define COM\_PORT 222

### **Functions**

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

# 5.20 mpx\_core/modules/R1/comHand.c File Reference

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

#### **Functions**

• int comHand ()

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

### 5.21 mpx core/modules/R1/comHand.h File Reference

### **Functions**

· int comHand ()

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

### 5.22 mpx\_core/modules/R1/userFunctions.c File Reference

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

#### **Functions**

• char \* itoa (int num)

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

int BCDtoDec (int BCD)

Description: Changes binary number to decimal numbers.

• int DectoBCD (int Decimal)

Description: Changes decimal numbers to binary numbers.

- void printf (char msg[])
- int EdgeCase (char \*pointer)

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

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

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

void GetTime ()

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

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

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

· void GetDate ()

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

void Version ()

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

• char toLowercase (char c)

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

void Help (char \*request)

Brief Description: Gives helpful information for one of the functions.

void Suspend (Char \*Process Name)

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

• void Resume (Char \*Process\_Name)

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

void Set\_Priority (Char \*Process\_Name, int Priority)

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

void Show\_PCB (char \*Process\_Name)

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

• void Show\_All ()

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

void Show Ready ()

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

void Show\_Blocked ()

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

#### 5.22.1 Function Documentation

### 5.22.1.1 BCDtoDec()

```
int BCDtoDec ( int \ \textit{BCD} \ )
```

Description: Changes binary number to decimal numbers.

**Parameters** 

value Binary number to be changed to decimal

Definition at line 64 of file userFunctions.c.

```
64 {
65          return (((BCD»4)*10) + (BCD & 0xF));
66     }
```

### 5.22.1.2 DectoBCD()

Description: Changes decimal numbers to binary numbers.

**Parameters** 

Decimal | Decimal number to be changed to binary

Definition at line 71 of file userFunctions.c.

### 5.22.1.3 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 83 of file userFunctions.c.

```
83
84
         int valid = 0;
         if (strcmp(pointer, "00") == 0) {
  valid = 1;
85
86
87
           return valid;
88
        int i, j;
for (i = 0; i < strlen(pointer); i++) {
   valid = 0;
   for(j = 0; j <= 99; j++) {</pre>
89
90
91
92
93
                    if(strcmp(pointer,itoa(j)) == 0)
                         valid = 1;
94
95
               if (valid == 0) {
96
97
                return valid;
98
99
100
          return valid;
101
```

### 5.22.1.4 GetDate()

```
void GetDate ( )
```

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

No parameters.

Definition at line 225 of file userFunctions.c.

```
225
          int check = 2;
226
          outb(0x70,0x07);
227
           unsigned char day = BCDtoDec(inb(0x71));
228
229
            outb(0x70,0x08);
230
            unsigned char month = BCDtoDec(inb(0x71));
231
            outb(0x70,0x32);
           unsigned char millennium = BCDtoDec(inb(0x71));
char msg[2] = "-";
232
233
234
           char msg3[10] = "Date: ";
235
           printf(msg3);
236
            sys_req(WRITE, COM1, itoa(day), &check);
237
           printf(msg);
238
            sys_req(WRITE, COM1, itoa(month), &check);
            printf(msg);
239
240
            sys_req(WRITE, COM1, itoa(millennium), &check);
241
        outb (0x70, 0x09);
```

```
242     if (BCDtoDec(inb(0x71)) == 0) {
243          sys_req(WRITE, COM1, "00", &check);
244     }
245     else {
246          unsigned char year = BCDtoDec(inb(0x71));
247          sys_req(WRITE, COM1, itoa(year), &check);
248     }
249     printf("\n");
```

### 5.22.1.5 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 147 of file userFunctions.c.

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

### 5.22.1.6 Help()

```
void Help ( {\tt char} \ * \ request \ )
```

Brief Description: Gives helpful information for one of the functions.

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

### **Parameters**

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

```
Definition at line 274 of file userFunctions.c.
```

```
275
        int check = 1;
            if (request[0] == ' \setminus 0') {
276
                    printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
2.77
                printf("\n getDate \n setDate \n getTime \n setTime \n version \n shutdown \n\n");
278
279
            else if (strcmp(request, "GetDate") == 0) {
280
                   printf("\n getDate returns the current date that is loaded onto the operating
281
       system.\n");
282
            else if (strcmp(request, "SetDate") == 0) {
283
       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
284
       number, Example 02 or 00");
285
            286
287
       onto the operating system. \n");
288
            else if (strcmp(request, "SetTime") == 0) {
289
       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
290
       number, Example 02 or 00");
291
            else if (strcmp(request, "Version") == 0) {
293
                    printf("\n version returns the current operating software version that the system is
       running.\n");
294
        else if(strcmp(request, "shutdown") == 0) {
295
296
         printf("\n shutdown shuts down the system.\n");
297
298
300
        else if(strcmp(request, "suspend") == 0) {
        printf("\n Suspend takes in the name of a PCB then places it into the suspended state and reinserts
301
       it into the correct queue.\n");
302
303
             if (strcmp(FirstToken, "resume") == 0) {
        printf("\n Resume takes in the name of a PCB then removes it from the suspended state and adds it to
304
       the correct queue.\n");
305
        else if(strcmp(FirstToken, "setPriority") == 0) {
306
307
        printf("\n SetPriority takes in the name of a PCB and the priority it needs to be set to then
       reinstates the specified PCB into a new location by priority.\n");
308
309
        else if(strcmp(FirstToken, "showPCB") == 0) {
        310
       user.\n");
311
312
        else if(strcmp(FirstToken, "showAll") == 0) {
        printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
313
       queues.\n");
314
        else if(strcmp(FirstToken, "showReady") == 0) {
315
       printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that currently are in the ready state.\n");
316
317
        else if(strcmp(FirstToken, "showBlocked") == 0) {
318
       printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that currently are in the blocked state.\n");
319
320
321
322 /******* R2 Temp Commands
       **********************
323
        else if(strcmp(FirstToken, "createPCB") == 0) {
324
        printf("\n CreatePCB takes in the process_name, process_class, and process_priority. Then assigns
       this new process into the correct queue.\n");
325
326
             if (strcmp(FirstToken, "deletePCB") == 0) {
        \texttt{printf("} \ \texttt{N} \ \texttt{DeletePCB} \ \texttt{takes} \ \texttt{in the process\_name then deletes} \ \texttt{it from the queue and free's all the}
327
       memory that was previously allocated to the specified PCB.\n");
328
        else if(strcmp(FirstToken, "block") == 0) {
329
        printf("\n Block takes in the process_name then sets it's state to blocked and reinserts it back
330
       into the correct queue.\n");
331
332
        else if(strcmp(FirstToken, "unblock") == 0) {
333
        printf("\n Unblock takes in the process_name then sets it's state to ready and reinserts it back
       into the correct queue.\n^n);
334
335
       printf("x1b[31m"nThe requested command does not exist please refer to the Help function for a full list of commands.n""x1b[0m");
336
337
338 }
```

#### 5.22.1.7 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 33 of file userFunctions.c.

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

### 5.22.1.8 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 371 of file userFunctions.c.

```
371
372
373
374
// Name Error check
375 // Error check (Valid Name)
376 //if (Process_Name != valid name) {
377 // printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
378 //}
379
380 }
```

### 5.22.1.9 Set\_Priority()

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

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

#### **Parameters**

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

Definition at line 388 of file userFunctions.c.

```
388
                                                                  {
389
390
391
      // Name Error check
392
       // Error check (Valid Name)
      //if (Process_Name != valid name) {    // printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
393
394
395
396
      // Priority error check
397
        if (Priority == i) {
398
399
400
401
        else{
402
          printf("\x1b[31m""\nERROR: Not a valid Priority \n""\x1b[0m")
403
404
405
406 }
```

#### 5.22.1.10 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 178 of file userFunctions.c.

```
178
179
        outb (0x70, 0x07);
        int tempDay = BCDtoDec(inb(0x71));
180
181
        outb (0x70, 0x08);
182
        int tempMonth = BCDtoDec(inb(0x71));
183
        outb (0x70, 0x32);
184
        int tempMillennium = BCDtoDec(inb(0x71));
        outb (0x70, 0x09);
185
186
        int tempYear = BCDtoDec(inb(0x71));
187
        cli();
            outb(0x70,0x07);
189
            outb(0x71,DectoBCD (day));
190
            outb (0x70, 0x08);
            outb(0x71,DectoBCD (month));
outb(0x70,0x32);
191
192
193
            outb(0x71,DectoBCD (millennium));
194
            outb(0x70,0x09);
195
            outb(0x71,DectoBCD (year));
196
197
            sti();
        outb (0x70, 0x07);
        unsigned char newDay = BCDtoDec(inb(0x71));
198
199
        outb(0x70,0x08);
200
        unsigned char newMonth = BCDtoDec(inb(0x71));
201
        outb(0x70,0x32);
202
        unsigned char newMillennium = BCDtoDec(inb(0x71));
203
        outb (0x70, 0x09);
        unsigned char newYear = BCDtoDec(inb(0x71));
204
        if (newDay != day || newMonth != month || newMillennium != millennium || newYear != year) {
205
          printf("Your input was invalid\n");
206
207
208
            outb (0x70, 0x07);
            outb(0x71,DectoBCD (tempDay));
209
            outb(0x70,0x08);
outb(0x71,DectoBCD (tempMonth));
210
211
            outb(0x70,0x32);
212
213
            outb(0x71,DectoBCD (tempMillennium));
214
            outb(0x70,0x09);
215
            outb(0x71,DectoBCD (tempYear));
216
            sti();
217
218
        printf("Date Set\n");
}
        else
219
220
```

### 5.22.1.11 SetTime()

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

#### **Parameters**

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

Definition at line 108 of file userFunctions.c.

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

### 5.22.1.12 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 439 of file userFunctions.c.

```
440
       int check = 20:
441
       int i;
442
       int j;
       for(i = 0; i < sizeof(ready queue);i++)</pre>
443
         char rProcess_Name = ready queue [i] Process_Name;
         int rClass = ready queue [i] class; char rState = ready queue[i] state;
445
446
         char rStatus = ready queue[i] status;
447
         int rPriority = ready queue[i] priority;
sys_req(WRITE, COM1, rProcess_Name, &check);
448
450
          sys_req(WRITE, COM1, itoa(rClass), &check);
451
          sys_req(WRITE, COM1, rState, &check);
452
          sys_req(WRITE, COM1, rStatus, &check);
453
         sys_req(WRITE, COM1, itoa(rPriority), &check);
454
455
       for(j = 0; j < sizeof(blocked queue); j++){</pre>
456
         char bProcess_Name = blocked queue [j] Process_Name;
         int bClass = blocked queue [j] class;
char bState = blocked queue[j] state;
457
458
         char bStatus = blocked queue[j] status;
int bPriority = blocked queue[j] priority;
sys_req(WRITE, COM1, bProcess_Name, &check);
459
460
461
462
          sys_req(WRITE, COM1, itoa(bClass), &check);
463
          sys_req(WRITE, COM1, bState, &check);
464
          sys_req(WRITE, COM1, bStatus, &check);
465
          sys_req(WRITE, COM1, itoa(bPriority), &check);
466
467 }
```

### 5.22.1.13 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. 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.

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.
--------------	---

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.

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**

Proces	ss_Name	Character pointer that matches the name of process.
--------	---------	---

Definition at line 492 of file userFunctions.c.

### 5.22.1.14 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 413 of file userFunctions.c.

```
414
       int class, check, state, prior;
415
       char[] name;
416
       check = 10;
       PCB* pcb = FindPCB(Process_Name);
417
418
       class = pcb->Process_Class;
419
       name = pcb->Process_Name;
       state = pcb->ReadyState;
status = pcb->SuspendedState;
420
421
422
       prior = pcb->Priority;
423
424
       if(name == NULL) {
425
        printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
       } else{
426
         sys_req(WRITE, COM1, name, &check);
427
         sys_req(WRITE, COMI, itoa(class), &check);
sys_req(WRITE, COMI, itoa(state), &check);
sys_req(WRITE, COMI, itoa(status), &check);
428
429
430
431
         sys_req(WRITE, COM1, itoa(priot), &check);
432 }
433 }
```

### 5.22.1.15 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 472 of file userFunctions.c.

```
473
       int check = 20;
474
       int i;
       for(i = 0; i < sizeof(ready queue);i++)</pre>
475
         char Process_Name = ready queue [i] Process_Name;
char Class = ready queue [i] class;
476
477
478
         char State = ready queue[i] state;
479
         char Status = ready queue[i] status;
         char Priority = ready queue[i] priority;
sys_req(WRITE, COM1, Process_Name, &check);
480
481
         sys_req(WRITE, COM1, Class, &check);
482
483
         sys_req(WRITE, COM1, State, &check);
484
         sys_req(WRITE, COM1, Status, &check);
485
         sys_req(WRITE, COM1, Priority, &check);
486
487 }
```

### 5.22.1.16 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 356 of file userFunctions.c.

```
356
357
358 // Name Error check
359 // Error check (Valid Name)
360 //if (Process_Name != valid name) {
361 // printf("\xlb[31m""\nERROR: Not a valid process name \n""\xlb[0m");
362 //}
363
364 }
```

### 5.22.1.17 toLowercase()

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

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

(char)

### **Parameters**

c Character that is to be changed to its lowercase equivalent

Definition at line 262 of file userFunctions.c.

#### 5.22.1.18 Version()

```
void Version ( )
```

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

(the iteration that module is currently on).

No parameters.

Definition at line 255 of file userFunctions.c.

```
255 {
256 printf("Version: R2.0 \n");
257 }
```

## 5.23 mpx\_core/modules/R1/userFunctions.h File Reference

### **Macros**

- #define **RED** "\x1B[31m"
- #define GRN "\x1B[32m"
- #define YEL "\x1B[33m"
- #define **BLU** "\x1B[34m"
- #define MAG "\x1B[35m"
- #define CYN "\x1B[36m"
- #define WHT "\x1B[37m"
- #define RESET "\x1B[0m"

#### **Functions**

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

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

void GetTime ()

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

• int DectoBCD (int Decimal)

Description: Changes decimal numbers to binary numbers.

char \* itoa (int num)

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

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

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

• int BCDtoDec (int BCD)

Description: Changes binary number to decimal numbers.

void GetDate ()

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

· void Version ()

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

void Help (char \*request)

Brief Description: Gives helpful information for one of the functions.

- void printf (char msg[])
- int EdgeCase (char \*pointer)

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

• char toLowercase (char c)

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

void Suspend (Char \*Process\_Name)

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

void Resume (Char \*Process\_Name)

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

void Set\_Priority (Char \*Process\_Name, int Priority)

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

- void Show\_PCB (Char \*Process Name)
- void Show All ()

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

void Show Ready ()

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

• void Show Blocked ()

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

- void Create\_PCB (char \*Process\_Name, int Priority, int Class)
- void Delete\_PCB (Char \*Process Name)
- void **Block** (Char \*Process\_Name)
- void Unblock (Char \*Process\_Name)

### 5.23.1 Function Documentation

### 5.23.1.1 BCDtoDec()

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

Description: Changes binary number to decimal numbers.

#### **Parameters**

value Binary number to be changed to decimal

Definition at line 64 of file userFunctions.c.

#### 5.23.1.2 DectoBCD()

Description: Changes decimal numbers to binary numbers.

#### **Parameters**

Decimal Decimal number to be changed to binary

Definition at line 71 of file userFunctions.c.

### 5.23.1.3 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 83 of file userFunctions.c.

```
83
84
          int valid = 0;
         if (strcmp(pointer, "00") == 0) {
  valid = 1;
85
86
87
            return valid;
88
         int i, j;
for (i = 0; i < strlen(pointer); i++) {
    valid = 0;
    for(j = 0; j <= 99; j++) {</pre>
89
90
92
                    if(strcmp(pointer,itoa(j)) == 0)
  valid = 1;
93
94
95
96
               if (valid == 0) {
                 return valid;
99
100
          return valid;
        }
101
```

### 5.23.1.4 GetDate()

```
void GetDate ( )
```

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

No parameters.

Definition at line 225 of file userFunctions.c.

```
233
            char msg[2] = "-";
234
            char msg3[10] = "Date: ";
235
            printf(msg3);
236
            sys_req(WRITE, COM1, itoa(day), &check);
237
            printf(msq);
            sys_reg(WRITE, COM1, itoa(month), &check);
238
           printf(msg);
240
            sys_req(WRITE, COM1, itoa(millennium), &check);
241
        outb (0x70, 0x09);
        if(BCDtoDec(inb(0x71)) == 0){
242
         sys_req(WRITE, COM1, "00", &check);
243
244
245
       else {
246
            unsigned char year = BCDtoDec(inb(0x71));
247
            sys_req(WRITE, COM1, itoa(year), &check);
248
            printf("\n");
249
250
```

### 5.23.1.5 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 147 of file userFunctions.c.

```
148
        int check = 2;
149
        int hour;
150
        int minute;
151
        int second;
            outb(0x70,0x04);
152
153
            unsigned char hours = inb(0x71);
            outb(0x70,0x02);
154
155
            unsigned char minutes = inb(0x71);
156
            outb (0x70, 0x00);
            unsigned char seconds = inb(0x71);
char msg1[2] = ":";
157
158
            char msg2[10] = "Time: ";
159
160
            printf(msg2);
            hour = BCDtoDec(hours);
161
162
            sys_req(WRITE, COM1, itoa(hour), &check);
163
            printf(msg1);
164
            minute = BCDtoDec(minutes);
            sys_req(WRITE, COM1, itoa(minute), &check);
165
166
            printf(msq1);
167
            second = BCDtoDec(seconds);
            sys_req(WRITE, COM1, itoa(second), &check);
169
          printf("\n");
170
```

### 5.23.1.6 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 274 of file userFunctions.c.
274
275
        int check = 1:
276
           if (request[0] == '\0') {
277
                   printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
278
               printf("\n getDate \n setDate \n getTime \n setTime \n version \n shutdown \n\n");
279
           else if (strcmp(request, "GetDate") == 0) {
280
                  printf("\n getDate returns the current date that is loaded onto the operating
281
      system.\n");
282
           else if (strcmp(request, "SetDate") == 0) {
283
      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
284
      number, Example 02 or 00");
285
286
           else if (strcmp(request, "GetTime") == 0) {
287
                   printf("\n getTime returns the current time as hours, minutes, seconds that is loaded
       onto the operating system. \n");
288
      289
290
      number, Example 02 or 00");
291
           else if (strcmp(request, "Version") == 0) { printf("\n version returns the current operating software version that the system is
292
293
       running.\n");
294
295
            if (strcmp(request, "shutdown") == 0) {
296
         printf("\n shutdown shuts down the system.\n");
297
298
else if(strcmp(request, "suspend") == 0) {
300
301
       \texttt{printf("} \backslash \texttt{n} \text{ Suspend takes in the name of a PCB then places it into the suspended state and reinserts}
       it into the correct queue.\n");
302
303
       else if(strcmp(FirstToken, "resume") == 0) {
       304
      the correct queue.\n");
305
306
       else if(strcmp(FirstToken, "setPriority") == 0) {
307
       printf("\n SetPriority takes in the name of a PCB and the priority it needs to be set to then
       reinstates the specified PCB into a new location by priority.\n");
308
309
       else if(strcmp(FirstToken, "showPCB") == 0) {
       printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
310
      user.\n");
311
       else if(strcmp(FirstToken, "showAll") == 0) {
312
313
        \texttt{printf("$\normalfootnote{$^{\circ}$} } \text{ In ShowAll takes no parameters but returns all PCB's that are currently in any of the } \\
      queues.\n");
314
315
       else if(strcmp(FirstToken, "showReady") == 0) {
       printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
316
      currently are in the ready state.\n");
317
       else if(strcmp(FirstToken, "showBlocked") == 0) {
318
       printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
319
      currently are in the blocked state.\n");
320
321
322 /************************ R2 Temp Commands
      *******************
323
       else if(strcmp(FirstToken, "createPCB") == 0) {
       printf("\n CreatePCB takes in the process_name, process_class, and process_priority. Then assigns
324
       this new process into the correct queue.\n");
325
       else if(strcmp(FirstToken, "deletePCB") == 0) {
326
       printf("\n DeletePCB takes in the process_name then deletes it from the queue and free's all the
327
      memory that was previously allocated to the specified PCB.\n");
328
       else if(strcmp(FirstToken, "block") == 0) {
329
       printf("\n Block takes in the process_name then sets it's state to blocked and reinserts it back
330
       into the correct queue.\n");
331
       else if(strcmp(FirstToken, "unblock") == 0) {
332
       printf("\n Unblock takes in the process_name then sets it's state to ready and reinserts it back
333
       into the correct queue.\n");
```

```
334    }
335    else    {
336        printf("\x1b[31m""\nThe requested command does not exist please refer to the Help function for a
        full list of commands.\n""\x1b[0m");
337     }
338 }
```

#### 5.23.1.7 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 33 of file userFunctions.c.

```
34
35
                int i, j, k, count;
36
                i = num;
37
                j = 0;
           count = 0;
while(i){ // count number of digits
38
39
40
                count++;
41
                i /= 10;
43
44
           char* arr1;
45
           char arr2[count];
           arr1 = (char*)sys_alloc_mem(count); //memory allocation
46
           while(num) { // seperate last digit from number and add ASCII
48
                arr2[++j] = num%10 + '0';
num /= 10;
50
51
            }
52
            for(k = 0; k < j; k++){ // reverse array results
53
                arr1[k] = arr2[j-k];
55
56
           arr1[k] = ' \setminus 0';
57
            return(char*)arr1;
58
59
       }
```

#### 5.23.1.8 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 371 of file userFunctions.c.

```
371
372
373
374  // Name Error check
375  // Error check (Valid Name)
376  //if (Process_Name != valid name) {
377  // printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
378  //}
379
380 }
```

### 5.23.1.9 Set\_Priority()

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

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

### **Parameters**

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

Definition at line 388 of file userFunctions.c.

```
388
389
390
391
       // Name Error check
      // Error check (Valid Name)
//if (Process_Name != valid name) {
392
393
          printf("\x1b[31m""\nERROR: Not a valid process name <math>\n""\x1b[0m");
394
395
396
       // Priority error check
397
       for (i = 0; i < 9; i++) {
        if (Priority == i) {
398
399
           break;
400
401
        else{
          printf("\x1b[31m""\nERROR: Not a valid Priority <math>\n""\x1b[0m")
403
      }
404
405
406 }
```

### 5.23.1.10 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 178 of file userFunctions.c.

```
178
        outb (0x70, 0x07);
179
180
        int tempDay = BCDtoDec(inb(0x71));
181
        outb(0x70,0x08);
        int tempMonth = BCDtoDec(inb(0x71));
182
183
        outb (0x70, 0x32);
        int tempMillennium = BCDtoDec(inb(0x71));
184
185
        outb (0x70, 0x09);
186
        int tempYear = BCDtoDec(inb(0x71));
187
        cli();
188
            outb (0x70, 0x07);
            outb(0x71, DectoBCD (day));
outb(0x70, 0x08);
189
190
            outb(0x71,DectoBCD (month));
191
192
            outb(0x70,0x32);
193
            outb(0x71,DectoBCD (millennium));
194
            outb (0x70, 0x09);
195
            outb(0x71, DectoBCD (year));
196
             sti();
197
        outb(0x70,0x07);
198
        unsigned char newDay = BCDtoDec(inb(0x71));
199
        outb (0x70, 0x08);
200
        unsigned char newMonth = BCDtoDec(inb(0x71));
201
        outb (0x70, 0x32);
202
        unsigned char newMillennium = BCDtoDec(inb(0x71));
203
        outb(0x70,0x09);
204
        unsigned char newYear = BCDtoDec(inb(0x71));
        if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
   printf("Your input was invalid\n");
205
206
207
          cli();
            outb(0x70,0x07);
208
            outb(0x71,DectoBCD (tempDay));
209
210
            outb(0x70,0x08);
211
            outb(0x71,DectoBCD (tempMonth));
212
            outb (0x70, 0x32);
            outb(0x71,DectoBCD (tempMillennium));
213
214
            outb (0x70, 0x09);
            outb(0x71,DectoBCD (tempYear));
215
216
            sti();
217
218
        else
        printf("Date Set\n");
}
219
220
```

#### 5.23.1.11 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 108 of file userFunctions.c.

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

#### 5.23.1.12 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 439 of file userFunctions.c.

```
439
440
       int check = 20;
441
       int i;
442
       int j;
443
       for(i = 0; i < sizeof(ready queue);i++)</pre>
          char rProcess_Name = ready queue [i] Process_Name;
int rClass = ready queue [i] class;
char rState = ready queue[i] state;
444
445
446
          char rStatus = ready queue[i] status;
448
          int rPriority = ready queue[i] priority;
449
          sys_req(WRITE, COM1, rProcess_Name, &check);
450
          sys_req(WRITE, COM1, itoa(rClass), &check);
451
          sys_req(WRITE, COM1, rState, &check);
          sys_req(WRITE, COM1, rStatus, &check);
sys_req(WRITE, COM1, itoa(rPriority), &check);
452
453
454
```

```
455
      for(j = 0; j < sizeof(blocked queue); j++) {</pre>
456
       char bProcess_Name = blocked queue [j] Process_Name;
        int bClass = blocked queue [j] class;
char bState = blocked queue[j] state;
457
458
        char bStatus = blocked queue[j] status;
459
        int bPriority = blocked queue[j] priority;
460
461
        sys_req(WRITE, COM1, bProcess_Name, &check);
        sys_req(WRITE, COM1, itoa(bClass), &check);
462
463
        sys_req(WRITE, COM1, bState, &check);
464
        sys_req(WRITE, COM1, bStatus, &check);
465
        sys_req(WRITE, COM1, itoa(bPriority), &check);
466
467 }
```

#### 5.23.1.13 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. 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.	

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.
--------------	---

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.

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 492 of file userFunctions.c.

#### 5.23.1.14 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 472 of file userFunctions.c.

```
473
       int check = 20;
474
       int i;
475
       for(i = 0; i < sizeof(ready queue);i++)</pre>
476
         char Process_Name = ready queue [i] Process_Name;
         char Class = ready queue [i] class;
char State = ready queue[i] state;
477
         char Status = ready queue[i] status;
480
         char Priority = ready queue[i] priority;
         sys_req(WRITE, COM1, Process_Name, &check);
481
482
         sys_req(WRITE, COM1, Class, &check);
         sys_req(WRITE, COM1, State, &check);
sys_req(WRITE, COM1, Status, &check);
483
484
485
         sys_req(WRITE, COM1, Priority, &check);
487 }
```

### 5.23.1.15 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 356 of file userFunctions.c.

```
356 {
357
```

```
358  // Name Error check
359  // Error check (Valid Name)
360  //if (Process_Name != valid name) {
361  // printf("\xlb[3lm""\nERROR: Not a valid process name \n""\xlb[0m");
362  //}
363  364 }
```

### 5.23.1.16 toLowercase()

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

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

(char)

#### **Parameters**

c Character that is to be changed to its lowercase equivalent

Definition at line 262 of file userFunctions.c.

### 5.23.1.17 Version()

```
void Version ( )
```

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

(the iteration that module is currently on).

No parameters.

Definition at line 255 of file userFunctions.c.

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

# Index

atoi	string.h, 19
string.c, 28	mpx_core/include/core/asm.h, 15
string.h, 18	mpx_core/include/core/interrupts.h, 15
	mpx_core/include/core/io.h, 15
BCDtoDec	mpx_core/include/core/serial.h, 16
userFunctions.c, 35	mpx_core/include/core/tables.h, 16
userFunctions.h, 47	mpx_core/include/mem/heap.h, 17
data than 7	mpx_core/include/mem/paging.h, 17
date_time, 7	mpx_core/include/string.h, 18
DectoBCD	mpx_core/include/system.h, 22
userFunctions.c, 35	mpx_core/kernel/core/interrupts.c, 23
userFunctions.h, 47	mpx_core/kernel/core/kmain.c, 24
EdgeCase	mpx_core/kernel/core/serial.c, 25
userFunctions.c, 36	mpx_core/kernel/core/system.c, 25
userFunctions.h, 48	mpx_core/kernel/core/tables.c, 25
doctr dilotions.ii, 40	mpx_core/kernel/mem/heap.c, 26
footer, 7	mpx_core/kernel/mem/paging.c, 26
	mpx_core/lib/string.c, 27
gdt_descriptor_struct, 8	mpx_core/modules/mpx_supt.c, 32
gdt_entry_struct, 8	mpx_core/modules/mpx_supt.h, 32
GetDate	mpx_core/modules/R1/comHand.c, 33
userFunctions.c, 36	mpx_core/modules/R1/comHand.h, 34
userFunctions.h, 48	mpx_core/modules/R1/userFunctions.c, 34
GetTime	mpx_core/modules/R1/userFunctions.h, 46
userFunctions.c, 37	
userFunctions.h, 49	page_dir, 11
	page_entry, 11
header, 8	page_table, 11
heap, 9	param, 12
Help	PCB, 12
userFunctions.c, 37	Output 12
userFunctions.h, 49	Queue, 13
idt optry struct 0	Resume
idt_entry_struct, 9	userFunctions.c, 39
idt_struct, 10 inb	userFunctions.h, 51
io.h, 15 index_entry, 10	Set_Priority
index_entry, 10	userFunctions.c, 40
io.h	userFunctions.h, 52
inb, 15	SetDate
isspace	userFunctions.c, 40
string.c, 28	userFunctions.h, 52
string.h, 19	SetTime
itoa	userFunctions.c, 41
userFunctions.c, 39	userFunctions.h, 53
userFunctions.h, 51	Show_All
	userFunctions.c, 42
memset	userFunctions.h, 54
string.c, 29	Show_Blocked

60 INDEX

	0.0.40
userFunctions.c, 42	SetDate, 40
userFunctions.h, 55	SetTime, 41
Show_PCB	Show_All, 42
userFunctions.c, 43 Show_Ready	Show_Blocked, 42 Show PCB, 43
userFunctions.c, 44	Show_Ready, 44
userFunctions.h, 56	Suspend, 44
streat	toLowercase, 45
string.c, 29	Version, 45
string.c, 29 string.h, 20	userFunctions.h
strcmp	BCDtoDec, 47
string.c, 29	DectoBCD, 47
string.h, 20	EdgeCase, 48
strcpy	GetDate, 48
string.c, 30	GetTime, 49
string.h, 21	Help, 49
string.c	itoa, 51
atoi, 28	Resume, 51
isspace, 28	Set_Priority, 52
memset, 29	SetDate, 52
strcat, 29	SetTime, 53
strcmp, 29	Show All, 54
strcpy, 30	Show Blocked, 55
strlen, 30	Show_Ready, 56
strtok, 31	Suspend, 56
string.h	toLowercase, 57
atoi, 18	Version, 57
isspace, 19	
memset, 19	Version
strcat, 20	userFunctions.c, 45
strcmp, 20	userFunctions.h, 57
strcpy, 21	
strlen, 21	
strtok, 21	
strlen	
string.c, 30	
string.h, 21	
strtok	
strtok string.c, 31	
strtok string.c, 31 string.h, 21	
strtok string.c, 31 string.h, 21 struct, 13	
strtok string.c, 31 string.h, 21 struct, 13 Suspend	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44	
strtok string.c, 31 string.h, 21 struct, 13 Suspend	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56 toLowercase	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56  toLowercase userFunctions.c, 45	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56 toLowercase	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56  toLowercase userFunctions.c, 45	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56  toLowercase userFunctions.c, 45 userFunctions.h, 57	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56  toLowercase userFunctions.c, 45 userFunctions.h, 57  userFunctions.c BCDtoDec, 35 DectoBCD, 35	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56  toLowercase userFunctions.c, 45 userFunctions.h, 57  userFunctions.c BCDtoDec, 35	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56  toLowercase userFunctions.c, 45 userFunctions.h, 57  userFunctions.c BCDtoDec, 35 DectoBCD, 35 EdgeCase, 36 GetDate, 36	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56  toLowercase userFunctions.c, 45 userFunctions.h, 57  userFunctions.c BCDtoDec, 35 DectoBCD, 35 EdgeCase, 36 GetDate, 36 GetTime, 37	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56  toLowercase userFunctions.c, 45 userFunctions.h, 57  userFunctions.c BCDtoDec, 35 DectoBCD, 35 EdgeCase, 36 GetDate, 36 GetTime, 37 Help, 37	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56  toLowercase userFunctions.c, 45 userFunctions.h, 57  userFunctions.c BCDtoDec, 35 DectoBCD, 35 EdgeCase, 36 GetDate, 36 GetTime, 37 Help, 37 itoa, 39	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56  toLowercase userFunctions.c, 45 userFunctions.h, 57  userFunctions.c BCDtoDec, 35 DectoBCD, 35 EdgeCase, 36 GetDate, 36 GetTime, 37 Help, 37 itoa, 39 Resume, 39	
strtok string.c, 31 string.h, 21 struct, 13 Suspend userFunctions.c, 44 userFunctions.h, 56  toLowercase userFunctions.c, 45 userFunctions.h, 57  userFunctions.c BCDtoDec, 35 DectoBCD, 35 EdgeCase, 36 GetDate, 36 GetTime, 37 Help, 37 itoa, 39	