

RUNTIME TERROR OS

R5

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 CMCB Struct Reference	5
3.2.1 Detailed Description	6
3.3 context Struct Reference	6
3.3.1 Detailed Description	6
3.4 date_time Struct Reference	6
3.4.1 Detailed Description	7
3.5 footer Struct Reference	7
3.5.1 Detailed Description	7
3.6 gdt_descriptor_struct Struct Reference	7
3.6.1 Detailed Description	7
3.7 gdt_entry_struct Struct Reference	7
3.7.1 Detailed Description	8
3.8 header Struct Reference	8
3.8.1 Detailed Description	8
3.9 heap Struct Reference	8
3.9.1 Detailed Description	8
3.10 idt_entry_struct Struct Reference	9
3.10.1 Detailed Description	9
3.11 idt_struct Struct Reference	9
3.11.1 Detailed Description	9
3.12 index_entry Struct Reference	9
3.12.1 Detailed Description	10
3.13 index_table Struct Reference	10
3.13.1 Detailed Description	10
3.14 List Struct Reference	10
3.14.1 Detailed Description	10
3.15 MemList Struct Reference	10
3.15.1 Detailed Description	11
3.16 page_dir Struct Reference	11
3.16.1 Detailed Description	11
3.17 page_entry Struct Reference	11
3.17.1 Detailed Description	11
3.18 page_table Struct Reference	12
3.18.1 Detailed Description	12

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

4.17.1.6 strcpy()	30
4.17.1.7 strlen()	30
4.17.1.8 strtok()	30
4.18 modules/mpx_supt.c File Reference	31
4.19 modules/mpx_supt.h File Reference	32
4.20 modules/R1/comHand.h File Reference	33
4.20.1 Function Documentation	33
4.20.1.1 comHand()	33
4.21 modules/R1/userFunctions.c File Reference	37
4.21.1 Function Documentation	38
4.21.1.1 BCDtoDec()	38
4.21.1.2 Block()	39
4.21.1.3 Create_PCB()	39
4.21.1.4 DectoBCD()	40
4.21.1.5 Delete_PCB()	40
4.21.1.6 EdgeCase()	41
4.21.1.7 GetDate()	41
4.21.1.8 GetTime()	42
4.21.1.9 Help()	43
4.21.1.10 itoa()	44
4.21.1.11 Resume()	45
4.21.1.12 Set_Priority()	46
4.21.1.13 SetDate()	46
4.21.1.14 SetTime()	47
4.21.1.15 Show_All()	48
4.21.1.16 Show_Blocked()	48
4.21.1.17 Show_PCB()	50
4.21.1.18 Show_Ready()	51
4.21.1.19 Suspend()	53
4.21.1.20 toLowercase()	53
4.21.1.21 Unblock()	54
4.21.1.22 Version()	54
4.21.2 Variable Documentation	54
4.21.2.1 AlarmList	55
4.21.2.2 else	55
4.22 modules/R1/userFunctions.h File Reference	55
4.22.1 Function Documentation	56
4.22.1.1 BCDtoDec()	56
4.22.1.2 Block()	57
4.22.1.3 Create_PCB()	57
4.22.1.4 DectoBCD()	58
4.22.1.5 Delete_PCB()	58

4.22.1.6 EdgeCase()	59
4.22.1.7 GetDate()	59
4.22.1.8 GetTime()	60
4.22.1.9 Help()	61
4.22.1.10 itoa()	62
4.22.1.11 Resume()	63
4.22.1.12 Set_Priority()	64
4.22.1.13 SetDate()	64
4.22.1.14 SetTime()	65
4.22.1.15 Show_All()	66
4.22.1.16 Show_Blocked()	66
4.22.1.17 Show_PCB()	68
4.22.1.18 Show_Ready()	69
4.22.1.19 Suspend()	71
4.22.1.20 toLowercase()	71
4.22.1.21 Unblock()	72
4.22.1.22 Version()	72
4.23 modules/sys_proc_loader.c File Reference	73
4.24 modules/sys_proc_loader.h File Reference	73

Index	75
--------------	-----------

Chapter 1

Class Index

1.1 Class List

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

Alarm	5
CMCB	5
context	6
date_time	6
footer	7
gdt_descriptor_struct	7
gdt_entry_struct	7
header	8
heap	8
idt_entry_struct	9
idt_struct	9
index_entry	9
index_table	10
List	10
MemList	10
page_dir	11
page_entry	11
page_table	12
param	12
PCB	12
Queue	13

Chapter 2

File Index

2.1 File List

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

include/string.h	18
include/system.h	22
include/core/asm.h	15
include/core/interrupts.h	15
include/core/io.h	15
include/core/serial.h	16
include/core/tables.h	16
include/mem/heap.h	17
include/mem/paging.h	17
kernel/core/interrupts.c	23
kernel/core/kmain.c	24
kernel/core/serial.c	24
kernel/core/system.c	25
kernel/core/tables.c	25
kernel/mem/heap.c	26
kernel/mem/paging.c	26
lib/string.c	27
modules/mpx_supt.c	31
modules/mpx_supt.h	32
modules/procsr3.c	??
modules/procsr3.h	??
modules/sys_proc_loader.c	73
modules/sys_proc_loader.h	73
modules/R1/comHand.c	??
modules/R1/comHand.h	33
modules/R1/userFunctions.c	37
modules/R1/userFunctions.h	55
modules/R2/PCB.c	??
modules/R2/PCB.h	??
modules/R5/MCB.c	??
modules/R5/MCB.h	??
modules/R5/R5commands.c	??
modules/R5/R5commands.h	??

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 CMCB Struct Reference

Public Attributes

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

3.2.1 Detailed Description

Definition at line 4 of file MCB.h.

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

- modules/R5/MCB.h

3.3 context Struct Reference

Public Attributes

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

3.3.1 Detailed Description

Definition at line 34 of file PCB.h.

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

- modules/R2/PCB.h

3.4 date_time Struct Reference

Public Attributes

- int **sec**
- int **min**
- int **hour**
- int **day_w**
- int **day_m**
- int **day_y**
- int **mon**
- int **year**

3.4.1 Detailed Description

Definition at line 32 of file system.h.

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

- [include/system.h](#)

3.5 footer Struct Reference

Public Attributes

- [header](#) **head**

3.5.1 Detailed Description

Definition at line 18 of file heap.h.

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

- [include/mem/heap.h](#)

3.6 gdt_descriptor_struct Struct Reference

Public Attributes

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

3.6.1 Detailed Description

Definition at line 25 of file tables.h.

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

- [include/core/tables.h](#)

3.7 gdt_entry_struct Struct Reference

Public Attributes

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

3.7.1 Detailed Description

Definition at line 32 of file tables.h.

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

- [include/core/tables.h](#)

3.8 header Struct Reference

Public Attributes

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

3.8.1 Detailed Description

Definition at line 13 of file heap.h.

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

- [include/mem/heap.h](#)

3.9 heap Struct Reference

Public Attributes

- [index_table](#) **index**
- u32int **base**
- u32int **max_size**
- u32int **min_size**

3.9.1 Detailed Description

Definition at line 35 of file heap.h.

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

- [include/mem/heap.h](#)

3.10 idt_entry_struct Struct Reference

Public Attributes

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

3.10.1 Detailed Description

Definition at line 8 of file tables.h.

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

- include/core/[tables.h](#)

3.11 idt_struct Struct Reference

Public Attributes

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

3.11.1 Detailed Description

Definition at line 18 of file tables.h.

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

- include/core/[tables.h](#)

3.12 index_entry Struct Reference

Public Attributes

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

3.12.1 Detailed Description

Definition at line 22 of file heap.h.

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

- [include/mem/heap.h](#)

3.13 index_table Struct Reference

Public Attributes

- [index_entry](#) **table** [TABLE_SIZE]
- **int** **id**

3.13.1 Detailed Description

Definition at line 29 of file heap.h.

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

- [include/mem/heap.h](#)

3.14 List Struct Reference

Public Attributes

- [Alarm](#) * **head**
- [Alarm](#) * **tail**

3.14.1 Detailed Description

Definition at line 24 of file userFunctions.h.

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

- [modules/R1/userFunctions.h](#)

3.15 MemList Struct Reference

Public Attributes

- [CMCB](#) * **head**

3.15.1 Detailed Description

Definition at line 18 of file `MCB.h`.

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

- `modules/R5/MCB.h`

3.16 `page_dir` Struct Reference

Public Attributes

- [page_table](#) * `tables` [1024]
- `u32int tables_phys` [1024]

3.16.1 Detailed Description

Definition at line 36 of file `paging.h`.

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

- `include/mem/paging.h`

3.17 `page_entry` Struct Reference

Public Attributes

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

3.17.1 Detailed Description

Definition at line 14 of file `paging.h`.

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

- `include/mem/paging.h`

3.18 page_table Struct Reference

Public Attributes

- [page_entry](#) **pages** [1024]

3.18.1 Detailed Description

Definition at line 28 of file `paging.h`.

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

- `include/mem/paging.h`

3.19 param Struct Reference

Public Attributes

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

3.19.1 Detailed Description

Definition at line 34 of file `mpx_supt.h`.

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

- `modules/mpx_supt.h`

3.20 PCB Struct Reference

Public Attributes

- unsigned char **stack** [MEM1K]
- unsigned char * **stackTop**
- struct [PCB](#) * **prev**
- struct [PCB](#) * **next**
- char **Process_Name** [10]
- int **Process_Class**
- int **Priority**
- int **ReadyState**
- int **SuspendedState**

3.20.1 Detailed Description

Definition at line 15 of file PCB.h.

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

- modules/R2/PCB.h

3.21 Queue Struct Reference

Public Attributes

- int **count**
- [PCB](#) * **head**
- [PCB](#) * **tail**

3.21.1 Detailed Description

Definition at line 27 of file PCB.h.

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

- modules/R2/PCB.h

Chapter 4

File Documentation

4.1 include/core/asm.h File Reference

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

4.2 include/core/interrupts.h File Reference

Functions

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

4.3 include/core/io.h File Reference

Macros

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

4.3.1 Macro Definition Documentation

4.3.1.1 inb

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

Value:

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

Definition at line 17 of file io.h.

4.4 include/core/serial.h File Reference

Macros

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

Functions

- `int init_serial (int device)`
- `int serial_println (const char *msg)`
- `int serial_print (const char *msg)`
- `int set_serial_out (int device)`
- `int set_serial_in (int device)`
- `int * polling (char *buffer, int *count)`

4.5 include/core/tables.h File Reference

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

Classes

- struct [idt_entry_struct](#)
- struct [idt_struct](#)
- struct [gdt_descriptor_struct](#)
- struct [gdt_entry_struct](#)

Functions

- struct [idt_entry_struct](#) `__attribute__((packed)) idt_entry`
- void `idt_set_gate (u8int idx, u32int base, u16int sel, u8int flags)`
- void `gdt_init_entry (int idx, u32int base, u32int limit, u8int access, u8int flags)`
- void `init_idt ()`
- void `init_gdt ()`

Variables

- u16int `base_low`
- u16int `sselect`
- u8int `zero`
- u8int `flags`
- u16int `base_high`
- u16int `limit`
- u32int `base`
- u16int `limit_low`
- u8int `base_mid`
- u8int `access`

4.6 include/mem/heap.h File Reference

Classes

- struct [header](#)
- struct [footer](#)
- struct [index_entry](#)
- struct [index_table](#)
- struct [heap](#)

Macros

- #define **TABLE_SIZE** 0x1000
- #define **KHEAP_BASE** 0xD000000
- #define **KHEAP_MIN** 0x10000
- #define **KHEAP_SIZE** 0x1000000

Functions

- u32int **_kmalloc** (u32int size, int align, u32int *phys_addr)
- u32int **kmalloc** (u32int size)
- u32int **kfree** ()
- void **init_kheap** ()
- u32int **alloc** (u32int size, [heap](#) *hp, int align)
- [heap](#) * **make_heap** (u32int base, u32int max, u32int min)

4.7 include/mem/paging.h File Reference

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

Classes

- struct [page_entry](#)
- struct [page_table](#)
- struct [page_dir](#)

Macros

- #define **PAGE_SIZE** 0x1000

Functions

- void **set_bit** (u32int addr)
- void **clear_bit** (u32int addr)
- u32int **get_bit** (u32int addr)
- u32int **first_free** ()
- void **init_paging** ()
- void **load_page_dir** ([page_dir](#) *new_page_dir)
- [page_entry](#) * **get_page** (u32int addr, [page_dir](#) *dir, int make_table)
- void **new_frame** ([page_entry](#) *page)

4.8 include/string.h File Reference

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

Functions

- int [isspace](#) (const char *c)
- void * [memset](#) (void *s, int c, size_t n)
- char * [strcpy](#) (char *s1, const char *s2)
- char * [strcat](#) (char *s1, const char *s2)
- int [strlen](#) (const char *s)
- int [strcmp](#) (const char *s1, const char *s2)
- char * [strtok](#) (char *s1, const char *s2)
- int [atoi](#) (const char *s)

4.8.1 Function Documentation

4.8.1.1 atoi()

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

Description: Convert an ASCII string to an integer

Parameters

s	String
---	--------

Definition at line 50 of file string.c.

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


4.8.1.2 isspace()

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

Description: Determine if a character is whitespace.

Parameters

<i>c</i>	character to check
----------	--------------------

Definition at line 121 of file string.c.

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

4.8.1.3 memset()

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

Description: Set a region of memory.

Parameters

<i>s</i>	destination
<i>c</i>	byte to write
<i>n</i>	count

Definition at line 139 of file string.c.

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

4.8.1.4 strcat()

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

Description: Concatenate the contents of one string onto another.

Parameters

<i>s1</i>	destination
<i>s2</i>	source

Definition at line 108 of file string.c.

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

4.8.1.5 strcmp()

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

Description: String comparison

Parameters

<i>s1</i>	string 1
<i>s2</i>	string 2

Definition at line 81 of file string.c.

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

4.8.1.6 strcpy()

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

Description: Copy one string to another.

Parameters

<i>s1</i>	destination
<i>s2</i>	source

Definition at line 38 of file string.c.

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

4.8.1.7 strlen()

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

Description: Returns the length of a string.

Parameters

<i>s</i>	input string
----------	--------------

Definition at line 26 of file string.c.

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

4.8.1.8 strtok()

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

Description: Split string into tokens

Parameters

<i>s1</i>	String
<i>s2</i>	delimiter

Definition at line 153 of file string.c.

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

```

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

```

4.9 include/system.h File Reference

Classes

- struct [date_time](#)

Macros

- #define **NULL** 0
- #define **no_warn**(p) if (p) while (1) break
- #define **asm** __asm__
- #define **volatile** __volatile__
- #define **sti**() asm volatile ("sti::")
- #define **cli**() asm volatile ("cli::")
- #define **nop**() asm volatile ("nop::")
- #define **hlt**() asm volatile ("hlt::")
- #define **iret**() asm volatile ("iret::")
- #define **GDT_CS_ID** 0x01
- #define **GDT_DS_ID** 0x02

Typedefs

- typedef unsigned int **size_t**
- typedef unsigned char **u8int**
- typedef unsigned short **u16int**
- typedef unsigned long **u32int**

Functions

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

4.10 kernel/core/interrupts.c File Reference

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

Macros

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

Functions

- void **divide_error** ()
- void **debug** ()
- void **nmi** ()
- void **breakpoint** ()
- void **overflow** ()
- void **bounds** ()
- void **invalid_op** ()
- void **device_not_available** ()
- void **double_fault** ()
- void **coprocessor_segment** ()
- void **invalid_tss** ()
- void **segment_not_present** ()
- void **stack_segment** ()
- void **general_protection** ()
- void **page_fault** ()
- void **reserved** ()
- void **coprocessor** ()
- void **rtc_isr** ()
- void **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 <system.h>
#include <core/io.h>
#include <core/serial.h>
#include <core/tables.h>
#include <core/interrupts.h>
#include <mem/heap.h>
#include <mem/paging.h>
#include <modules/mpx_supt.h>
#include "modules/R1/comHand.h"
#include "modules/sys_proc_loader.h"
#include "modules/R1/userFunctions.h"
#include "modules/R5/MCB.h"
```

Functions

- void **kmain** (void)

4.12 kernel/core/serial.c File Reference

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

Macros

- #define **NO_ERROR** 0

Functions

- int **init_serial** (int device)
- int **serial_println** (const char *msg)
- int **serial_print** (const char *msg)
- int **set_serial_out** (int device)
- int **set_serial_in** (int device)
- int * **polling** (char *cmdBuffer, int *count)

Variables

- int **serial_port_out** = 0
- int **serial_port_in** = 0

4.13 kernel/core/system.c File Reference

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

Functions

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

4.14 kernel/core/tables.c File Reference

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

Functions

- void **write_gdt_ptr** (u32int, size_t)
- void **write_idt_ptr** (u32int)
- void **idt_set_gate** (u8int idx, u32int base, u16int sel, u8int flags)
- void **init_idt** ()
- void **gdt_init_entry** (int idx, u32int base, u32int limit, u8int access, u8int flags)
- void **init_gdt** ()

Variables

- gdt_descriptor **gdt_ptr**
- gdt_entry **gdt_entries** [5]
- idt_descriptor **idt_ptr**
- idt_entry **idt_entries** [256]

4.15 kernel/mem/heap.c File Reference

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

Functions

- u32int **_kmalloc** (u32int size, int page_align, u32int *phys_addr)
- u32int **kmalloc** (u32int size)
- u32int **alloc** (u32int size, heap *h, int align)
- heap * **make_heap** (u32int base, u32int max, u32int min)

Variables

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

4.16 kernel/mem/paging.c File Reference

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

Functions

- void **set_bit** (u32int addr)
- void **clear_bit** (u32int addr)
- u32int **get_bit** (u32int addr)
- u32int **find_free** ()
- page_entry * **get_page** (u32int addr, page_dir *dir, int make_table)
- void **init_paging** ()
- void **load_page_dir** (page_dir *new_dir)
- void **new_frame** (page_entry *page)

Variables

- u32int **mem_size** = 0x4000000
- u32int **page_size** = 0x1000
- u32int **nframes**
- u32int * **frames**
- [page_dir](#) * **kdir** = 0
- [page_dir](#) * **cdir** = 0
- u32int **phys_alloc_addr**
- [heap](#) * **kheap**

4.17 lib/string.c File Reference

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

Functions

- int [strlen](#) (const char *s)
- char * [strcpy](#) (char *s1, const char *s2)
- int [atoi](#) (const char *s)
- int [strcmp](#) (const char *s1, const char *s2)
- char * [strcat](#) (char *s1, const char *s2)
- int [isspace](#) (const char *c)
- void * [memset](#) (void *s, int c, size_t n)
- char * [strtok](#) (char *s1, const char *s2)

4.17.1 Function Documentation

4.17.1.1 atoi()

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

Description: Convert an ASCII string to an integer

Parameters

s	String
----------	--------

Definition at line 50 of file string.c.

```
51 {
52     int res=0;
53     int charVal=0;
54     char sign = ' ';
55     char c = *s;
```

```

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

```

4.17.1.2 isspace()

```

int isspace (
    const char * c )

```

Description: Determine if a character is whitespace.

Parameters

<i>c</i>	character to check
----------	--------------------

Definition at line 121 of file string.c.

```

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

```

4.17.1.3 memset()

```

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

```

Description: Set a region of memory.

Parameters

<i>s</i>	destination
<i>c</i>	byte to write
<i>n</i>	count

Definition at line 139 of file string.c.

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

4.17.1.4 strcat()

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

Description: Concatenate the contents of one string onto another.

Parameters

<i>s1</i>	destination
<i>s2</i>	source

Definition at line 108 of file string.c.

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

4.17.1.5 strcmp()

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

Description: String comparison

Parameters

<i>s1</i>	string 1
<i>s2</i>	string 2

Definition at line 81 of file string.c.

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

```
93  }
94  return ( *(unsigned char *)s1 - *(unsigned char *)s2 );
95 }
```

4.17.1.6 strcpy()

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

Description: Copy one string to another.

Parameters

<i>s1</i>	destination
<i>s2</i>	source

Definition at line 38 of file string.c.

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

4.17.1.7 strlen()

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

Description: Returns the length of a string.

Parameters

<i>s</i>	input string
----------	--------------

Definition at line 26 of file string.c.

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

4.17.1.8 strtok()

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

Description: Split string into tokens

Parameters

<i>s1</i>	String
<i>s2</i>	delimiter

Definition at line 153 of file string.c.

```

154 {
155     static char *tok_tmp = NULL;
156     const char *p = s2;
157
158     //new string
159     if (s1!=NULL){
160         tok_tmp = s1;
161     }
162     //old string cont'd
163     else {
164         if (tok_tmp==NULL){
165             return NULL;
166         }
167         s1 = tok_tmp;
168     }
169
170     //skip leading s2 characters
171     while ( *p && *s1 ){
172         if (*s1==*p){
173             ++s1;
174             p = s2;
175             continue;
176         }
177         ++p;
178     }
179
180     //no more to parse
181     if (!*s1){
182         return (tok_tmp = NULL);
183     }
184
185     //skip non-s2 characters
186     tok_tmp = s1;
187     while (*tok_tmp){
188         p = s2;
189         while (*p){
190             if (*tok_tmp==*p++){
191                 *tok_tmp++ = '\0';
192                 return s1;
193             }
194         }
195         ++tok_tmp;
196     }
197
198     //end of string
199     tok_tmp = NULL;
200     return s1;
201 }
```

4.18 modules/mpx_supt.c File Reference

```

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

Functions

- int **sys_req** (int op_code, int device_id, char *buffer_ptr, int *count_ptr)
- void **mpx_init** (int cur_mod)
- void **sys_set_malloc** (u32int(*func)(u32int))
- void **sys_set_free** (int(*func)(void *))

- void * **sys_alloc_mem** (u32int size)
- int **sys_free_mem** (void *ptr)
- void **idle** ()
- 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 23 of file comHand.c.

```

23         {
24
25             Help("\0");
26
27             char cmdBuffer[100];
28             int bufferSize = 99;
29             int quit = 0;
30             int shutdown = 0;
31
32             while(quit != 1) {
33                 memset(cmdBuffer, '\0', 100);
34                 sys_req(READ, DEFAULT_DEVICE, cmdBuffer, &bufferSize);
35                 char* FirstToken = strtok(cmdBuffer, "-");
36                 char* SecondToken = strtok(NULL, "-");
37                 char* ThirdToken = strtok(NULL, "-");
38                 char* FourthToken = strtok(NULL, "-");
39                 char* FifthToken = strtok(NULL, "-");
40                 if(shutdown == 0) {
41 /*****
42             R1 comHand
43 *****/
44                     if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,NULL) == 0) {
45                         Help("\0");
46                     }
47                     //R1 Commands
48                     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"version") == 0 &&
49 strcmp(ThirdToken,NULL) == 0) {
50                         Help("Version");
51                     }
52                     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"getDate") == 0 &&
53 strcmp(ThirdToken,NULL) == 0) {
54                         Help("GetDate");
55                     }
56                     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"setDate") == 0 &&
57 strcmp(ThirdToken,NULL) == 0) {

```

```

55         Help("SetDate");
56     }
57     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"getTime") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
58         Help("GetTime");
59     }
60     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"setTime") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
61         Help("SetTime");
62     }
63     // R2 Commands
64     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"suspend") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
65         Help("suspend");
66     }
67     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"resume") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
68         Help("resume");
69     }
70     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"setPriority") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
71         Help("setPriority");
72     }
73     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showPCB") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
74         Help("showPCB");
75     }
76     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showAll") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
77         Help("showAll");
78     }
79     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showReady") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
80         Help("showReady");
81     }
82     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"showBlocked") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
83         Help("showBlocked");
84     }
85     // Temporary R2 commands
86     // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"createPCB") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
87         // Help("createPCB");
88         // }
89     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"deletePCB") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
90         Help("deletePCB");
91     }
92     // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"block") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
93         // Help("block");
94         // }
95     // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"unblock") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
96         // Help("unblock");
97         // }
98     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"shutdown") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
99         Help("shutdown");
100     }
101     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"infinite") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
102         Help("infinte");
103     }
104     // R4 Commands
105     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"loadr3") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
106         Help("loadr3");
107     }
108     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"alarm") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
109         Help("alarm");
110     }
111     // Bonus Command
112     else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
113         Help("clear");
114     }
115     // Temporary R5 Commands
116     // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
117         // Help("heap");
118         // }
119     // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
120         // Help("alloc");
121         // }

```



```

122         // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
123         // Help("free");
124         // }
125         // else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
126         // Help("empty");
127         // }
128         // R5 Commands
129         else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
130             Help("showFree");
131         }
132         else if(strcmp(FirstToken,"help") == 0 && strcmp(SecondToken,"clear") == 0 &&
strcmp(ThirdToken,NULL) == 0) {
133             Help("showAlloc");
134         }
135
136
137
138
139
140
141         else if(strcmp(FirstToken,"version") == 0 && strcmp(SecondToken,NULL) == 0)
142             Version();
143         else if(strcmp(FirstToken,"clear") == 0 && strcmp(SecondToken,NULL) == 0)
144             clear();
145
146         else if(strcmp(FirstToken,"getDate") == 0 && strcmp(SecondToken,NULL) == 0)
147             GetDate();
148
149         else if(strcmp(FirstToken,"setDate") == 0){
150             if (EdgeCase(SecondToken) == 1 && EdgeCase(ThirdToken) == 1 &&
EdgeCase(FourthToken) == 1 && EdgeCase(FifthToken) == 1) {
151                 SetDate(atoi(SecondToken), atoi(ThirdToken), atoi(FourthToken),
atoi(FifthToken));
152             }
153             else
154                 printf("\x1b[31m"\nERROR: Invalid parameters for setDate \n"\x1b[0m");
155         }
156         else if(strcmp(FirstToken,"getTime") == 0 && strcmp(SecondToken,NULL) == 0) //Return
the current time
held by the registers.
157             GetTime();
158         else if(strcmp(FirstToken,"setTime") == 0 && strcmp(FifthToken,NULL) == 0){
159             if (EdgeCase(SecondToken) == 1 && EdgeCase(ThirdToken) == 1 &&
EdgeCase(FourthToken) == 1) {
160                 SetTime(atoi(SecondToken), atoi(ThirdToken), atoi(FourthToken));
//input as Hour-Minute-Seconds
161             }
162             else
163                 printf("\x1b[31m"\nERROR: Invalid parameters for setTime \n"\x1b[0m");
164         }
165
166
167
168
169
170
171         /*****
172             R2 comHand
173             *****/
174         else if(strcmp(FirstToken,"suspend") == 0 && strcmp(ThirdToken,NULL) == 0 &&
strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
175             Suspend(SecondToken);
176         }
177         else if(strcmp(FirstToken,"resume") == 0 && strcmp(ThirdToken,NULL) == 0 &&
strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
178             Resume(SecondToken);
179         }
180         else if(strcmp(FirstToken,"setPriority") == 0 && strcmp(FourthToken,NULL) == 0 &&
strcmp(FifthToken,NULL) == 0) {
181             if(EdgeCase(ThirdToken) == 1) {
182                 Set_Priority(SecondToken, atoi(ThirdToken)); //input as
setPriority-Process_Name-Priority
183             }
184             else
185                 printf("\x1b[31m"\nERROR: Invalid parameters for setPriority, priority must
be entered as a integer. \n"\x1b[0m");
186         }
187         else if(strcmp(FirstToken,"showPCB") == 0 && strcmp(ThirdToken,NULL) == 0 &&
strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
188             Show_PCB(SecondToken);
189             printf("\n");
190         }
191         else if(strcmp(FirstToken,"showAll") == 0 && strcmp(SecondToken,NULL) == 0 &&
strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
192             Show_All();

```

```

193         printf("\n");
194     }
195     else if(strcmp(FirstToken,"showReady") == 0 && strcmp(SecondToken,NULL) == 0 &&
strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
196         Show_Ready();
197         printf("\n");
198     }
199     else if(strcmp(FirstToken,"showBlocked") == 0 && strcmp(SecondToken,NULL) == 0 &&
strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
200         Show_Blocked();
201         printf("\n");
202     }
203
204
205
206         /***** R2 Temp Commands *****/
207         //Removed from active for R3/R4
208         /*
209         else if(strcmp(FirstToken,"createPCB") == 0) {
210             if( strlen(SecondToken) < 11) {
211                 Create_PCB(SecondToken, atoi(ThirdToken), atoi(FourthToken));
212             }
213             //input as Process_Name-Priority-Class
214             else
215                 printf("\x1b[31m"\nERROR: Invalid parameters for createPCB, Process_name
must only contain 10 or fewer characters. \n"\x1b[0m");
216         }
217         */
218         else if(strcmp(FirstToken,"deletePCB") == 0 && strcmp(ThirdToken,NULL) == 0 &&
strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
219             Delete_PCB(SecondToken);
220         }
221
222
223         //Removed from active for R3/R4
224         /*
225         else if(strcmp(FirstToken,"block") == 0 && strcmp(ThirdToken,NULL) == 0 &&
strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
226             Block(SecondToken);
227         }
228         else if(strcmp(FirstToken,"unblock") == 0 && strcmp(ThirdToken,NULL) == 0 &&
strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
229             Unblock(SecondToken);
230         }
231         */
232
233         /*****
234         R3 comHand
235         *****/
236         //Removed for R4
237         /*
238         else if(strcmp(FirstToken,"yield") == 0 && strcmp(SecondToken,NULL) == 0 &&
strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
239             yield();
240             printf("\n");
241         }
242         else if(strcmp(FirstToken,"loadr3") == 0 && strcmp(SecondToken,NULL) == 0 &&
strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
243             loader();
244             printf("\n");
245         }
246         */
247
248         /*****
249         R4 comHand
250         *****/
251         else if(strcmp(FirstToken,"alarm") == 0) {
252             if (EdgeCase(ThirdToken) == 1 && EdgeCase(FourthToken) == 1 &&
EdgeCase(FifthToken) == 1) {
253                 if (atoi(ThirdToken) < 24 && atoi(FourthToken) < 60 && atoi(FifthToken) <
60) {
254                     loaderalarm(SecondToken, atoi(ThirdToken), atoi(FourthToken),
atoi(FifthToken));
255                     printf("\n"); //input as Message-Hour-Minute-Seconds
256                 }
257                 else
258                     printf("\x1b[31m"\nERROR: Invalid parameters for alarm, must be a valid
time \n"\x1b[0m");
259             }
260             else
261                 printf("\x1b[31m"\nERROR: Invalid parameters for alarm \n"\x1b[0m");
262         }
263         else if(strcmp(FirstToken,"loadr3") == 0 && strcmp(SecondToken,NULL) == 0 &&
strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
264             loader();
265             printf("\n");
266         }

```

```

266         else if(strcmp(FirstToken,"infinite") == 0 && strcmp(SecondToken,NULL) == 0 &&
strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
267             loaderinfinite();
268             printf("\n");
269         }
270
271         /*****
272         R5 comHand
273         *****/
274         // else if(strcmp(FirstToken,"heap") == 0 && strcmp(ThirdToken,NULL) == 0 &&
strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
275             // Init_Heap(atoi(SecondToken));
276             // }
277             // else if(strcmp(FirstToken,"alloc") == 0 && strcmp(ThirdToken,NULL) == 0 &&
strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
278                 // Alloc_Mem(atoi(SecondToken));
279                 // }
280                 // else if(strcmp(FirstToken,"free") == 0 && strcmp(ThirdToken,NULL) == 0 &&
strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
281                     // Free_Mem(atoi(SecondToken));
282                     // }
283                     // else if(strcmp(FirstToken,"empty") == 0 && strcmp(SecondToken,NULL) == 0 &&
strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
284                         // IsEmpty();
285                         // }
286                         else if(strcmp(FirstToken,"showFree") == 0 && strcmp(SecondToken,NULL) == 0 &&
strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
287                             ShowFree();
288                         }
289                         else if(strcmp(FirstToken,"showAlloc") == 0 && strcmp(SecondToken,NULL) == 0 &&
strcmp(ThirdToken,NULL) == 0 && strcmp(FourthToken,NULL) == 0 && strcmp(FifthToken,NULL) == 0) {
290                             ShowAlloc();
291                         }
292
293         /*****
294         shutdown comHand
295         *****/
296         else if(strcmp(FirstToken,"shutdown") == 0 && strcmp(SecondToken,NULL) == 0){
297             printf("\x1b[33m"\nAre you sure you want to shutdown? [yes/no]\n"\x1b[0m");
298             shutdown = 1;
299         }
300         else {
301             printf("\x1b[31m"\nERROR: Not a valid command \n"\x1b[0m");
302         }
303     }
304     else{
305         if(strcmp(FirstToken,"yes") == 0 && shutdown == 1) {
306             quit = 1;
307         }
308         else if(strcmp(FirstToken,"no") == 0){
309             printf("\x1b[33m"\nShutdown Cancelled\x1b[0m \n");
310             shutdown = 0;
311         }
312         else
313             printf("\x1b[31m"\nERROR: Please enter \"yes\" or \"no\" \n"\x1b[0m");
314     }
315     sys_req(IDLE, DEFAULT_DEVICE, NULL, NULL);
316 }
317 getReady() -> head = NULL;
318 sys_req(EXIT, DEFAULT_DEVICE, NULL, NULL);
319 return 0; //shutdown procedure
320 }

```

4.21 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)
- else if (**strcmp**(request,"showFree")==0)
- else if (**strcmp**(request,"showAlloc")==0)
- 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

- **else**
- **List** AlarmList

4.21.1 Function Documentation

4.21.1.1 BCDtoDec()

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

Description: Changes binary number to decimal numbers.

Parameters

<i>value</i>	Binary number to be changed to decimal
--------------	--

Definition at line 81 of file userFunctions.c.

```
81      {
82      return ((BCD>>4)*10) + (BCD & 0xF);
83 }
```

4.21.1.2 Block()

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

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

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process.
---------------------	---

Definition at line 957 of file userFunctions.c.

```
957      {
958      PCB* pcb = FindPCB(ProcessName);
959      if (pcb == NULL) {
960          printf("\x1b[31m"\nERROR: Not a valid process name \n"\x1b[0m");
961      }
962      else {
963          if(pcb->ReadyState == BLOCKED) {
964              printf("\x1b[32m"\nThis Process is already BLOCKED \n"\x1b[0m");
965          }
966          else {
967              RemovePCB(pcb);
968              pcb->ReadyState = BLOCKED;
969              InsertPCB(pcb);
970          }
971      }
972 }
```

4.21.1.3 Create_PCB()

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

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

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

Parameters

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

Definition at line 900 of file userFunctions.c.

```

900                                     {
901     if (FindPCB(ProcessName) == NULL) {
902         if(Priority >= 0 && Priority < 10){
903             if(Class == 0 || Class == 1){
904                 PCB* pcb = SetupPCB(ProcessName, Class, Priority);
905                 InsertPCB(pcb);
906             } else{
907                 printf("\x1b[31m"\nERROR: Not a valid Class \n"\x1b[0m");
908             }
909         } else{
910             printf("\x1b[31m"\nERROR: Not a valid Priority \n"\x1b[0m");
911         }
912     } else{
913         printf("\x1b[31m"\nERROR: This Process Name already exists \n"\x1b[0m");
914     }
915 }
```

4.21.1.4 DectoBCD()

```

int DectoBCD (
    int Decimal )
```

Description: Changes decimal numbers to binary numbers.

Parameters

<i>Decimal</i>	Decimal number to be changed to binary
----------------	--

Definition at line 88 of file userFunctions.c.

```

88                                     {
89     return (((Decimal/10) << 4) | (Decimal % 10));
90 }
```

4.21.1.5 Delete_PCB()

```

void Delete_PCB (
    char * ProcessName )
```

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

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process.
---------------------	---

Definition at line 926 of file userFunctions.c.

```

926     {
927     PCB* pcb = FindPCB(ProcessName);
928     if (pcb == NULL) {
929         printf("\x1b[31m"\nERROR: Not a valid process name \n"\x1b[0m");
930     }
931     else if(strcmp(pcb->Process_Name,"InfProc") == 0) {
932         if(pcb->SuspendedState == YES) {
933             RemovePCB(pcb);
934             FreePCB(pcb);
935         }
936         else
937             printf("\x1b[31m"\nERROR:This process cannot be deleted unless it is in the suspended
state\n"\x1b[0m");
938     }
939     else if(pcb -> Process_Class == SYSTEM) {
940         printf("\x1b[31m"\nERROR: System Processes cannot be deleted from the system. \n"\x1b[0m");
941     }
942     else {
943         RemovePCB(pcb);
944         FreePCB(pcb);
945     }
946 }
```

4.21.1.6 EdgeCase()

```

int EdgeCase (
    char * pointer )
```

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

Parameters

<i>Compares</i>	pointer char to validate if it is a number or not.
-----------------	--

Definition at line 109 of file userFunctions.c.

```

109     {
110     int valid = 0;
111     if (strcmp(pointer, "00") == 0) {
112         valid = 1;
113         return valid;
114     }
115     else if (strcmp(pointer, "0") == 0) {
116         valid = 1;
117         return valid;
118     }
119     else {
120         int j;
121         valid = 0;
122         for(j = 0; j <= 99; j++) {
123             if(strcmp(pointer, itoa(j)) == 0)
124                 valid = 1;
125         }
126         if(valid == 0) {
127             return valid;
128         }
129     }
130     return valid;
131 }
```

4.21.1.7 GetDate()

```

void GetDate ( )
```

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

No parameters.

Definition at line 271 of file userFunctions.c.

```

271     {
272
273     outb(0x70,0x07);
274     unsigned char day = BCDtoDec(inb(0x71));
275     outb(0x70,0x08);
276     unsigned char month = BCDtoDec(inb(0x71));
277     outb(0x70,0x32);
278     unsigned char millennium = BCDtoDec(inb(0x71));
279     char msg[2] = "-";
280     char msg3[10] = "Date: ";
281     printf(msg3);
282
283     printf(itoa(day));
284     //sys_req(WRITE, COM1, itoa(day), &check);
285     printf(msg);
286     printf(itoa(month));
287     //sys_req(WRITE, COM1, itoa(month), &check);
288     printf(msg);
289     printf(itoa(millennium));
290     //sys_req(WRITE, COM1, itoa(millennium), &check);
291     outb(0x70,0x09);
292     if(BCDtoDec(inb(0x71)) == 0){
293         printf("00");
294         //sys_req(WRITE, COM1, "00", &check);
295     }
296     else {
297         unsigned char year = BCDtoDec(inb(0x71));
298         printf(itoa(year));
299         //sys_req(WRITE, COM1, itoa(year), &check);
300     }
301     printf("\n");
302 }

```

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 190 of file userFunctions.c.

```

190     {
191
192     int hour;
193     int minute;
194     int second;
195     outb(0x70,0x04);
196     unsigned char hours = inb(0x71);
197     outb(0x70,0x02);
198     unsigned char minutes = inb(0x71);
199     outb(0x70,0x00);
200     unsigned char seconds = inb(0x71);
201     char msg1[2] = ":";
202     char msg2[10] = "Time: ";
203     printf(msg2);
204     hour = BCDtoDec(hours);
205     printf(itoa(hour));
206     //sys_req(WRITE, COM1, itoa(hour), &check);
207     printf(msg1);
208     minute = BCDtoDec(minutes);
209     printf(itoa(minute));
210     //sys_req(WRITE, COM1, itoa(minute), &check);
211     printf(msg1);
212     second = BCDtoDec(seconds);
213     printf(itoa(second));
214     //sys_req(WRITE, COM1, itoa(second), &check);
215     printf("\n");
216 }

```


4.21.1.9 Help()

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

Brief Description: Gives helpful information for one of the functions

Description: Can except a string as a pointer, if the pointer is null then the function will print a complete list of available commands to the console. If the pointer is a available 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

<i>request</i>	Character pointer that matches the name of the function that you need help with.
----------------	--

Definition at line 331 of file userFunctions.c.

```
331     {
332     if (request[0] == '\0') {
333         //removed for R3/R4 from active command list
334         //\n createPCB \n block \n unblock
335         //\n heap      alloc \n free      empty
336         printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
337         printf("\n getDate      setDate \n getTime      setTime \n version      suspend \n resume
setPriority \n showPCB      showAll \n showReady      showBlocked \n deletePCB      shutdown \n alarm
clear \n loadr3      infinte \n showFree      showAlloc \n\n");
338     }
339     else if (strcmp(request, "GetDate") == 0) {
340         printf("\n getDate returns the current date that is loaded onto the operating system.\n");
341     }
342     else if (strcmp(request, "SetDate") == 0) {
343         printf("\n setDate allows the user to reset the correct date into the system, as follows
setDate-\"BLU\"hour\"RESET\"-\"BLU\"month\"RESET\"-\"BLU\"year\"RESET\".\n Time must be inputed as a two digit
number, Example 02 or 00");
344     }
345     else if (strcmp(request, "GetTime") == 0) {
346         printf("\n getTime returns the current time as hours, minutes, seconds that is loaded onto the
operating system.\n");
347     }
348     else if (strcmp(request, "SetTime") == 0) {
349         printf("\n setTime allows the user to reset the correct time into the system, as follows
setTime-\"BLU\"hour\"RESET\"-\"BLU\"minute\"RESET\"-\"BLU\"second\"RESET\".\n Time must be inputed as a two digit
number, Example 02 or 00");
350     }
351     else if (strcmp(request, "Version") == 0) {
352         printf("\n version returns the current operating software version that the system is
running.\n");
353     }
354     else if (strcmp(request, "infinte") == 0) {
355         printf("\n infinite Loads the infinite process into the ready queue.\n");
356     }
357     else if (strcmp(request, "loadr3") == 0) {
358         printf("\n loadr3 Loads in all five of the R3 test processes in a suspended state into the
queue.\n");
359     }
360     else if (strcmp(request, "alarm") == 0) {
361         printf("\n alarm creates a user specified alarm with a user set message and time
alarm-MSG-hour-minute-second.\n");
362     }
363     else if (strcmp(request, "clear") == 0) {
364         printf("\n clear erases the console of all typed commands and refreshes it with just the command
list.\n");
365     }
366
367     else if (strcmp(request, "shutdown") == 0) {
368         printf("\n shutdown shuts down the system.\n");
369     }
370
371
372
373     /*****
374         R2 Commands
375         *****/
376     else if (strcmp(request, "suspend") == 0) {
377         printf("\n Suspend takes in the name of a PCB (suspend-NAME) then places it into the suspended
state and reinserts it into the correct queue.\n");
```

```

377 }
378 else if(strcmp(request,"resume") == 0) {
379     printf("\n Resume takes in the name of a PCB (resume-NAME) then removes it from the suspended
        state and adds it to the correct queue.\n");
380 }
381 else if(strcmp(request,"setPriority") == 0) {
382     printf("\n SetPriority takes in the name of a PCB and the priority (setPriority-NAME-PRIORITY)
        it needs to be set to then reinstates the specified PCB into a new location by priority.\n");
383 }
384 else if(strcmp(request,"showPCB") == 0) {
385     printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
        user.\n");
386 }
387 else if(strcmp(request,"showAll") == 0) {
388     printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
        queues.\n");
389 }
390 else if(strcmp(request,"showReady") == 0) {
391     printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
        currently are in the ready state.\n");
392 }
393 else if(strcmp(request,"showBlocked") == 0) {
394     printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
        currently are in the blocked state.\n");
395 }
396 }
397 /***** R2 Temp Commands *****/
398 else if(strcmp(request,"deletePCB") == 0) {
399     printf("\n DeletePCB takes in the process_name (deletePCB-NAME) then deletes it from the queue
        and free's all the memory that was previously allocated to the specified PCB.\n");
400 }
401 //removed for R3/R4 from active command list
402 /*
403 else if(strcmp(request,"createPCB") == 0) {
404     printf("\n CreatePCB takes in the process_name, process_class, and
        process_priority.(createPCB-NAME-PRIORITY-CLASS) Then assigns this new process into the correct
        queue.\n");
405 }
406 else if(strcmp(request,"block") == 0) {
407     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");
408 }
409 else if(strcmp(request,"unblock") == 0) {
410     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");
411 }
412 */
413
414
415 /***** R5 Temp Commands *****/
416 // else if(strcmp(request,"heap") == 0) {
417 //     printf("\n heap initializes the memory heap for the entire system.\n");
418 // }
419 // else if(strcmp(request,"alloc") == 0) {
420 //     printf("\n alloc allocates the specified amount of memory to the specific process
        (alloc-process_name-size).\n");
421 // }
422 // else if(strcmp(request,"free") == 0) {
423 //     printf("\n free frees the specified memory at the address given (free-address).\n");
424 // }
425 // else if(strcmp(request,"empty") == 0) {
426 //     printf("\n isempty returns true or false depending on if the heap has allocated memory.\n");
427 // }

```

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](http://www.geeksforgeeks.org).

Parameters

<i>num</i>	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 50 of file userFunctions.c.

```

50         {
51
52     int i,j,k,count;
53     i = num;
54     j = 0;
55     count = 0;
56     while(i){ // count number of digits
57         count++;
58         i /= 10;
59     }
60
61     char* arr1;
62     char arr2[count];
63     arr1 = (char*)sys_alloc_mem(count); //memory allocation
64
65     while(num){ // seperate last digit from number and add ASCII
66         arr2[++j] = num%10 + '0';
67         num /= 10;
68     }
69
70     for(k = 0; k < j; k++){ // reverse array results
71         arr1[k] = arr2[j-k];
72     }
73     arr1[k] = '\0';
74
75     return(char*)arr1;
76 }
```

4.21.1.11 Resume()

```

void Resume (
    char * ProcessName )
```

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

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process.
---------------------	---

Definition at line 483 of file userFunctions.c.

```

483         {
484     PCB* pcb = FindPCB(ProcessName);
485     if (pcb == NULL) {
486         printf(RED"\nERROR: Not a valid process name \n"RESET);
487     }
488     else {
489         if(pcb->SuspendedState == NO) {
490             printf(GRN"\nThis Process is already in the NONSUSPENDED state \n"RESET);
491         }
492         else if (pcb -> Process_Class == APPLICATION) {
493             pcb->SuspendedState = NO;
494         }
495         else
496             printf("\x1b[31m"\nERROR: Cannot Alter System Process \n"\x1b[0m");
497     }
498 }
```

4.21.1.12 Set_Priority()

```
void Set_Priority (
    char * ProcessName,
    int Priority )
```

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

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process.
<i>Priority</i>	integer that matches the priority number.

Definition at line 510 of file userFunctions.c.

```
510                                     {
511     PCB* pcb = FindPCB(ProcessName);
512     if (pcb == NULL) {
513         printf("\x1b[31m"\nERROR: Not a valid process name \n"\x1b[0m");
514     }
515     else if (Priority >= 10) {
516         printf("\x1b[31m"\nERROR: Not a valid Priority \n"\x1b[0m");
517     }
518     else if (pcb -> Process_Class == APPLICATION) {
519         RemovePCB(pcb);
520         pcb->Priority = Priority;
521         InsertPCB(pcb);
522     }
523     else
524         printf("\x1b[31m"\nERROR: Cannot Alter System Process \n"\x1b[0m");
525 }
```

4.21.1.13 SetDate()

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

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

Parameters

<i>day</i>	Integer to be set in the Day position
<i>month</i>	Integer to be set in the Month position
<i>millenial</i>	Integer to be set in the Millenial position
<i>year</i>	Integer to be set in the Year position

Definition at line 224 of file userFunctions.c.

```
224                                     {
225     outb(0x70, 0x07);
```

```

226 int tempDay = BCDtoDec(inb(0x71));
227 outb(0x70,0x08);
228 int tempMonth = BCDtoDec(inb(0x71));
229 outb(0x70,0x32);
230 int tempMillennium = BCDtoDec(inb(0x71));
231 outb(0x70,0x09);
232 int tempYear = BCDtoDec(inb(0x71));
233 cli();
234 outb(0x70,0x07);
235 outb(0x71,DectoBCD (day));
236 outb(0x70,0x08);
237 outb(0x71,DectoBCD (month));
238 outb(0x70,0x32);
239 outb(0x71,DectoBCD (millennium));
240 outb(0x70,0x09);
241 outb(0x71,DectoBCD (year));
242 sti();
243 outb(0x70,0x07);
244 unsigned char newDay = BCDtoDec(inb(0x71));
245 outb(0x70,0x08);
246 unsigned char newMonth = BCDtoDec(inb(0x71));
247 outb(0x70,0x32);
248 unsigned char newMillennium = BCDtoDec(inb(0x71));
249 outb(0x70,0x09);
250 unsigned char newYear = BCDtoDec(inb(0x71));
251 if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
252     printf("Your input was invalid\n");
253     cli();
254     outb(0x70,0x07);
255     outb(0x71,DectoBCD (tempDay));
256     outb(0x70,0x08);
257     outb(0x71,DectoBCD (tempMonth));
258     outb(0x70,0x32);
259     outb(0x71,DectoBCD (tempMillennium));
260     outb(0x70,0x09);
261     outb(0x71,DectoBCD (tempYear));
262     sti();
263 }
264 else
265     printf("Date Set\n");
266 }

```

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

<i>hours</i>	Integer to be set in the Hour position
<i>minutes</i>	Integer to be set in the Minutes position
<i>seconds</i>	Integer to be set in the Seconds position

Definition at line 151 of file userFunctions.c.

```

151 {
152     outb(0x70,0x04);
153     unsigned char tempHours = BCDtoDec(inb(0x71));
154     outb(0x70,0x02);
155     unsigned char tempMinutes = BCDtoDec(inb(0x71));
156     outb(0x70,0x00);
157     unsigned char tempSeconds = BCDtoDec(inb(0x71));
158     cli(); //outb(device + 1, 0x00); //disable interrupts
159     outb(0x70,0x04);
160     outb(0x71, DectoBCD(hours)); // change to bcd
161     outb(0x70,0x02);

```

```

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

```

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, class, state, suspend status, and priority of each of the [PCB](#)'s in the ready and blocked queues.

Definition at line 610 of file userFunctions.c.

```

610     {
611         Show_Ready();
612         Show_Blocked();
613     }

```

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, class, state, suspend status, and priority of each of the [PCB](#)'s in the blocked queue.

Definition at line 759 of file userFunctions.c.

```

759     {
760         if(getBlocked()->head == NULL) {
761             printf("\x1b[32m\n The Blocked Queue is empty \n"\x1b[0m");
762         }
763         else {
764             int class, state, prior, status;
765             char name[20];
766             char block[] = "\x1b[34m"Blocked Queue: \n"\x1b[0m";
767             char cname[] = "Name: ";
768             char cclass[] = "Class: ";
769             char cstate[] = "State: ";
770             char cstatus[] = "Status: ";
771             char cprior[] = "Priority: ";
772             char line[] = "\n";

```

```

773
774     printf(block);
775     //sys_req(WRITE, COM1, block, &check );
776
777     PCB* pcb = getBlocked()->head;
778
779     if(pcb->next == NULL) {
780         class = pcb->Process_Class;
781         strcpy(name,pcb->Process_Name);
782         state = pcb->ReadyState;
783         status = pcb->SuspendedState;
784         prior = pcb->Priority;
785
786         printf(cname);
787         printf(name);
788         printf(line);
789
790         printf(cclass);
791         if(pcb->Process_Class == 0) {
792             printf("0");
793         }
794         else {
795             printf(itoa(class));
796             //sys_req(WRITE, COM1, itoa(class), &check);
797         }
798         printf(line);
799
800         printf(cstate);
801         if(pcb->ReadyState == 0) {
802             printf("0");
803         }
804         else {
805             printf(itoa(state));
806             //sys_req(WRITE, COM1, itoa(state), &check);
807         }
808         printf(line);
809
810         printf(cstatus);
811         if(pcb->SuspendedState == 0) {
812             printf("0");
813         }
814         else {
815             printf(itoa(status));
816             //sys_req(WRITE, COM1, itoa(status), &check);
817         }
818         printf(line);
819
820         printf(cprior);
821         if(pcb->Priority == 0) {
822             printf("0");
823             printf("\n\n");
824         }
825         else {
826             printf(itoa(prior));
827             //sys_req(WRITE, COM1, itoa(prior), &check);
828             printf("\n\n");
829         }
830     }
831     else {
832         while(pcb != NULL) {
833             class = pcb->Process_Class;
834             strcpy(name,pcb->Process_Name);
835             state = pcb->ReadyState;
836             status = pcb->SuspendedState;
837             prior = pcb->Priority;
838
839             printf(cname);
840             printf(name);
841             printf(line);
842
843             printf(cclass);
844             if(pcb->Process_Class == 0) {
845                 printf("0");
846             }
847             else {
848                 printf(itoa(class));
849                 //sys_req(WRITE, COM1, itoa(class), &check);
850             }
851             printf(line);
852
853             printf(cstate);
854             if(pcb->ReadyState == 0) {
855                 printf("0");
856             }
857             else {
858                 printf(itoa(state));
859                 //sys_req(WRITE, COM1, itoa(state), &check);

```

```

860         }
861         printf(line);
862
863         printf(cstatus);
864         if(pcb->SuspendedState == 0) {
865             printf("0");
866         }
867         else {
868             printf(itoa(status));
869             //sys_req(WRITE, COM1, itoa(status), &check);
870         }
871         printf(line);
872
873         printf(cprior);
874         if(pcb->Priority == 0) {
875             printf("0");
876             printf("\n\n");
877         }
878         else {
879             printf(itoa(prior));
880             //sys_req(WRITE, COM1, itoa(prior), &check);
881             printf("\n\n");
882         }
883         pcb = pcb->next;
884     }
885 }
886 }
887 }

```

4.21.1.17 Show_PCB()

```

void Show_PCB (
    char * ProcessName )

```

Brief Description: Displays the process name, class, state, suspended status, and priority of a [PCB](#).

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process
---------------------	--

Definition at line 535 of file userFunctions.c.

```

535     {
536         if (FindPCB(ProcessName) == NULL) {
537             printf("\x1b[31m""\nERROR: PCB does not exist \n""\x1b[0m");
538         }
539         else {
540
541             char name[10];
542             char cname[] = "Name: ";
543             char cclass[] = "Class: ";
544             char cstate[] = "State: ";
545             char cstatus[] = "Status: ";
546             char cprior[] = "Priority: ";
547             char line[] = "\n";
548             PCB* pcb = FindPCB(ProcessName);
549             strcpy(name,pcb->Process_Name);
550             int class = pcb->Process_Class;
551             int state = pcb->ReadyState;
552             int status = pcb->SuspendedState;
553             int prior = pcb->Priority;
554
555             if(name == NULL){
556                 printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
557             }
558             else {
559                 printf(cname);
560                 printf(ProcessName);
561                 printf(line);

```



```

562         printf(cclass);
563         if(pcb->Process_Class == 0) {
564             printf("0");
565         }
566         else {
567             printf(itoa(class));
568             //sys_req(WRITE, COM1, itoa(class), &check);
569         }
570         printf(line);
571         printf(cstate);
572         if(pcb->ReadyState == 0) {
573             printf("0");
574         }
575         else {
576             printf(itoa(state));
577             //sys_req(WRITE, COM1, itoa(state), &check);
578         }
579         printf(line);
580         printf(cstatus);
581         if(pcb->SuspendedState == 0) {
582             printf("0");
583         }
584         else {
585             printf(itoa(status));
586             //sys_req(WRITE, COM1, itoa(status), &check);
587         }
588         printf(line);
589         printf(cprior);
590         if(pcb->Priority == 0) {
591             printf("0");
592             printf("\n\n");
593         }
594         else {
595             printf(itoa(prior));
596             //sys_req(WRITE, COM1, itoa(prior), &check);
597             printf("\n\n");
598         }
599     }
600 }
601 }

```

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, class, state, suspend status, and priority of each of the [PCB](#)'s in the ready queue.

Definition at line 622 of file userFunctions.c.

```

622     {
623         if(getReady()->head == NULL) {
624             printf("\x1b[32m"\n The Ready Queue is empty \n""\x1b[0m");
625         }
626         else {
627             int class, state, prior, status;
628             char name[10];
629             char ready[] = "\x1b[34m"\nReady Queue:\n""\x1b[0m";
630             char cname[] = "Name: ";
631             char cclass[] = "Class: ";
632             char cstate[] = "State: ";
633             char cstatus[] = "Status: ";
634             char cprior[] = "Priority: ";
635             char line[] = "\n";
636
637             printf(ready);
638             //sys_req(WRITE, COM1, ready, &check );
639
640             PCB* pcb = getReady()->head;
641
642             if(pcb->next == NULL) {
643                 class = pcb->Process_Class;
644                 strcpy(name,pcb->Process_Name);
645                 state = pcb->ReadyState;
646                 status = pcb->SuspendedState;

```

```

647         prior = pcb->Priority;
648
649         printf(cname);
650         printf(name);
651         printf(line);
652
653         printf(cclass);
654         if(pcb->Process_Class == 0) {
655             printf("0");
656         }
657         else {
658             printf(itoa(class));
659             //sys_req(WRITE, COM1, itoa(class), &check);
660         }
661         printf(line);
662
663         printf(cstate);
664         if(pcb->ReadyState == 0) {
665             printf("0");
666         }
667         else {
668             printf(itoa(state));
669             //sys_req(WRITE, COM1, itoa(state), &check);
670         }
671         printf(line);
672
673         printf(cstatus);
674         if(pcb->SuspendedState == 0) {
675             printf("0");
676         }
677         else {
678             printf(itoa(status));
679             //sys_req(WRITE, COM1, itoa(status), &check);
680         }
681         printf(line);
682
683         printf(cprior);
684         if(pcb->Priority == 0) {
685             printf("0");
686             printf("\n\n");
687         }
688         else {
689             printf(itoa(prior));
690             //sys_req(WRITE, COM1, itoa(prior), &check);
691             printf("\n\n");
692         }
693     }
694     else {
695         while(pcb != NULL) {
696             class = pcb->Process_Class;
697             strcpy(name,pcb->Process_Name);
698             state = pcb->ReadyState;
699             status = pcb->SuspendedState;
700             prior = pcb->Priority;
701
702             printf(cname);
703             printf(name);
704             printf(line);
705
706             printf(cclass);
707             if(pcb->Process_Class == 0) {
708                 printf("0");
709             }
710             else {
711                 printf(itoa(class));
712                 //sys_req(WRITE, COM1, itoa(class), &check);
713             }
714             printf(line);
715
716             printf(cstate);
717             if(pcb->ReadyState == 0) {
718                 printf("0");
719             }
720             else {
721                 printf(itoa(state));
722                 //sys_req(WRITE, COM1, itoa(state), &check);
723             }
724             printf(line);
725
726             printf(cstatus);
727             if(pcb->SuspendedState == 0) {
728                 printf("0");
729             }
730             else {
731                 printf(itoa(status));
732                 //sys_req(WRITE, COM1, itoa(status), &check);
733             }

```

```

734         printf(line);
735
736         printf(cprior);
737         if(pcb->Priority == 0) {
738             printf("0");
739             printf("\n\n");
740         }
741         else {
742             printf(itoa(prior));
743             //sys_req(WRITE, COM1, itoa(prior), &check);
744             printf("\n\n");
745         }
746         pcb = pcb->next;
747     }
748 }
749 }
750 }

```

4.21.1.19 Suspend()

```

void Suspend (
    char * ProcessName )

```

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

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process.
---------------------	---

Definition at line 457 of file userFunctions.c.

```

457     {
458         PCB* pcb = FindPCB(ProcessName);
459         if (pcb == NULL) {
460             printf("\x1b[31m"\nERROR: Not a valid process name \n"\x1b[0m");
461         }
462         else {
463             if(pcb->SuspendedState == YES) {
464                 printf("\x1b[32m"\nThis Process is already SUSPENDED \n"\x1b[0m");
465             }
466             else if (pcb -> Process_Class == APPLICATION) {
467                 pcb->SuspendedState = YES;
468             }
469             else
470                 printf("\x1b[31m"\nERROR: Cannot Alter System Process \n"\x1b[0m");
471         }
472     }

```

4.21.1.20 toLowercase()

```

char toLowercase (
    char c )

```

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

Parameters

<i>c</i>	Character that is to be changed to its lowercase equivalent
----------	---

Definition at line 314 of file userFunctions.c.

```

314         {
315     if((c >= 65) && (c <= 90)) {
316         c = c + 32;
317     }
318     return c;
319 }
```

4.21.1.21 Unblock()

```

void Unblock (
    char * ProcessName )
```

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

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process.
---------------------	---

Definition at line 983 of file userFunctions.c.

```

983     {
984     PCB* pcb = FindPCB(ProcessName);
985     if (pcb == NULL) {
986         printf("\x1b[31m"\nERROR: Not a valid process name \n"\x1b[0m");
987     }
988     else {
989         if(pcb->ReadyState == READY) {
990             printf("\x1b[32m"\nThis Process is already in the READY state \n"\x1b[0m");
991         }
992         else {
993             RemovePCB(pcb);
994             pcb->ReadyState = READY;
995             InsertPCB(pcb);
996         }
997     }
998 }
```

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 307 of file userFunctions.c.

```

307     {
308     printf("Version: R5.2 \n");
309 }
```

4.21.2 Variable Documentation

4.21.2.1 AlarmList

[List](#) AlarmList

Initial value:

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

Definition at line 1045 of file userFunctions.c.

4.21.2.2 else

else

Initial value:

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

Definition at line 437 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 (
    int BCD )
```

Description: Changes binary number to decimal numbers.

Parameters

<i>value</i>	Binary number to be changed to decimal
--------------	--

Definition at line 81 of file userFunctions.c.

```
81     {
82         return (((BCD>>4)*10) + (BCD & 0xF));
```

```
83 }
```

4.22.1.2 Block()

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

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

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process.
---------------------	---

Definition at line 957 of file userFunctions.c.

```
957     {
958     PCB* pcb = FindPCB(ProcessName);
959     if (pcb == NULL) {
960         printf("\x1b[31m"\nERROR: Not a valid process name \n"\x1b[0m");
961     }
962     else {
963         if(pcb->ReadyState == BLOCKED) {
964             printf("\x1b[32m"\nThis Process is already BLOCKED \n"\x1b[0m");
965         }
966         else {
967             RemovePCB(pcb);
968             pcb->ReadyState = BLOCKED;
969             InsertPCB(pcb);
970         }
971     }
972 }
```

4.22.1.3 Create_PCB()

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

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

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

Parameters

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

Definition at line 900 of file userFunctions.c.

```

900                                     {
901     if (FindPCB(ProcessName) == NULL) {
902         if(Priority >= 0 && Priority < 10){
903             if(Class == 0 || Class == 1){
904                 PCB* pcb = SetupPCB(ProcessName, Class, Priority);
905                 InsertPCB(pcb);
906             } else{
907                 printf("\x1b[31m"\nERROR: Not a valid Class \n""\x1b[0m");
908             }
909         } else{
910             printf("\x1b[31m"\nERROR: Not a valid Priority \n""\x1b[0m");
911         }
912     } else{
913         printf("\x1b[31m"\nERROR: This Process Name already exists \n""\x1b[0m");
914     }
915 }
```

4.22.1.4 DectoBCD()

```

int DectoBCD (
    int Decimal )
```

Description: Changes decimal numbers to binary numbers.

Parameters

<i>Decimal</i>	Decimal number to be changed to binary
----------------	--

Definition at line 88 of file userFunctions.c.

```

88                                     {
89     return (((Decimal/10) << 4) | (Decimal % 10));
90 }
```

4.22.1.5 Delete_PCB()

```

void Delete_PCB (
    char * ProcessName )
```

Brief Description: Removes [PCB](#) from appropriate queue and frees all associated memory.

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process.
---------------------	---

Definition at line 926 of file userFunctions.c.

```

926                                     {
927     PCB* pcb = FindPCB(ProcessName);
928     if (pcb == NULL) {
929         printf("\x1b[31m"\nERROR: Not a valid process name \n""\x1b[0m");
930     }
931     else if(strcmp(pcb->Process_Name,"InfProc") == 0) {
932         if(pcb->SuspendedState == YES) {
```



```

933         RemovePCB(pcb);
934         FreePCB(pcb);
935     }
936     else
937     {
938         printf("\x1b[31m"\nERROR:This process cannot be deleted unless it is in the suspended
state\n""\x1b[0m");
939     }
940     else if (pcb->Process_Class == SYSTEM) {
941         printf("\x1b[31m"\nERROR: System Processes cannot be deleted from the system. \n""\x1b[0m");
942     }
943     else {
944         RemovePCB(pcb);
945         FreePCB(pcb);
946     }
947 }

```

4.22.1.6 EdgeCase()

```

int EdgeCase (
    char * pointer )

```

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

Parameters

<i>Compares</i>	pointer char to validate if it is a number or not.
-----------------	--

Definition at line 109 of file userFunctions.c.

```

109     {
110     int valid = 0;
111     if (strcmp(pointer, "00") == 0) {
112         valid = 1;
113         return valid;
114     }
115     else if (strcmp(pointer, "0") == 0) {
116         valid = 1;
117         return valid;
118     }
119     else {
120         int j;
121         valid = 0;
122         for(j = 0; j <= 99; j++) {
123             if (strcmp(pointer, itoa(j)) == 0)
124                 valid = 1;
125         }
126         if (valid == 0) {
127             return valid;
128         }
129     }
130     return valid;
131 }

```

4.22.1.7 GetDate()

```

void GetDate ( )

```

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

No parameters.

Definition at line 271 of file userFunctions.c.

```

271     {

```

```

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

```

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 190 of file userFunctions.c.

```

190         {
191
192         int hour;
193         int minute;
194         int second;
195         outb(0x70,0x04);
196         unsigned char hours = inb(0x71);
197         outb(0x70,0x02);
198         unsigned char minutes = inb(0x71);
199         outb(0x70,0x00);
200         unsigned char seconds = inb(0x71);
201         char msg1[2] = ":";
202         char msg2[10] = "Time: ";
203         printf(msg2);
204         hour = BCDtoDec(hours);
205         printf(itoa(hour));
206         //sys_req(WRITE, COM1, itoa(hour), &check);
207         printf(msg1);
208         minute = BCDtoDec(minutes);
209         printf(itoa(minute));
210         //sys_req(WRITE, COM1, itoa(minute), &check);
211         printf(msg1);
212         second = BCDtoDec(seconds);
213         printf(itoa(second));
214         //sys_req(WRITE, COM1, itoa(second), &check);
215         printf("\n");
216 }

```

4.22.1.9 Help()

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

Brief Description: Gives helpful information for one of the functions

Description: Can except a string as a pointer, if the pointer is null then the function will print a complete list of available commands to the console. If the pointer is a available 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

<i>request</i>	Character pointer that matches the name of the function that you need help with.
----------------	--

Definition at line 331 of file userFunctions.c.

```
331     {
332     if (request[0] == '\0') {
333         //removed for R3/R4 from active command list
334         //\n createPCB \n block \n unblock
335         //\n heap      alloc \n free      empty
336         printf("\n to chain commands and parameters, please use \"-\" between keywords \n");
337         printf("\n getDate      setDate \n getTime      setTime \n version      suspend \n resume
setPriority \n showPCB      showAll \n showReady      showBlocked \n deletePCB      shutdown \n alarm
clear \n loadr3      infinte \n showFree      showAlloc \n\n");
338     }
339     else if (strcmp(request, "GetDate") == 0) {
340         printf("\n getDate returns the current date that is loaded onto the operating system.\n");
341     }
342     else if (strcmp(request, "SetDate") == 0) {
343         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");
344     }
345     else if (strcmp(request, "GetTime") == 0) {
346         printf("\n getTime returns the current time as hours, minutes, seconds that is loaded onto the
operating system.\n");
347     }
348     else if (strcmp(request, "SetTime") == 0) {
349         printf("\n setTime allows the user to reset the correct time into the system, as follows
setTime-\"BLU\"hour\"RESET\"-\"BLU\"minute\"RESET\"-\"BLU\"second\"RESET\".\n Time must be inputed as a two digit
number, Example 02 or 00");
350     }
351     else if (strcmp(request, "Version") == 0) {
352         printf("\n version returns the current operating software version that the system is
running.\n");
353     }
354     else if (strcmp(request, "infinte") == 0) {
355         printf("\n infinite Loads the infinite process into the ready queue.\n");
356     }
357     else if (strcmp(request, "loadr3") == 0) {
358         printf("\n loadr3 Loads in all five of the R3 test processes in a suspended state into the
queue.\n");
359     }
360     else if (strcmp(request, "alarm") == 0) {
361         printf("\n alarm creates a user specified alarm with a user set message and time
alarm-MSG-hour-minute-second.\n");
362     }
363     else if (strcmp(request, "clear") == 0) {
364         printf("\n clear erases the console of all typed commands and refreshes it with just the command
list.\n");
365     }
366     else if (strcmp(request, "shutdown") == 0) {
367         printf("\n shutdown shuts down the system.\n");
368     }
369 }
370
371
372
373     /*****
374     R2 Commands
375     *****/
376     else if (strcmp(request, "suspend") == 0) {
377         printf("\n Suspend takes in the name of a PCB (suspend-NAME) then places it into the suspended
state and reinserts it into the correct queue.\n");
```

```

377 }
378 else if(strcmp(request,"resume") == 0) {
379     printf("\n Resume takes in the name of a PCB (resume-NAME) then removes it from the suspended
        state and adds it to the correct queue.\n");
380 }
381 else if(strcmp(request,"setPriority") == 0) {
382     printf("\n SetPriority takes in the name of a PCB and the priority (setPriority-NAME-PRIORITY)
        it needs to be set to then reinstates the specified PCB into a new location by priority.\n");
383 }
384 else if(strcmp(request,"showPCB") == 0) {
385     printf("\n ShowPCB takes in the name of a PCB and returns all the associated attributes to the
        user.\n");
386 }
387 else if(strcmp(request,"showAll") == 0) {
388     printf("\n ShowAll takes no parameters but returns all PCB's that are currently in any of the
        queues.\n");
389 }
390 else if(strcmp(request,"showReady") == 0) {
391     printf("\n ShowReady takes in no parameters but returns all PCB's and there attributes that
        currently are in the ready state.\n");
392 }
393 else if(strcmp(request,"showBlocked") == 0) {
394     printf("\n ShowBlocked takes in no parameters but returns all PCB's and there attributes that
        currently are in the blocked state.\n");
395 }
396 }
397 /***** R2 Temp Commands *****/
398 else if(strcmp(request,"deletePCB") == 0) {
399     printf("\n DeletePCB takes in the process_name (deletePCB-NAME) then deletes it from the queue
        and free's all the memory that was previously allocated to the specified PCB.\n");
400 }
401 //removed for R3/R4 from active command list
402 /*
403 else if(strcmp(request,"createPCB") == 0) {
404     printf("\n CreatePCB takes in the process_name, process_class, and
        process_priority.(createPCB-NAME-PRIORITY-CLASS) Then assigns this new process into the correct
        queue.\n");
405 }
406 else if(strcmp(request,"block") == 0) {
407     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");
408 }
409 else if(strcmp(request,"unblock") == 0) {
410     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");
411 }
412 */
413
414
415 /***** R5 Temp Commands *****/
416 // else if(strcmp(request,"heap") == 0) {
417 //     printf("\n heap initializes the memory heap for the entire system.\n");
418 // }
419 // else if(strcmp(request,"alloc") == 0) {
420 //     printf("\n alloc allocates the specified amount of memory to the specific process
        (alloc-process_name-size).\n");
421 // }
422 // else if(strcmp(request,"free") == 0) {
423 //     printf("\n free frees the specified memory at the address given (free-address).\n");
424 // }
425 // else if(strcmp(request,"empty") == 0) {
426 //     printf("\n isempty returns true or false depending on if the heap has allocated memory.\n");
427 // }

```

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](http://www.geeksforgeeks.org).

Parameters

<i>num</i>	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 50 of file userFunctions.c.

```

50         {
51
52     int i,j,k,count;
53     i = num;
54     j = 0;
55     count = 0;
56     while(i){ // count number of digits
57         count++;
58         i /= 10;
59     }
60
61     char* arr1;
62     char arr2[count];
63     arr1 = (char*)sys_alloc_mem(count); //memory allocation
64
65     while(num){ // seperate last digit from number and add ASCII
66         arr2[++j] = num%10 + '0';
67         num /= 10;
68     }
69
70     for(k = 0; k < j; k++){ // reverse array results
71         arr1[k] = arr2[j-k];
72     }
73     arr1[k] = '\0';
74
75     return(char*)arr1;
76 }
```

4.22.1.11 Resume()

```

void Resume (
    char * ProcessName )
```

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

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process.
---------------------	---

Definition at line 483 of file userFunctions.c.

```

483         {
484     PCB* pcb = FindPCB(ProcessName);
485     if (pcb == NULL) {
486         printf(RED"\nERROR: Not a valid process name \n"RESET);
487     }
488     else {
489         if(pcb->SuspendedState == NO) {
490             printf(GRN"\nThis Process is already in the NONSUSPENDED state \n"RESET);
491         }
492         else if (pcb -> Process_Class == APPLICATION) {
493             pcb->SuspendedState = NO;
494         }
495         else
496             printf("\x1b[31m"\nERROR: Cannot Alter System Process \n"\x1b[0m");
497     }
498 }
```

4.22.1.12 Set_Priority()

```
void Set_Priority (
    char * ProcessName,
    int Priority )
```

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

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process.
<i>Priority</i>	integer that matches the priority number.

Definition at line 510 of file userFunctions.c.

```
510                                     {
511     PCB* pcb = FindPCB(ProcessName);
512     if (pcb == NULL) {
513         printf("\x1b[31m"\nERROR: Not a valid process name \n"\x1b[0m");
514     }
515     else if(Priority >= 10){
516         printf("\x1b[31m"\nERROR: Not a valid Priority \n"\x1b[0m");
517     }
518     else if(pcb -> Process_Class == APPLICATION) {
519         RemovePCB(pcb);
520         pcb->Priority = Priority;
521         InsertPCB(pcb);
522     }
523     else
524         printf("\x1b[31m"\nERROR: Cannot Alter System Process \n"\x1b[0m");
525 }
```

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

<i>day</i>	Integer to be set in the Day position
<i>month</i>	Integer to be set in the Month position
<i>millenial</i>	Integer to be set in the Millenial position
<i>year</i>	Integer to be set in the Year position

Definition at line 224 of file userFunctions.c.

```
224                                     {
225     outb(0x70, 0x07);
```

```

226 int tempDay = BCDtoDec(inb(0x71));
227 outb(0x70,0x08);
228 int tempMonth = BCDtoDec(inb(0x71));
229 outb(0x70,0x32);
230 int tempMillennium = BCDtoDec(inb(0x71));
231 outb(0x70,0x09);
232 int tempYear = BCDtoDec(inb(0x71));
233 cli();
234 outb(0x70,0x07);
235 outb(0x71,DectoBCD (day));
236 outb(0x70,0x08);
237 outb(0x71,DectoBCD (month));
238 outb(0x70,0x32);
239 outb(0x71,DectoBCD (millennium));
240 outb(0x70,0x09);
241 outb(0x71,DectoBCD (year));
242 sti();
243 outb(0x70,0x07);
244 unsigned char newDay = BCDtoDec(inb(0x71));
245 outb(0x70,0x08);
246 unsigned char newMonth = BCDtoDec(inb(0x71));
247 outb(0x70,0x32);
248 unsigned char newMillennium = BCDtoDec(inb(0x71));
249 outb(0x70,0x09);
250 unsigned char newYear = BCDtoDec(inb(0x71));
251 if(newDay != day || newMonth != month || newMillennium != millennium || newYear != year){
252     printf("Your input was invalid\n");
253     cli();
254     outb(0x70,0x07);
255     outb(0x71,DectoBCD (tempDay));
256     outb(0x70,0x08);
257     outb(0x71,DectoBCD (tempMonth));
258     outb(0x70,0x32);
259     outb(0x71,DectoBCD (tempMillennium));
260     outb(0x70,0x09);
261     outb(0x71,DectoBCD (tempYear));
262     sti();
263 }
264 else
265     printf("Date Set\n");
266 }

```

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

<i>hours</i>	Integer to be set in the Hour position
<i>minutes</i>	Integer to be set in the Minutes position
<i>seconds</i>	Integer to be set in the Seconds position

Definition at line 151 of file userFunctions.c.

```

151 {
152     outb(0x70,0x04);
153     unsigned char tempHours = BCDtoDec(inb(0x71));
154     outb(0x70,0x02);
155     unsigned char tempMinutes = BCDtoDec(inb(0x71));
156     outb(0x70,0x00);
157     unsigned char tempSeconds = BCDtoDec(inb(0x71));
158     cli(); //outb(device + 1, 0x00); //disable interrupts
159     outb(0x70,0x04);
160     outb(0x71, DectoBCD(hours)); // change to bcd
161     outb(0x70,0x02);

```

```

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

```

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, class, state, suspend status, and priority of each of the [PCB](#)'s in the ready and blocked queues.

Definition at line 610 of file userFunctions.c.

```

610     {
611         Show_Ready();
612         Show_Blocked();
613     }

```

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, class, state, suspend status, and priority of each of the [PCB](#)'s in the blocked queue.

Definition at line 759 of file userFunctions.c.

```

759     {
760         if(getBlocked()->head == NULL) {
761             printf("\x1b[32m\n The Blocked Queue is empty \n"\x1b[0m");
762         }
763         else {
764             int class, state, prior, status;
765             char name[20];
766             char block[] = "\x1b[34m"Blocked Queue: \n"\x1b[0m";
767             char cname[] = "Name: ";
768             char cclass[] = "Class: ";
769             char cstate[] = "State: ";
770             char cstatus[] = "Status: ";
771             char cprior[] = "Priority: ";
772             char line[] = "\n";

```



```

773
774     printf(block);
775     //sys_req(WRITE, COM1, block, &check );
776
777     PCB* pcb = getBlocked()->head;
778
779     if(pcb->next == NULL) {
780         class = pcb->Process_Class;
781         strcpy(name,pcb->Process_Name);
782         state = pcb->ReadyState;
783         status = pcb->SuspendedState;
784         prior = pcb->Priority;
785
786         printf(cname);
787         printf(name);
788         printf(line);
789
790         printf(cclass);
791         if(pcb->Process_Class == 0) {
792             printf("0");
793         }
794         else {
795             printf(itoa(class));
796             //sys_req(WRITE, COM1, itoa(class), &check);
797         }
798         printf(line);
799
800         printf(cstate);
801         if(pcb->ReadyState == 0) {
802             printf("0");
803         }
804         else {
805             printf(itoa(state));
806             //sys_req(WRITE, COM1, itoa(state), &check);
807         }
808         printf(line);
809
810         printf(cstatus);
811         if(pcb->SuspendedState == 0) {
812             printf("0");
813         }
814         else {
815             printf(itoa(status));
816             //sys_req(WRITE, COM1, itoa(status), &check);
817         }
818         printf(line);
819
820         printf(cprior);
821         if(pcb->Priority == 0) {
822             printf("0");
823             printf("\n\n");
824         }
825         else {
826             printf(itoa(prior));
827             //sys_req(WRITE, COM1, itoa(prior), &check);
828             printf("\n\n");
829         }
830     }
831     else {
832         while(pcb != NULL) {
833             class = pcb->Process_Class;
834             strcpy(name,pcb->Process_Name);
835             state = pcb->ReadyState;
836             status = pcb->SuspendedState;
837             prior = pcb->Priority;
838
839             printf(cname);
840             printf(name);
841             printf(line);
842
843             printf(cclass);
844             if(pcb->Process_Class == 0) {
845                 printf("0");
846             }
847             else {
848                 printf(itoa(class));
849                 //sys_req(WRITE, COM1, itoa(class), &check);
850             }
851             printf(line);
852
853             printf(cstate);
854             if(pcb->ReadyState == 0) {
855                 printf("0");
856             }
857             else {
858                 printf(itoa(state));
859                 //sys_req(WRITE, COM1, itoa(state), &check);

```

```

860         }
861         printf(line);
862
863         printf(cstatus);
864         if(pcb->SuspendedState == 0) {
865             printf("0");
866         }
867         else {
868             printf(itoa(status));
869             //sys_req(WRITE, COM1, itoa(status), &check);
870         }
871         printf(line);
872
873         printf(cprior);
874         if(pcb->Priority == 0) {
875             printf("0");
876             printf("\n\n");
877         }
878         else {
879             printf(itoa(prior));
880             //sys_req(WRITE, COM1, itoa(prior), &check);
881             printf("\n\n");
882         }
883         pcb = pcb->next;
884     }
885 }
886 }
887 }

```

4.22.1.17 Show_PCB()

```

void Show_PCB (
    char * ProcessName )

```

Brief Description: Displays the process name, class, state, suspended status, and priority of a [PCB](#).

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process
---------------------	--

Definition at line 535 of file userFunctions.c.

```

535     {
536         if (FindPCB(ProcessName) == NULL) {
537             printf("\x1b[31m""\nERROR: PCB does not exist \n""\x1b[0m");
538         }
539         else {
540
541             char name[10];
542             char cname[] = "Name: ";
543             char cclass[] = "Class: ";
544             char cstate[] = "State: ";
545             char cstatus[] = "Status: ";
546             char cprior[] = "Priority: ";
547             char line[] = "\n";
548             PCB* pcb = FindPCB(ProcessName);
549             strcpy(name,pcb->Process_Name);
550             int class = pcb->Process_Class;
551             int state = pcb->ReadyState;
552             int status = pcb->SuspendedState;
553             int prior = pcb->Priority;
554
555             if(name == NULL){
556                 printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
557             }
558             else {
559                 printf(cname);
560                 printf(ProcessName);
561                 printf(line);

```

```

562         printf(cclass);
563         if(pcb->Process_Class == 0) {
564             printf("0");
565         }
566         else {
567             printf(itoa(class));
568             //sys_req(WRITE, COM1, itoa(class), &check);
569         }
570         printf(line);
571         printf(cstate);
572         if(pcb->ReadyState == 0) {
573             printf("0");
574         }
575         else {
576             printf(itoa(state));
577             //sys_req(WRITE, COM1, itoa(state), &check);
578         }
579         printf(line);
580         printf(cstatus);
581         if(pcb->SuspendedState == 0) {
582             printf("0");
583         }
584         else {
585             printf(itoa(status));
586             //sys_req(WRITE, COM1, itoa(status), &check);
587         }
588         printf(line);
589         printf(cprior);
590         if(pcb->Priority == 0) {
591             printf("0");
592             printf("\n\n");
593         }
594         else {
595             printf(itoa(prior));
596             //sys_req(WRITE, COM1, itoa(prior), &check);
597             printf("\n\n");
598         }
599     }
600 }
601 }

```

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, class, state, suspend status, and priority of each of the [PCB](#)'s in the ready queue.

Definition at line 622 of file userFunctions.c.

```

622     {
623         if(getReady()->head == NULL) {
624             printf("\x1b[32m"\n The Ready Queue is empty \n"\x1b[0m");
625         }
626         else {
627             int class, state, prior, status;
628             char name[10];
629             char ready[] = "\x1b[34m"\nReady Queue:\n"\x1b[0m";
630             char cname[] = "Name: ";
631             char cclass[] = "Class: ";
632             char cstate[] = "State: ";
633             char cstatus[] = "Status: ";
634             char cprior[] = "Priority: ";
635             char line[] = "\n";
636
637             printf(ready);
638             //sys_req(WRITE, COM1, ready, &check );
639
640             PCB* pcb = getReady()->head;
641
642             if(pcb->next == NULL) {
643                 class = pcb->Process_Class;
644                 strcpy(name,pcb->Process_Name);
645                 state = pcb->ReadyState;
646                 status = pcb->SuspendedState;

```

```

647         prior = pcb->Priority;
648
649         printf(cname);
650         printf(name);
651         printf(line);
652
653         printf(cclass);
654         if(pcb->Process_Class == 0) {
655             printf("0");
656         }
657         else {
658             printf(itoa(class));
659             //sys_req(WRITE, COM1, itoa(class), &check);
660         }
661         printf(line);
662
663         printf(cstate);
664         if(pcb->ReadyState == 0) {
665             printf("0");
666         }
667         else {
668             printf(itoa(state));
669             //sys_req(WRITE, COM1, itoa(state), &check);
670         }
671         printf(line);
672
673         printf(cstatus);
674         if(pcb->SuspendedState == 0) {
675             printf("0");
676         }
677         else {
678             printf(itoa(status));
679             //sys_req(WRITE, COM1, itoa(status), &check);
680         }
681         printf(line);
682
683         printf(cprior);
684         if(pcb->Priority == 0) {
685             printf("0");
686             printf("\n\n");
687         }
688         else {
689             printf(itoa(prior));
690             //sys_req(WRITE, COM1, itoa(prior), &check);
691             printf("\n\n");
692         }
693     }
694     else {
695         while(pcb != NULL) {
696             class = pcb->Process_Class;
697             strcpy(name, pcb->Process_Name);
698             state = pcb->ReadyState;
699             status = pcb->SuspendedState;
700             prior = pcb->Priority;
701
702             printf(cname);
703             printf(name);
704             printf(line);
705
706             printf(cclass);
707             if(pcb->Process_Class == 0) {
708                 printf("0");
709             }
710             else {
711                 printf(itoa(class));
712                 //sys_req(WRITE, COM1, itoa(class), &check);
713             }
714             printf(line);
715
716             printf(cstate);
717             if(pcb->ReadyState == 0) {
718                 printf("0");
719             }
720             else {
721                 printf(itoa(state));
722                 //sys_req(WRITE, COM1, itoa(state), &check);
723             }
724             printf(line);
725
726             printf(cstatus);
727             if(pcb->SuspendedState == 0) {
728                 printf("0");
729             }
730             else {
731                 printf(itoa(status));
732                 //sys_req(WRITE, COM1, itoa(status), &check);
733             }

```

```

734         printf(line);
735
736         printf(cprior);
737         if(pcb->Priority == 0) {
738             printf("0");
739             printf("\n\n");
740         }
741         else {
742             printf(itoa(prior));
743             //sys_req(WRITE, COM1, itoa(prior), &check);
744             printf("\n\n");
745         }
746         pcb = pcb->next;
747     }
748 }
749 }
750 }

```

4.22.1.19 Suspend()

```

void Suspend (
    char * ProcessName )

```

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

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process.
---------------------	---

Definition at line 457 of file userFunctions.c.

```

457     {
458         PCB* pcb = FindPCB(ProcessName);
459         if (pcb == NULL) {
460             printf("\x1b[31m"\nERROR: Not a valid process name \n"\x1b[0m");
461         }
462         else {
463             if(pcb->SuspendedState == YES) {
464                 printf("\x1b[32m"\nThis Process is already SUSPENDED \n"\x1b[0m");
465             }
466             else if (pcb -> Process_Class == APPLICATION) {
467                 pcb->SuspendedState = YES;
468             }
469             else
470                 printf("\x1b[31m"\nERROR: Cannot Alter System Process \n"\x1b[0m");
471         }
472     }

```

4.22.1.20 toLowercase()

```

char toLowercase (
    char c )

```

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

Parameters

<i>c</i>	Character that is to be changed to its lowercase equivalent
----------	---

Definition at line 314 of file userFunctions.c.

```

314         {
315     if ((c >= 65) && (c <= 90)) {
316         c = c + 32;
317     }
318     return c;
319 }
```

4.22.1.21 Unblock()

```

void Unblock (
    char * ProcessName )
```

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

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

Parameters

<i>Process_Name</i>	Character pointer that matches the name of process.
---------------------	---

Definition at line 983 of file userFunctions.c.

```

983         {
984     PCB* pcb = FindPCB(ProcessName);
985     if (pcb == NULL) {
986         printf("\x1b[31m""\nERROR: Not a valid process name \n""\x1b[0m");
987     }
988     else {
989         if (pcb->ReadyState == READY) {
990             printf("\x1b[32m""\nThis Process is already in the READY state \n""\x1b[0m");
991         }
992         else {
993             RemovePCB(pcb);
994             pcb->ReadyState = READY;
995             InsertPCB(pcb);
996         }
997     }
998 }
```

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 307 of file userFunctions.c.

```

307     {
308         printf("Version: R5.2 \n");
309     }
```

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](#)
AlarmList
 [userFunctions.c, 54](#)
atoi
 [string.c, 27](#)
 [string.h, 18](#)

BCDtoDec
 [userFunctions.c, 38](#)
 [userFunctions.h, 56](#)
Block
 [userFunctions.c, 39](#)
 [userFunctions.h, 57](#)

CMCB, [5](#)
comHand
 [comHand.h, 33](#)
comHand.h
 [comHand, 33](#)
context, [6](#)
Create_PCB
 [userFunctions.c, 39](#)
 [userFunctions.h, 57](#)

date_time, [6](#)
DectoBCD
 [userFunctions.c, 40](#)
 [userFunctions.h, 58](#)
Delete_PCB
 [userFunctions.c, 40](#)
 [userFunctions.h, 58](#)

EdgeCase
 [userFunctions.c, 41](#)
 [userFunctions.h, 59](#)
else
 [userFunctions.c, 55](#)

footer, [7](#)

gdt_descriptor_struct, [7](#)
gdt_entry_struct, [7](#)
GetDate
 [userFunctions.c, 41](#)
 [userFunctions.h, 59](#)
GetTime
 [userFunctions.c, 42](#)
 [userFunctions.h, 60](#)

header, [8](#)
heap, [8](#)

Help
 [userFunctions.c, 42](#)
 [userFunctions.h, 60](#)

idt_entry_struct, [9](#)
idt_struct, [9](#)
inb
 [io.h, 15](#)
include/core/asm.h, [15](#)
include/core/interrupts.h, [15](#)
include/core/io.h, [15](#)
include/core/serial.h, [16](#)
include/core/tables.h, [16](#)
include/mem/heap.h, [17](#)
include/mem/paging.h, [17](#)
include/string.h, [18](#)
include/system.h, [22](#)
index_entry, [9](#)
index_table, [10](#)
io.h
 [inb, 15](#)
isspace
 [string.c, 28](#)
 [string.h, 18](#)
itoa
 [userFunctions.c, 44](#)
 [userFunctions.h, 62](#)

kernel/core/interrupts.c, [23](#)
kernel/core/kmain.c, [24](#)
kernel/core/serial.c, [24](#)
kernel/core/system.c, [25](#)
kernel/core/tables.c, [25](#)
kernel/mem/heap.c, [26](#)
kernel/mem/paging.c, [26](#)

lib/string.c, [27](#)
List, [10](#)

MemList, [10](#)
memset
 [string.c, 28](#)
 [string.h, 19](#)
modules/mpx_supt.c, [31](#)
modules/mpx_supt.h, [32](#)
modules/R1/comHand.h, [33](#)
modules/R1/userFunctions.c, [37](#)
modules/R1/userFunctions.h, [55](#)
modules/sys_proc_loader.c, [73](#)
modules/sys_proc_loader.h, [73](#)

- page_dir, [11](#)
- page_entry, [11](#)
- page_table, [12](#)
- param, [12](#)
- PCB, [12](#)
- Queue, [13](#)
- Resume
 - userFunctions.c, [45](#)
 - userFunctions.h, [63](#)
- Set_Priority
 - userFunctions.c, [45](#)
 - userFunctions.h, [63](#)
- SetDate
 - userFunctions.c, [46](#)
 - userFunctions.h, [64](#)
- SetTime
 - userFunctions.c, [47](#)
 - userFunctions.h, [65](#)
- Show_All
 - userFunctions.c, [48](#)
 - userFunctions.h, [66](#)
- Show_Blocked
 - userFunctions.c, [48](#)
 - userFunctions.h, [66](#)
- Show_PCB
 - userFunctions.c, [50](#)
 - userFunctions.h, [68](#)
- Show_Ready
 - userFunctions.c, [51](#)
 - userFunctions.h, [69](#)
- strcat
 - string.c, [29](#)
 - string.h, [19](#)
- strcmp
 - string.c, [29](#)
 - string.h, [20](#)
- strcpy
 - string.c, [30](#)
 - string.h, [20](#)
- string.c
 - atoi, [27](#)
 - isspace, [28](#)
 - memset, [28](#)
 - strcat, [29](#)
 - strcmp, [29](#)
 - strcpy, [30](#)
 - strlen, [30](#)
 - strtok, [30](#)
- string.h
 - atoi, [18](#)
 - isspace, [18](#)
 - memset, [19](#)
 - strcat, [19](#)
 - strcmp, [20](#)
 - strcpy, [20](#)
 - strlen, [21](#)
 - strtok, [21](#)
- strlen
 - string.c, [30](#)
 - string.h, [21](#)
- strtok
 - string.c, [30](#)
 - string.h, [21](#)
- Suspend
 - userFunctions.c, [53](#)
 - userFunctions.h, [71](#)
- toLowerCase
 - userFunctions.c, [53](#)
 - userFunctions.h, [71](#)
- Unblock
 - userFunctions.c, [54](#)
 - userFunctions.h, [72](#)
- userFunctions.c
 - AlarmList, [54](#)
 - BCDtoDec, [38](#)
 - Block, [39](#)
 - Create_PCB, [39](#)
 - DectoBCD, [40](#)
 - Delete_PCB, [40](#)
 - EdgeCase, [41](#)
 - else, [55](#)
 - GetDate, [41](#)
 - GetTime, [42](#)
 - Help, [42](#)
 - itoa, [44](#)
 - Resume, [45](#)
 - Set_Priority, [45](#)
 - SetDate, [46](#)
 - SetTime, [47](#)
 - Show_All, [48](#)
 - Show_Blocked, [48](#)
 - Show_PCB, [50](#)
 - Show_Ready, [51](#)
 - Suspend, [53](#)
 - toLowerCase, [53](#)
 - Unblock, [54](#)
 - Version, [54](#)
- userFunctions.h
 - BCDtoDec, [56](#)
 - Block, [57](#)
 - Create_PCB, [57](#)
 - DectoBCD, [58](#)
 - Delete_PCB, [58](#)
 - EdgeCase, [59](#)
 - GetDate, [59](#)
 - GetTime, [60](#)
 - Help, [60](#)
 - itoa, [62](#)
 - Resume, [63](#)
 - Set_Priority, [63](#)
 - SetDate, [64](#)
 - SetTime, [65](#)
 - Show_All, [66](#)

Show_Blocked, [66](#)
Show_PCB, [68](#)
Show_Ready, [69](#)
Suspend, [71](#)
toLowerCase, [71](#)
Unblock, [72](#)
Version, [72](#)

Version

userFunctions.c, [54](#)
userFunctions.h, [72](#)