Jithub

R1

Generated by Doxygen 1.8.17

Chapter 1

File Index

1.1 File List

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

Chapter 2

File Documentation

2.1 include/comhand.h File Reference

A set of functions that allow users to interact with the OS.

Macros

- #define **VERSION** 1

Functions

- void comhand (void)

  calls all of the different functions that a user would use within the OS
- void printMenu ()

  Prints the menu that has different functions the user can choose from to use.
- int shutdown ()

  allows the user to shutdown the OS, but requires confirmation
- void version (void)

  Shows the user which version of the OS they are using.
- void getTime (void)

  Gets the current time set within the OS.
- void setTime (void)

  Allows the user to set the time of the OS, but doesn't allow them to input an invalid time.
- void getDate (void)

  Gets the current date set within the OS.
- void setDate (void)

  Allows the user to set the date within the OS, but does not allow any invalid dates.
- void help (void)

  Displays a list of all of the different functions and what they are made to do.

2.1.1 Detailed Description

A set of functions that allow users to interact with the OS.

2.1.2 Function Documentation

2.1.2.1 comhand()

```
void comhand (
            void  )
```

calls all of the different functions that a user would use within the OS

Author

Sam Desai

Jackson Monk

2.1.2.2 getDate()

```
void getDate (
            void  )
```

Gets the current date set within the OS.

Author

Sam Desai

Jackson Monk

Attaches the index of the time register (mins, seconds, hours, etc) to the index register of the RTC. Then, using inb to read from the RTC data port (0x71), puts the value in an integer and formats the value to decimal using the fromBCD function. This is done for days, month, and year. Finally, checks for single digits in days or month and adds padding zeros, then prints out the value.

2.1.2.3 getTime()

```
void getTime (
            void  )
```

Gets the current time set within the OS.

Author

Sam Desai

Jackson Monk

Attaches the index of the time register (mins, seconds, hours, etc) to the index register of the RTC. Then, using inb to read from the RTC data port (0x71), puts the value in an integer and formats the value to decimal using the fromBCD function. This is done for hours, minutes, and seconds. Finally, checks for single digits in minutes or seconds and adds padding zeros, then prints out the value.

2.1.2.4 help()

```
void help (
            void  )
```

Displays a list of all of the different functions and what they are made to do.

Author

Sam Desai

Jackson Monk

Prints a list of commands with explanations. This function will also take one input and give specific details for that input.

2.1.2.5 printMenu()

```
void printMenu ( )
```

Prints the menu that has different functions the user can choose from to use.

Author

Sam Desai

Jackson Monk

2.1.2.6 setDate()

```
void setDate (
            void  )
```

Allows the user to set the date within the OS, but does not allow any invalid dates.

Author

Sam Desai

Jackson Monk

Starts by validating the input with numerous tests. Disables interrupts, converts the inputs into BCD format by calling the toBCD function and attaches the index of the time register (days, minutes, years, etc) to the index register of the RTC. Lastly, uses outb to write to the BCD values to the data port, prints out the time if successful, and reenables interrupts.

2.1.2.7 setTime()

```
void setTime (
             void  )
```

Allows the user to set the time of the OS, but doesn't allow them to input an invalid time.

Author

Sam Desai

Jackson Monk

Noah Marner

Starts by validating the minutes and seconds by checking if they are over 59 and adds padding to single digit hour field. Then checks if all inputs are digits. If any validating reveals errors, sets a flag to 1 to exit the function. Disables interrupts, converts the inputs into BCD format by calling the toBCD function and attaches the index of the time register (mins, seconds, hours, etc) to the index register of the RTC. Lastly, uses outb to write to the BCD values to the data port, prints out the time if successful, and reenables interrupts.

2.1.2.8 shutdown()

```
int shutdown ( )
```

allows the user to shutdown the OS, but requires confirmation

Author

Sam Desai

Jackson Monk

Returns

int - returns a 1 if the user confirmed shutdown or a 0 if the user denied shutdown

2.1.2.9 version()

```
void version (
             void  )
```

Shows the user which version of the OS they are using.

Author

Sam Desai

Jackson Monk

2.2 include/ctype.h File Reference

A subset of standard C library functions.

Functions

- int isspace (int c)

2.2.1 Detailed Description

A subset of standard C library functions.

2.2.2 Function Documentation

2.2.2.1 isspace()

```
int isspace (
            int c )
```

Determine if a character is whitespace.

Parameters

c	Character to check

Returns

Non-zero if space, 0 if not space

2.3 include/memory.h File Reference

MPX-specific dynamic memory functions.

```
#include <stddef.h>
```
Include dependency graph for memory.h:

2.4 include/mpx/gdt.h File Reference

Kernel functions to initialize the Global Descriptor Table.

Functions

- void gdt_init (void)

2.4.1 Detailed Description

Kernel functions to initialize the Global Descriptor Table.

2.4.2 Function Documentation

2.4.2.1 gdt_init()

```
void gdt_init (
            void  )
```

Creates and installs the Global Descriptor Table.

2.5 include/mpx/interrupts.h File Reference

Kernel functions related to software and hardware interrupts.

Macros

- #define cli() __asm__ volatile ("cli")
- #define sti() __asm__ volatile ("sti")

Functions

- void irq_init (void)
- void pic_init (void)
- void idt_init (void)
- void idt_install (int vector, void(*handler)(void *))

2.5.1 Detailed Description

Kernel functions related to software and hardware interrupts.

2.5.2 Macro Definition Documentation

2.5.2.1 cli

```
#define cli( ) __asm__ volatile ("cli")
```

Disable interrupts

2.5.2.2 sti

```
#define sti( ) __asm__ volatile ("sti")
```

Enable interrupts

2.5.3 Function Documentation

2.5.3.1 idt_init()

```
void idt_init (
            void  )
```

Creates and installs the Interrupt Descriptor Table.

2.5.3.2 idt_install()

```
void idt_install (
            int vector,
            void(*)(void *) handler )
```

Installs an interrupt handler

2.5.3.3 irq_init()

```
void irq_init (
            void  )
```

Installs the initial interrupt handlers for the first 32 IRQ lines. Most do a panic for now.

2.5.3.4 pic_init()

```
void pic_init (
            void  )
```

Initializes the programmable interrupt controllers and performs the necessary remapping of IRQs. Leaves interrupts turned off.

2.6 include/mpx/io.h File Reference

Kernel macros to read and write I/O ports.

Macros

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

2.6.1 Detailed Description

Kernel macros to read and write I/O ports.

2.6.2 Macro Definition Documentation

2.6.2.1 inb

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

Value:

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

Read one byte from an I/O port

Parameters

port	The port to read from

Returns

A byte of data read from the port

2.6.2.2 outb

```
#define outb(
              port,
              data ) __asm__ volatile ("outb %%al, %%dx" ::  "a" (data), "d" (port))
```

Write one byte to an I/O port

Parameters

port	The port to write to
data	The byte to write to the port

2.7 include/mpx/panic.h File Reference

Common system functions and definitions.

```
#include <stdnoreturn.h>
```
Include dependency graph for panic.h:

Functions

- noreturn __attribute__ ((no_caller_saved_registers)) void kpanic(const char *msg)

2.7.1 Detailed Description

Common system functions and definitions.

2.7.2 Function Documentation

2.7.2.1 __attribute__()

```
noreturn __attribute__ (
            (no_caller_saved_registers)  ) const
```

Kernel panic. Prints an error message and halts.

Parameters

msg	A message to display before halting

2.8 include/mpx/serial.h File Reference

Kernel functions and constants for handling serial I/O.

```
#include <stddef.h>
#include <mpx/device.h>
```
Include dependency graph for serial.h:

Functions

- int serial_init (device dev)
- int serial_out (device dev, const char *buffer, size_t len)
- int serial_poll (device dev, char *buffer, size_t len)

  Reads a string from a serial port.
- void backspace (int *pos, int *end, char *buffer, device dev)

  Helper method for serial_poll that does the work of a backspace.

2.8.1 Detailed Description

Kernel functions and constants for handling serial I/O.

2.8.2 Function Documentation

2.8.2.1 backspace()

```
void backspace (
            int * pos,
            int * end,
            char * buffer,
            device dev )
```

Helper method for serial_poll that does the work of a backspace.

Author

Noah Marner

Blake Wagner

This function is used when a backspace or delete character is input. If there is a backspace, the function is simply called. When a delete character is input, the position is set forward one and then the function is called.

Parameters

pos	The current position in the buffer
end	The farthest position that has been reached in the buffer
buffer	A buffer that stores the current string
dev	The serial port to read data from

2.8.2.2 serial_init()

```
int serial_init (
            device dev )
```

Initializes devices for user input and output

Parameters

device	A serial port to initialize (COM1, COM2, COM3, or COM4)

Returns

0 on success, non-zero on failure

2.8.2.3 serial_out()

```
int serial_out (
            device dev,
            const char * buffer,
            size_t len )
```

Writes a buffer to a serial port

Parameters

device	The serial port to output to
buffer	A pointer to an array of characters to output
len	The number of bytes to write

Returns

The number of bytes written

2.8.2.4 serial_poll()

```
int serial_poll (
            device dev,
            char * buffer,
            size_t len )
```

Reads a string from a serial port.

Author

Noah Marner

Blake Wagner

This function is used to read in data from the console. It reads characters until either the length limit is reached or an enter key is read in. Special characters such as backspace, delete and arrow keys are also handled in this function.

Parameters

device	The serial port to read data from
buffer	A buffer to write data into as it is read from the serial port
count	The maximum number of bytes to read

Returns

The number of bytes read on success, a negative number on failure

2.9 include/mpx/vm.h File Reference

Kernel functions for virtual memory and primitive allocation.

```
#include <stddef.h>
```

Include dependency graph for vm.h:

Functions

- void * kmalloc (size_t size, int align, void **phys_addr)
- void vm_init (void)

2.9.1 Detailed Description

Kernel functions for virtual memory and primitive allocation.

2.9.2 Function Documentation

2.9.2.1 kmalloc()

```
void* kmalloc (
            size_t size,
            int align,
            void ** phys_addr )
```

Allocates memory from a primitive heap.

Parameters

size	The size of memory to allocate
align	If non-zero, align the allocation to a page boundary
phys_addr	If non-NULL, a pointer to a pointer that will hold the physical address of the new memory

Returns

The newly allocated memory

2.9.2.2 vm_init()

```
void vm_init (
             void  )
```

Initializes the kernel page directory and initial kernel heap area. Performs identity mapping of the kernel frames such that the virtual addresses are equivalent to the physical addresses.

2.10 include/processes.h File Reference

Provided system process and user processes for testing.

Functions

- void proc1 (void)
- void proc2 (void)
- void proc3 (void)
- void proc4 (void)
- void proc5 (void)
- void sys_idle_process (void)

2.10.1 Detailed Description

Provided system process and user processes for testing.

2.10.2 Function Documentation

2.10.2.1 proc1()

```
void proc1 (
             void  )
```

A test process that prints a message then yields, exiting after 1 iteration.

2.10.2.2 proc2()

```
void proc2 (
            void  )
```

A test process that prints a message then yields, exiting after 2 iterations.

2.10.2.3 proc3()

```
void proc3 (
            void  )
```

A test process that prints a message then yields, exiting after 3 iterations.

2.10.2.4 proc4()

```
void proc4 (
            void  )
```

A test process that prints a message then yields, exiting after 4 iterations.

2.10.2.5 proc5()

```
void proc5 (
            void  )
```

A test process that prints a message then yields, exiting after 5 iterations.

2.10.2.6 sys_idle_process()

```
void sys_idle_process (
            void  )
```

System idle process. Used in dispatching. It will be dispatched if NO other processes are available to execute. Must be a system process.

2.11 include/stdio.h File Reference

A set of functions for input and output interactions.

Functions

- void putc (char c)

  Prints a single character to the console.
- void puts (const char *s)

  Prints a string the console.
- void printf (const char *format,...)

  Prints string to the console with insertable values specified by subsequent parameters.

2.11.1 Detailed Description

A set of functions for input and output interactions.

2.11.2 Function Documentation

2.11.2.1 printf()

```
void printf (
            const char * format,
             ...  )
```

Prints string to the console with insertable values specified by subsequent parameters.

Author

Samesh Desai
Noah Marner
Jackson Monk
Blake Wagner

Creates a pointer to look at the first argument. Loops while there are characters remaining in the format string. When we encounter a % sign in the format string, pulls from the argument pointer and increments it.

Parameters

format	The format in which the string is specified. Insertable values are specified with a % symbol

2.11.2.2 putc()

```
void putc (
            char c )
```

Prints a single character to the console.

Author

Samesh Desai
Noah Marner
Jackson Monk
Blake Wagner

This function utilizes sys_req(WRITE) to print a single character to the COM1 output register

Parameters

c	The character to print to the console

2.11.2.3 puts()

```
void puts (
            const char * s )
```

Prints a string the console.

Author

Samesh Desai

Noah Marner

Jackson Monk

Blake Wagner

This function utilizes sys_req(WRITE) to print a string to the COM1 output register

Parameters

s	The string to print to the console

2.12 include/stdlib.h File Reference

A subset of standard C library functions.

Functions

- int atoi (const char *s)

  Convert an ASCII string to an integer.
- char * itoa (int i, char *buffer)

  Converts a valid integer to a string and stores it in buffer.
- int isdigit (char c)

  Checks if the provided character is a digit.
- int toBCD (int num)

  Converts a string to a binary-coded decimal (bcd)
- int fromBCD (int bcd)

  Converts a binary-coded decimal to an int.

2.12.1 Detailed Description

A subset of standard C library functions.

2.12.2 Function Documentation

2.12.2.1 atoi()

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

Convert an ASCII string to an integer.

Parameters

s	A NUL-terminated string

Returns

The value of the string converted to an integer

2.12.2.2 fromBCD()

```
int fromBCD (
            int bcd )
```

Converts a binary-coded decimal to an int.

Author

Sam Desai

Jackson Monk

This function starts by getting the 10s place by anding the input value with 0x70 (1111 0000). Then it gets the ones place by anding with 0x0F (0000 1111). It gets the final value by adding these two values.

Parameters

c	The BCD value to convert

Returns

The integer representation of the integer

2.12.2.3 isdigit()

```
int isdigit (
            char c )
```

Checks if the provided character is a digit.

Author

Jackson Monk

This function checks if the character is equal to 0-9. If true it returns 1, if not, it returns 0

Parameters

c	The character to compare to a digit

Returns

An integer indicating whether or not the character is a digit. Non-zero if true, 0 if false

2.12.2.4 itoa()

```
char* itoa (
            int i,
            char * buffer )
```

Converts a valid integer to a string and stores it in buffer.

Author

Samesh Desai

Noah Marner

Jackson Monk

Blake Wagner

First, this function checks if the number is negative and sets the negative flag according to the result. Next, if the number is 0, we set the buffer to 0, increment position, add a null terminator, and exit. While i != 0, the remainder of i % 10 is found. The buffer at the current position is then set to the remainder plus the '0' character to calculate the number character, then i is divided by 10 to go to the digit. After the loop, the number is in reverse order; we add a '-' character to the end if the number is negative. Last, reverse the string and add a null terminator.

Parameters

i	The integer to convert to a string
buffer	The destination string for the result

Returns

The same string placed into buffer

2.12.2.5 toBCD()

```
int toBCD (
            int num )
```

Converts a string to a binary-coded decimal (bcd)

Author

Jackson Monk

Gets the 10s digit by dividing by 10 and the ones digit by modding by 10. We then shift the 10s digit four digits to the right (pad with four zeros). We then or the 10s digit and the ones digit together to get the final result.

Parameters

c	The integer to convert

Returns

The bcd representation of the integer

2.13 include/string.h File Reference

A subset of standard C library functions.

`#include <stddef.h>`
Include dependency graph for string.h:

Functions

- void ∗ memcpy (void ∗restrict dst, const void ∗restrict src, size_t n)
- void ∗ memset (void ∗address, int c, size_t n)
- int strcmp (const char ∗s1, const char ∗s2)
- int strcmp_ic (const char ∗s1, const char ∗s2)

  A string compare function that ignores case.
- size_t strlen (const char ∗s)
- char ∗ strtok (char ∗restrict s1, const char ∗restrict s2)
- void **str_copy** (char ∗dest, char ∗source, int start, size_t length)

2.13.1 Detailed Description

A subset of standard C library functions.

2.13.2 Function Documentation

2.13.2.1 memcpy()

```
void* memcpy (
            void *restrict dst,
            const void *restrict src,
            size_t n )
```

Copy a region of memory.

Parameters

dst	The destination memory region
src	The source memory region
n	The number of bytes to copy

Returns

A pointer to the destination memory region

2.13.2.2 memset()

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

Fill a region of memory.

Parameters

address	The start of the memory region
c	The byte to fill memory with
n	The number of bytes to fill

Returns

A pointer to the filled memory region

2.13.2.3 strcmp()

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

Compares two strings

Parameters

s1	The first string to compare
s2	The second string to compare

Returns

0 if strings are equal, <0 if s1 is lexicographically before s2, >0 otherwise

2.13.2.4 strcmp_ic()

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

A string compare function that ignores case.

Author

Noah Marner

This function compares two strings while ignoring case. This case is used in comhand to make comparing user input more friendly.

Parameters

s1	The first string to compare
s2	the second string to compare

Returns

0 if strings are equal ignoring case, 1 if not

2.13.2.5 strlen()

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

Returns the length of a string.

Parameters

s	A NUL-terminated string

Returns

The number of bytes in the string (not counting NUL terminator)

2.13.2.6 strtok()

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

Split string into tokens TODO

2.14 include/sys_req.h File Reference

System request function and constants.

`#include <mpx/device.h>`
Include dependency graph for sys_req.h:

Macros

- #define **INVALID_OPERATION** (-1)
- #define **INVALID_BUFFER** (-2)
- #define **INVALID_COUNT** (-3)

Enumerations

- enum **op_code** { **EXIT**, **IDLE**, **READ**, **WRITE** }

Functions

- int sys_req (op_code op,...)

2.14.1 Detailed Description

System request function and constants.

2.14.2 Function Documentation

2.14.2.1 sys_req()

```
int sys_req (
            op_code op,
             ...  )
```

Request an MPX kernel operation.

Parameters

op_code	One of READ, WRITE, IDLE, or EXIT
...	As required for READ or WRITE

Returns

Varies by operation