Jithub

R2

Generated by Doxygen 1.8.17

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 pcb Struct Reference

Collaboration diagram for pcb:

Data Fields

- char ∗ **name_ptr**
- int **priority**
- int **clas**
- char ∗ **state**
- struct pcb ∗ **next**
- unsigned char **stack** [1024]
- char ∗ **stack_ptr**

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

- include/dataStructs.h

3.2 queue Struct Reference

Collaboration diagram for queue:

Data Fields

- struct pcb ∗ **head**
- struct pcb ∗ **tail**
- int **pFlag**

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

- include/dataStructs.h

Chapter 4

File Documentation

4.1 include/comhand.h File Reference

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

```
#include <dataStructs.h>
```

Include dependency graph for comhand.h:

4.2 include/ctype.h File Reference

A subset of standard C library functions.

Functions

- int isspace (int c)

4.2.1 Detailed Description

A subset of standard C library functions.

4.2.2 Function Documentation

4.2.2.1 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

4.3 include/dataStructs.h File Reference

Data structures associated with processes and the functions to go with them.

```
#include <memory.h>
```

Include dependency graph for dataStructs.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct pcb
- struct queue

Typedefs

- typedef struct pcb **pcb**
- typedef struct queue **queue**

Functions

- void enqueue (queue ∗q, pcb ∗newPCB)

  Adds a new PCB to the queue.
- pcb ∗ dequeue (queue ∗q)

  Removes a pcb from the front of the queue.
- void printq (queue ∗q)

  Prints the contents of the queue.
- pcb ∗ pcb_allocate (void)

  Allocate memory for a PCB.
- int pcb_free (pcb ∗pcb)

  deallocates the memory associated with a PCB
- pcb ∗ pcb_setup (const char ∗name, int clas, int priority)

  Creates a new PCB.
- pcb ∗ pcb_find (queue ∗ready, queue ∗blocked, queue ∗susReady, queue ∗susBlocked, const char ∗name)

  Given a name, returns a pointer to the PCB.
- int pcb_insert (queue ∗ready, queue ∗blocked, queue ∗susReady, queue ∗susBlocked, pcb ∗pcb)

  Inserts a PCB into the proper queue.
- void pcb_remove (queue ∗ready, queue ∗blocked, queue ∗susReady, queue ∗susBlocked, pcb ∗pcb)

  Removes a PCB.

4.3.1 Detailed Description

Data structures associated with processes and the functions to go with them.

4.3.2 Function Documentation

4.3.2.1 dequeue()

```
pcb* dequeue (
             queue * q )
```

Removes a pcb from the front of the queue.

Author

Samesh Desai

Noah Marner

Jackson Monk

Blake Wagner

Parameters

q	

Returns

Returns the removed PCB

Checks if the queue is empty (Head is NULL). Creates a temp pcb to point to the head. If heads next exists, set head to point to head next and returns the temp pcb which points to the old head. Otherwise the queue only has 1 item so the head and tail are set to NULL and we return temp

4.3.2.2 enqueue()

```
void enqueue (
             queue * q,
             pcb * newPCB )
```

Adds a new PCB to the queue.

Author

Samesh Desai

Noah Marner

Jackson Monk

Blake Wagner

Parameters

q	
newPCB	

Receives a new PCB and an existing queue. Checks if the queue is empty by checking if the head and tail pointers are null. If so the new PCB is the new head and tail. Checks the priortiy of the PCB received and itterates through the queue until we find processes with the same priority. Process that have the same priority are first in first out. The last PCB with the same priority now has its next pointer looking at the PCB we are inserting. The new PCB next now looks at the old next of the last PCB with the same priority.

4.3.2.3 pcb_allocate()

```
pcb* pcb_allocate (
             void )
```

Allocate memory for a PCB.

Author

Samesh Desai

Noah Marner

Jackson Monk

Blake Wagner

Returns

PCB pointer

Uses sysallocmem function to allocate memeory for a pcb based on the size of the PCB struct. Then returns a pointer to the newly created PCB. If memory fails to allocate returns NULL and prints an error message.

4.3.2.4 pcb_find()

```
pcb* pcb_find (
             queue * ready,
             queue * blocked,
             queue * susReady,
             queue * susBlocked,
             const char * name )
```

Given a name, returns a pointer to the PCB.

Author

Blake Wagner

Jackson Monk

Noah Marner

Samesh Desai

Parameters

ready	
blocked	
susReady	
susBlocked	
name	

Given a name, iterates through all the queues (ready, blocked, susReady, susBlocked) by creating a temp PCB pointer starting at the head of each queue. If a PCB has a name that matches the provided name immediatly returns the temp pointer which is pointing to that PCB.

4.3.2.5 pcb_free()

```
int pcb_free (
            pcb * pcb )
```

deallocates the memory associated with a PCB

Author

Samesh Desai

Noah Marner

Blake Wagner

Jackson Monk

Parameters

pcb	

Returns

1 or -1

Receives a PCB pointer and uses sysmemfree to deallocate the memory associated with the passed in PCB. If the memory freeing is sucessful, returns 1. If not returns -1 and prints and error message.

4.3.2.6 pcb_insert()

```
int pcb_insert (
            queue * ready,
            queue * blocked,
            queue * susReady,
            queue * susBlocked,
            pcb * pcb )
```

Inserts a PCB into the proper queue.

Author

Noah Marner

Jackson Monk

Blake Wagner

Samesh Desai

Parameters

ready	
blocked	
susReady	
susBlocked	
pcb	

Based on the state of the provided PCB calls enqueue with the provided PCB to add the PCB into either the ready, blocked, susReady, or susBlocked. Returns 0 for a successful insertion and -1 if failed.

4.3.2.7 pcb_remove()

```
void pcb_remove (
             queue * ready,
             queue * blocked,
             queue * susReady,
             queue * susBlocked,
             pcb * pcb )
```

Removes a PCB.

Author

Noah Marner

Jackson Monk

Blake Wagner

Samesh Desai

Parameters

ready	
blocked	
susReady	
susBlocked	
pcb	

Determines the queue the process is in based on the state of the passed in PCB. Creates a temp current PCB pointer. Iterates through the queue to find the PCB to remove. If PCB to remove is the head set head to point to next and return temp. Otherwise previous now pointer to current next. This circumvents the PCB we wish to remove. Lastly, check if the desired PCB is the tail and if so set the new tail to be the previous PCB since we are removing the tail.

4.3.2.8 pcb_setup()

```
pcb* pcb_setup (
            const char * name,
            int clas,
            int priority )
```

Creates a new PCB.

Author

Jackson Monk

Blake Wagner

Noah Marner

Samesh Desai

Parameters

name	
clas	
priority	

Takes in a char∗ for name, int for clas, and int for prioirty. Calls PCB allocate to create a new PCB, then assigns the passed in parameters to the data members of the PCB struct. When creating a new PCB, the defualt state is ready. Calls sysallocatemem for the name and state. Returns a pointer to the newly created PCB.

4.3.2.9 printq()

```
void printq (
            queue * q )
```

Prints the contents of the queue.

Author

Samesh Desai

Noah Marner

Jackson Monk

Blake Wagner

Parameters

q	

Given a queue, creates a temp PCB pointer to look at head, prints the contents of that PCB, looks at the current PCBs next, and prints the conents. The functions keeps iterating through the queue until next points to NULL.

4.4 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)

4.4.1 Detailed Description

Kernel macros to read and write I/O ports.

4.4.2 Macro Definition Documentation

4.4.2.1 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

4.4.2.2 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

4.5 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.

4.5.1 Detailed Description

Kernel functions and constants for handling serial I/O.

4.5.2 Function Documentation

4.5.2.1 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

4.5.2.2 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

4.5.2.3 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

4.5.2.4 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

4.6 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.

4.6.1 Detailed Description

A set of functions for input and output interactions.

4.6.2 Function Documentation

4.6.2.1 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

4.6.2.2 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

4.6.2.3 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

4.7 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.

4.7.1 Detailed Description

A subset of standard C library functions.

4.7.2 Function Documentation

4.7.2.1 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

4.7.2.2 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

4.7.2.3 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

4.7.2.4 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

4.7.2.5 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