Programmers Manual

Created By:Sam, Glen, Brett, Luke, Connor

Table of Contents

1. MPX System Functions.. (3)
- com_hand.. (3)
- getSeconds.. (4)
- getMinutes.. (5)
- getHours.. (5)
- BcDtoDec.. (5)
- getDay.. (5)
- getMonth.. (6)
- getYear.. (7)
- getTime.. (7)
- getDate.. (7)
- setDate.. (8)
- setTime.. (8)
- serial_poll.. (8)

2. MPX PCB Functions.. (9)
- pcb_allocate.. (9)
- pcb_free ....(9)
- pcb_setup ....(10)
- pcb_find ....(10)
- pcb_insert .... (11)
- pcb_remove ....(11)
- copy_string……………………………………………………………………………….(12)

2. MPX PCB Data Structures.. (13)
- Enums.. (13)
- Struct PCB .... (14)
- Queue.h .... (15)
- Queue .... (15)
- Queue_node .... (15)
- Helper Methods .... (15)
- Is_empty……………………………………………………………………………… (16)
- Dequeue……………………………………………………………………………… (16)
- Enqueue………………………………………………………………………………(17)
- create_queue……………………………………………………………………… (18)
- readyQueue……………………………………………………………………...(18)
-blockedQueue………………………………………………………………...…..(18)

1. MPX System Functions

R1 com_hand()

void com_hand(void)

Author: SleepDeprivedDebuggers

This function contains all of the commands that can be run using the buffer. When a user enters a command it checks if the command is valid, and if it is indeed valid, runs the code that corresponds with the command.

Commands:

Help - uses the char arrays created at the top of com_hand() and prints a help menu to the screen

Shutdown - This command shuts down the MPX. The user will confirm the selection.

Version - is updated manually with each new version.

Get Time - This command checks that the user inputs the correct command being, Get Time, get time, gettime, GetTime, to get the time this method uses the computer's internal clock registers to provide the user with the current time (maybe off due to where the clock was created).

Set Time - This command first checks if the user imputed the correct command the correct commands are, Set Time, set time, settime, SetTime, once a correct command is entered then this method is run where the user is prompted to enter the hours(0-24) minutes (0-59) and seconds (0-59) where anything outside of this range is given an error and prompts the user to try again until a correct time is an input. After that then the time is written to the computer's clock memory and overrides the previous time this is the one that the user entered.

Set Date - This command first checks if the user imputed the correct command the correct commands are, Set Date, set date, setdate, SetDate, once a correct command is entered then this method is run where the user is prompted to enter the years(0-24) month (1-12) and day (1-31) where anything outside of this range is given an error and prompts the user to try again until a correct time is an input. After that then the date is written to the computer's clock memory and overrides the previous date this is the one the user entered

Create PCB - Takes user input to create a PCB. The PCB name entered by the user is checked with pcb_find to see if it exists. If the name does not exist then a PCB is created with a call to pcb_setup.

Delete PCB - Uses pcb_find to locate an existing PCB then calls pcb_remove and pcb_free. This process will not be removed if it is a system process.

Block PCB - asks the user for process name then sets said process state_E to BLOCKED, then inserts the PCB into the blockedQueue and writes to the screen that the process is now blocked

Unblock PCB - asks the user for process name then sets said process state_E to READY, then inserts the PCB into the readyQueue. Then writes to the screen that the process is now ready.

Suspend PCB - asks user for process name, then checks if said process state is READY, changes the process's state_D to suspended. Then writes to the screen that the process is suspended.

Resume PCB - asks the user for a process name, checks if the process is suspended, and then sets the process's state_D to READY. Then writes to the screen that the process is resumed.

Set PCB Priority - asks the user for a process name and allows the user to change the priority (0-9) for said process. Command first checks that the process is not a system process because the user is not allowed to modify system processes.

Show PCB - asks the user for a process name and shows all data attributed to that process. The following is printed to the screen for said PCB: Process Name, Process Class, Execution State, Dispatching state, Process Priority

Show ready - Shows all processes in the ready queue, follows the same process of showing all data for each PCB in the readyQueue as in Show PCB

Show blocked - Shows all processes in the blocked queue, follows the same process of showing all data for each PCB in the blockedQueue as in Show PCB

Show All - Shows all created processes from both queues, readyQueue, and blockedQueue. Follows the same process of showing all data for each PCB in both queues as in Show PCB

R1 getSeconds()

int getSeconds(void)

Author: Connor White

This function gets the seconds from the internal clock register and records them as usable values.

Returns: unsigned char s

Usage Example:
int seconds = getSeconds();

R1 getMinutes()

int getMinutes(void)

Author: Connor White

This function gets the minutes from the internal clock register and records them as usable values.

Returns: unsigned char m

Usage Example:
int minutes = int getMinutes();

R1 getHours()

int getHours(void)

Author: Connor White

This function gets the hours from the internal clock register and records them as usable values.

Returns: unsigned char h

Usage Example:
int hours = int getHours();

R1 BcDtoDec(int BcD)

int BCDtoDec(int BCD)

Author: Connor White

This function converts the Binary-coded decimal into a normal decimal that is easy to read.

Returns: The integer after conversion

Usage Example:
int decimalValue = BCDtoDec(binaryCodedDecimal);

R1 getDay()

int getDay(void)
Author: Sam Holroyd

This function gets the value of the day stored in the internal clock register and records it as a usable value

Returns:

Usage Example:
int day = int getDay();

R1 getMonth()

int getMonth(void)

Author: Sam Holroyd

This function gets the value of the month stored in the internal clock register and records it as a usable value

Usage Example:
int month = int getMonth();

R1 getYear()

int getYear(void)

Author: Sam Holroyd

This function gets the value of the year stored in the internal clock register and records it as a usable value

Usage Example:
int year = int getYear();

R1 getTime()

void getTime(void)

Author: Connor White, brett

This function takes the seconds, minutes, and hours for the respective getter methods and then calls sys_req to output each for them to get the proper time

Usage Example:
Get time();

R1 getDate()

void getDate(void)

Author: Connor White, brett

This function gets the date from the internal clock register by calling getDay(void), getMonth(void), and getYear(void) and writes the returned values to the buffer

Usage Example:
Get date();

R1 setDate(int day, int month, int year)

void setDate(int day, int month, int year)

Author: Sam Holroyd

This function Sets the date in the internal clock register using valid inputs that are given by the user

Usage Example:
setDate(10, 2, 2024);

R1 setTime(int sec, int min, int hour)

void setTime(int sec, int min, int hour)

Author: Sam Holroyd, brett

This function Sets the time in the internal clock register using valid inputs that are given by the user

Usage Example:
setTime(30, 15, 10);

R1 serial_poll(device dev, char *buffer, size_t len)

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

Author: Sam Holroyd

This function polls the serial port for data until the buffer is full or the enter key is hit. Data is then processed in com_hand.c.

Usage Example:
```
char inputBuffer[256];
serial_poll(serialDevice, inputBuffer, sizeof(inputBuffer));
```

2. MPX PCB Functions

R2 pcb_allocate()

pcb* pcb_allocate(void)

Author: Sam Holroyd

This function allocates memory for a new PCB.

Returns: Pointer to the allocated PCB if successful, otherwise NULL.

Usage Example:
```
  pcb* new_pcb = pcb_allocate();
  if (new_pcb == NULL) {
      // Handle allocation failure
  }
```

R2 pcb_free()

int pcb_free(pcb* pcb)

Author: Sam Holroyd

This function frees memory allocated for a PCB.

Parameters: pcb* pcb: Pointer to the PCB to be freed.

Returns: 1 on error (if the provided PCB pointer is NULL) and 0 on success.

Usage Example:
```
        if (pcb_free(pcb_to_free) ==  1) {
         // Handle error
         }
```

R2 pcb_setup()

pcb* pcb_setup(const char* name, int class, int priority)

Author: Sam Holroyd

This function sets up a new PCB with the provided name, class, and priority.

Parameters: const char* name: Name of the process.

int class: Class of the process (0 or 1).

int priority: Priority of the process (0 to 9).

Returns: Pointer to the newly created PCB if successful, otherwise NULL.

Usage Example:
```
pcb* new_pcb = pcb_setup("Process1", 0, 5);
  if (new_pcb == NULL) {
      // Handle setup failure
  }
```

R2 pcb_find()

pcb* pcb_find(const char* name)

Author: Connor White, brett

This function finds a PCB with the provided name in the ready and blocked queues and can be used to look at a process to see if it is in the queue for testing or searching.

Parameters: const char* name: Name of the process to find.

Returns: Pointer to the found PCB if it exists, otherwise NULL.

Usage Example:
```
pcb* found_pcb = pcb_find("Process1");
 if (found_pcb == NULL) {
// Handle PCB   found
}
```

R2 pcb_insert()

void pcb_insert(pcb* new_pcb)

Author(s): Glen Mauder, Brett

This function inserts a PCB into the ready or blocked queue based on its state.
In addition, if the process is inserted into the ready queue, it is placed into the queue based on its priority.

Parameters: pcb* new_pcb: Pointer to the PCB to be inserted.

Usage Example:

```
pcb_insert(new_pcb);
```

R2 pcb_remove()

int pcb_remove(pcb* target_pcb)

Author(s): Glen Mauder, brett

This function removes a PCB from the system by first checking whether the state of the process is ready or blocked. If it is ready then it removes the process from the ready queue if it is blocked then it will remove the process from the blocked queue.

Parameters: pcb* target_pcb: Pointer to the PCB to be removed.

Returns: Returns 1 if the PCB was successfully removed, otherwise 0.

Usage Example:

```
 if (pcb_remove(target_pcb) == 0) {
     // Handle PCB   found or other error
 }
```

Helper Methods

R2 copy_string

Void copy_string(char *dest, const char *src, size_t n)

Author: Sam Holroyd

This helper method copies characters to the destination char array until either the end of the source char array is reached or ‘n’ characters have been copied

Parameters:

char *dest - A pointer the destination char array where the characters will be copied

const char *src - A pointer the source char array from which characters will be copied

size_t n - The maximum number of characters to copy

Usage Example:

copy_string(dest, src, sizeof(src);

2.1 MPX PCB Data Structures

PCB.h

This header file contains the data structures used for storing and manipulating PCBs. Enums are used to store the different states a PCB can be in. A struct is used to define the different parts of a PCB

Enums:

- Execution State

    pcb_state_E

        READY - 0

        BLOCKED - 1

- Dispatching State

    pcb_state_D

        NOT_SUSPENDED - 0

        SUSPENDED - 1

- PCB Class

    pcb_class

        SYSTEM - 0

        USER - 1

Struct PCB

Char Name - Name for a PCB

pcb_class CLASS - Class for a PCB. Uses SYSTEM or USER

Int Priority - Priority of the PCB stored as an int

pcb_state_E State_E - The execution state of the PCB. Uses READY or BLOCKED

pcb_state_D State_D - The dispatching state of the PCB. Uses NOT_SUSPENDED or SUSPENDED

Char Stack[1024] - Memory space for the CPU Stack

void* StackPtr - A pointer to the end of the stack

Struct pcb* next - a pointer to the next pcb in a queue

R2 Queue.h

This is the header file that contains everything related to the queue struct we stored the PCBs. It contains the interface for the queue struct, the queue_node struct, all helper methods associated with the queue system, and 2 already-defined queues for tracking ready and blocked PCBs.

R2 Struct Queue:

Author: Luke

Data Structure that manages PCB structs in a FILO (First in, Last out) order.

Contains:
- queue_node frontPtr
  - Used to store the first PCB in the queue, which is the latest PCB added
- queue_node rearPtr
  - Used to store the last PCB in the queue, which is the first PCB added
- int count
  - Used to store the number of PCBs in the queue

R2 Struct Queue_node:

Author: Luke

Data structure that contains the data stored for a singular PCB

Contains:
- pcb* data

Used to store the address to the *information* assigned to a PCB

Pointer points to the information below:

```
Name[9];
pcb_class Class;
 int Priority;
pcb_state_E State_E;
pcb_state_D State_D;
char Stack[1024];
void* StackPtr;
struct pcb* next;
struct queue_node* nextPtr
```

Helper Methods

R2 is_empty

int is_empty(queue* queuePtr):

Author: Luke

Function is_empty() checks to see if the queue given in parameter queuePtr contains any nodes

Parameters: queuePtr - determines which specific queue we are checking

Returns:

1 if param queuePtr is null or queue is empty
0 if queuePtr contains references to nodes

Usage Example:

```
if (is_empty(queuePtr)) {
        return NULL;
 }
```

R2 dequeue

pcb* dequeue(queue* queuePtr):

Author: Luke

Function dequeue - takes the first queue node out of the given queue, decrements the count of the queue, and moves the frontNode pointer of the queue struct to the next node in the queue.

Parameters: queuePtr - determines which specific queue we are modifying

Returns a pointer to the data in the node recently removed

Usage Example:

```
if(!strcmp(target_pcb->Name,tempNode->data->Name))
        dequeue(blockedQueue);
```

R2 enqueue

int enqueue(queue* queuePtr, pcb* data) :

Author: Luke

Function enqueue - Takes the given queue and creates a new node in it with the data given in the param pcb* data. Sets nextNode pointer to the front node then updates the front node to the newly added node

Parameters:

queuePtr - determines which specific queue we are modifying

Param: data - contains a pointer to the data we are adding to a new node

Returns:

1 if success
0 if fail

Usage Example:

```
    if(readyQueue->frontPtr->data->Priority > readyQueue->rearPtr->data->Priority)
{
    node* specTemp = readyQueue->frontPtr;
    dequeue(readyQueue);
    enqueue(readyQueue, specTemp->data);
}
```

R2 create_queue

queue* create_queue(void):

Author: Luke

Function create_queue - Allocates memory for a new queue
Sets frontPtr, rearPtr to Null and count to 0

Parameters: None

Returns: Pointer to newly created queue struct

Usage Example:

```
if(blockedQueue == NULL){
        blockedQueue = create_queue();
 }
```

R2 readyQueue:

Author: Luke

extern queue* readyQueue

Signifies the queue pointer readyQueue as a global variable

R2 blockedQueue:

Author: Luke

extern queue* blockedQueue

Signifies the queue pointer blockedQueue as a global variable