

## Programmer's Manual

### Assembly Description

#### **1.1) kernel/sys\_call\_isr.s**

- Pushes all general purpose registers and remaining segment registers onto the stack

### Header Description

#### **1.1) user/comhand.h**

- Functions for main command handler

##### **1.1.1) void comhand(void)**

- Handles all commands called by the user

##### **1.1.2) void r3\_load\_pcb(void (\*proc\_function)(void), char\* proc\_name, int proc\_priority)**

- Attempts to create a pcb for a given function.

### Functions

#### **1.2) user/time.h**

- Functions for getting and setting time

##### **1.2.1) void get\_time(void)**

- Displays the system time

##### **1.2.2) void set\_time(int hours, int minutes, int seconds)**

- Changes the system time to a time input by the user with the given parameters

### Functions

#### **1.3) user/date.h**

- Functions for getting and setting date

##### **1.3.1) int readDateReg(char sect)**

- File used for date related commands

##### **1.3.2) void get\_date(void)**

- Displays the system date

##### **1.3.3) void set\_date(int month, int day, int year)**

- Changes the system date to a date input by the user with the given parameters

### Functions

#### **1.4) user/rtc\_util.h**

- Functions for reading bytes and converting BCD

##### **1.4.1) int read\_rtc\_register(int reg)**

- Reads a byte from the RTC register

##### **1.4.2) int bcd\_to\_binary(int bcd)**

- Converts BCD to Binary

### Functions

#### **1.5/1.6) mpx/pcb.h**

- Functions used for pcb.c related commands

##### **1.5.1) struct pcb\* search\_queue(char\* to\_find, enum queue\_tag queue\_sel);**

- Searches for a PCB in a specific queue using its name

**1.5.2) void enqueue(pcb\* to\_add, enum queue\_tag queue\_sel);**

- Adds a PCB to a different queue

**1.5.3) int dequeue(pcb\* to\_remove, enum queue\_tag queue\_sel);**

- Removes a PCB from a queue.

**1.5.4) struct pcb\* pcb\_allocate(void)**

- Allocates memory for a new PCB

**1.5.5) int pcb\_free(struct pcb\* to\_freePtr)**

- Frees memory allocated to a PCB

**1.5.6) struct pcb\* pcb\_setup(char\* process\_name, class\_type process\_class, int process\_priority)**

- Initializes a PCB with its name, class and priority

**1.5.7) struct pcb\* pcb\_find(char\* to\_find)**

- Finds a PCB with the corresponding name

**1.5.8) int pcb\_insert(struct pcb\* to\_insertPtr)**

- Inserts a PCB into a queue

**1.5.9) int pcb\_remove(struct pcb\* to\_removePtr)**

- Removes a PCB from a queue.

**1.5.10) struct process\_queue\* get\_ready\_queue(void);**

- Returns a pointer to the ready queue

**1.5.11) struct process\_queue\* get\_blocked\_queue(void);**

- Returns a pointer to the blocked queue

**1.5.12) struct process\_queue\* get\_sus\_ready\_queue(void);**

- Returns a pointer to the suspended ready queue

**1.5.13) struct process\_queue\* get\_sus\_blocked\_queue(void);**

- Returns a pointer to the suspended blocked queue

**1.6.1) int delete\_pcb(void);**

- Deletes a PCB

**1.6.2) int block\_pcb(void);**

- Blocks a PCB

**1.6.3) int unblock\_pcb(void);**

- Unblocks a PCB

**1.6.4) int suspend\_pcb(void);**

- Suspends a PCB

**1.6.5) int resume\_pcb(void);**

- Resumes a PCB

**1.6.6) int set\_pcb\_priority(void);**

- Sets the priority of a PCB

**1.6.7) int show\_pcb(void);**

- Displays information about the PCB

**1.6.8) void show\_ready(void);**

- Displays information about the PCB in the ready queue

**1.6.9) void show\_blocked(void);**

- Display information about the PCB in the blocked queue

#### **1.6.10) void show\_all(void);**

- Display information about all PCB

### **Function Description**

#### **1.1) serial.c**

##### **1.1.1) static int serial\_devno(device dev)**

**Author:** Garret Butler

**Parameters:** device dev

##### **1.1.2) serial\_poll(device dev, char \*buffer, size\_t len)**

**Author:** Garret Butler, Connor Groizard

**Parameters:** device dev, char\* buffer, size\_t len

**Return:** buffer size after user input

-The function is responsible for continuously reading characters from COM1.

It does this by processing each character according to different cases. There are many cases such as backspaces, spaces, escape character, delete key amongst others.

**Error:** None

#### **1.2) comhand.c**

##### **1.2.1) void getval(char\* resource)**

**Author:** Rama Al-Omar, Malone Ingham, Connor Groizard

**Parameters:** char\* resource

**Return:** None

- Retrieves and displays values based on the provided resource ("date" or "time").
- Calls corresponding functions to display the requested information.
  - get\_date()
  - get\_time()
- The function will display the time in a hh:mm:ss format
- The function will display the date in a mm/dd/yyyy format

**Error:** None

##### **1.2.2) int is\_number(const char\* str)**

**Author:** Connor Groizard

**Parameters:** const char\* str

**Return:** Int value, 1 for all characters are numerical digits, 0 if at least 1 character is not a numerical digit

- Checks if a given string consists of numeric digits only.
- Iterates through each character in the string and validates if its a digit.

**Error:**

- If the input contains a character that is not a numeric digit. Return 0.

**1.2.3) void setval(char\* resource)**

**Author:** Malone Ingham, Connor Groizard

**Parameters:** char\* resource

**Return:** 0 for success, -1 for errors

- Sets values based on the provided resource ("date" or "time").
  - set\_date()
  - set\_time()
- Reads user input, validates and processes it accordingly.
- The function will set the date in a mm/dd/yyyy format
- The function will set the time in a hh/mm/ss format

**Error:**

- If input can't be read properly: "Error reading input" return -1
- If format is invalid for date: "Invalid format. Use 'mm dd yyyy' " return -1
- If format is invalid for time: "Invalid format. Use 'hh mm ss' " return -1
- If request is invalid: "Invalid request" return -1

**1.2.4) void clear(device dev)**

**Author:** Rama Al-Omar

**Parameters:** device dev

**Return:** None

- Clears the terminal by blanking it

**Error:** None

**1.2.5) void version(void)**

**Author:** Rama Al-Omar

**Parameters:** None

**Return:** None

- Prints the version of the MPX, as well as a compilation date.

**Error:** None

**1.2.6) void help(void)**

**Author:** Rama Al-Omar

**Parameters:** None

**Return:** None

- Prints all available commands as well as a description for the user.

**Error:** None

**1.2.7) int shutdown(void)**

**Author:** Rama Al-Omar

**Parameters:** None

**Return:** Int value, 1 for confirmation for shutdown, 0 for cancelation of the shutdown

- Shuts down the MPX after a YES or NO question to the user.

**Error:** None

### **1.2.8) void writeNewLine(void)**

**Author:** Rama Al-Omar

**Parameters:** None

**Return:** None

- Writes a new line to ensure consistent formatting.

**Error:** None

### **1.2.9) void comhand(void)**

**Author:** Rama Al-Omar, Malone Ingham

**Parameters:** None

**Return:** None

- Takes in user input and evaluates what command it is, then executes the command by calling the relevant function. Also, prints the welcome message to the MPX in color.

**Error:**

- If command is not entered properly: "Improper command entered. Please try again. Ensure that the command is listed and in lowercase.\n"

### **1.2.10) void yield(void)**

**Author:** Rama Al-Omar

**Parameters:** None

**Return:** None

- Makes the Command Handler yield the CPU and finish any processes left in the queue.

**Error:** None

### **1.2.11) void load\_r3(void)**

**Author:** Rama Al-Omar

**Parameters:** None

**Return:** None

- Loads the test processes in R3 and initializes and saves the context at the top of the stack.

**Error:** None

### **1.2.12) void r3\_load\_pcb(void)**

**Author:** Rama Al-Omar

**Parameters:** None

**Return:** None

- Attempts to create a pcb and sets registers for the stack starting with ESP and ending with EBP.

**Error:** None

### **1.2.13) void load\_r3\_suspended(void)**

**Author:** Rama Al-Omar

**Parameters:** None

**Return:** None

- Loads the processes in a suspended state for R3 and initializes/saves the context at the top of the stack.

**Error:** None

## **1.3) time.c**

### **1.3.1) int readTimeReg(char sect)**

**Author:** Connor Groizard

**Parameters:** char sect

**Return:** Binary representation of the time

- Reads and retrieves specific time information from the RTC(Real-Time-Clock) registers.

**Error:**

-If there's an invalid sect character. Return -1

### **1.3.2) void get\_time(void)**

**Author:** Connor Groizard

**Parameters:** None

**Return:** None

- Gets the current time from the RTC(Real-Time-Clock) registers, then formats it as a string and is then written to the terminal.

**Error:** None

### **1.3.3) void set\_time(int hours, int minutes, int seconds)**

**Author:** Connor Groizard

**Parameters:** int hours, int minutes, int seconds

**Return:** None

- Sets the time in the RTC(Real-Time-Clock) registers, based on the values for Hours, Minutes and Seconds. Has data validation to make sure that the hours can only be in range 0-23, minutes 0-59 and seconds 0-59.

**Error:**

-If hours are not in valid range: "Error: Invalid hours. Hours should be in the range 0-23.\n"

-If minutes are not in valid range: "Error: Invalid minutes. Minutes should be in the range 0-59.\n"

-If seconds is not in valid range: "Error: Invalid seconds. Seconds should be in the range 0-59.\n"

## **1.4) date.c**

#### **1.4.1) int read\_rtc\_register(int reg)**

**Author:** Connor Groizard, Malone Ingham

**Parameters:** int reg

**Return:** Byte from specified register

- Reads a byte of data from a specific register in the RTC(Real-Time-Clock).
- outb() RTC register selected
- inb() RTC data port

**Error:** None

#### **1.4.2) int bcd\_to\_binary(int bcd)**

**Author:** Connor Groizard, Malone Ingham

**Parameters:** int bcd

**Return:** Binary representation of the BCD value

- Converts Binary-Coded Decimal(BCD) value into the corresponding Binary equivalent.

**Error:** None

#### **1.4.3) int binary\_to\_bcd(int bcd)**

**Author:** Connor Groizard, Malone Ingham

**Parameters:** int bcd

**Return:** BCD representation of the Binary value

- Converts Binary to BCD(Binary-Coded-Decimal). Responsible for correctly representing the years in the set date.

**Error:** None

#### **1.4.4) int days\_in\_month(int month, int year)**

**Author:** Connor Groizard

**Parameters:** int month, int year

**Return:** Number of days in a specified month of a specified year.

- Function to calculate number of days in a given month of a year. It does this by implementing a switch statement for different cases for different months. It also checks to see whether February is a leap year or not.

**Error:** None

#### **1.4.5) int readDateReg(char sect)**

**Author:** Connor Groizard, Malone Ingham

**Parameters:** char sect

**Return:** Binary representation of a specific date component

- Reads and retrieves specific time information from the RTC(Real-Time-Clock) registers.

**Error:**

- If there's an invalid sect character. Return -1

#### **1.4.6) void get\_date(void)**

**Author:** Malone Ingham, Connor Groizard

**Parameters:** None

**Return:** None

- Gets the current date from the RTC(Real-Time-Clock) registers, then formats it as a string and is then written to the terminal.

**Error:** None

#### **1.4.7) void set\_date(int month, int day, int year)**

**Author:** Malone Ingham, Connor Groizard

**Parameters:** int month, int day, int year

**Return:** None

- Sets the time in the RTC(Real-Time-Clock) registers, based on the values for Month,Day and Year. It has data validation to make sure that the month is in Range 1-12. It also has data validation to make sure that the day selected corresponds accordingly to the correct month so that the day cant be less than 1 or greater than the max amount of days in that month.

**Error:**

- If month is not in range:"Error: Invalid month.\n"
- If day is not in range:"Error: Invalid day for this month.\n"

### **1.5) rtc\_util.c**

#### **1.5.1) int read\_rtc\_register(int reg)**

**Author:** Connor Groizard, Malone Ingham

**Parameters:** int reg

**Return:** Byte from specified register

- Reads a byte of data from a specific register in the RTC(Real-Time-Clock).
  - outb() RTC register selected
  - inb() RTC data port-bash: ./mpx.sh: /bin/sh^M: bad interpreter: No such file or directory

**Error:** None

#### **1.5.2) int bcd\_to\_binary(int bcd)**

**Author:** Connor Groizard, Malone Ingham

**Parameters:** int bcd

**Return:** Binary representation of BCD.

- Converts Binary-Coded Decimal(BCD) value into the corresponding Binary equivalent.

**Error:** None

### **1.6) pcb.c**

#### **1.6.1) struct pcb\* search\_queue(char \* to\_find, enum queue\_tag queue\_sel)**

**Author:** Garret Butler

**Parameters:** char\* to\_find, queue\_tag queue\_sel



**Return:** pointer to pcb or NULL

-The function searches for a pcb(process control block) with a specific name in a specific queue. It then goes through the queue comparing the names of processes until it finds the correct one or the queue ends.

**Error:** None

### **1.6.2) void enqueue(pcb\* to\_add, enum queue\_tag queue\_sel)**

**Author:** Garret Butler

**Parameters:** pcb\* to\_add, enum queue\_tag queue\_sel

**Return:** Nothing

-The function adds a pcb(process control block) to a specific queue.

If it's a ready queue, meaning priority matters, then it will be inserted according to its specific priority. So that high priority PCBs are placed before the lower priority ones. If its a blocked queue, then it will add the pcb to the end.

**Error:** None

### **1.6.3) int dequeue(pcb\* to\_remove, enum queue\_tag queue\_sel)**

**Author:** Garret Butler

**Parameters:** pcb\* to\_remove, enum queue\_tag queue\_sel

**Return:** 0 as success, 1 as failure

-The function removes a pcb(process control block) from a specific queue.

It does this by checking if the pcb is in a queue, then removes it by modifying the pointers.

If the pcb is the only thing in the queue, then it will just clear the queue.

If the pcb is at the head/tail of the queue, then it will just adjust the pointers.

**Error:**

- If there is an invalid queue selection, Error:"Dequeue: Invalid queue selection". Return 1
- If the PCB is not in queue, Error:"Dequeue: PCB not in the queue". Return 1

### **1.6.4) struct pcb\* pcb\_allocate(void)**

**Author:** Garret Butler

**Parameters:** None

**Return:** Pointer to a pcb structure

-The function is responsible for creating and initializing a new pcb (process control block). It does so by allocating memory for the pcb, Then it sets all the attributes to default values(process\_name, process\_class, process\_priority, exe\_state, disp\_state, pcb\_stack, stackPtr, next\_pcbPtr and prev\_pcbPtr. Then it just returns a pointer for the new pcb.

**Error:** None

### **1.6.5) int pcb\_free(struct pcb\* to\_freePtr)**

**Author:** Garret Butler

**Parameter:** struct pcb\* to\_freePtr

**Return:** Int value for status of the memory deallocation

-The function is responsible for deallocating memory from a specific pcb (process control block).

**Error:** None

#### **1.6.6) struct pcb\* pcb\_setup(char\* process\_name, class\_type process\_class, int process\_priority)**

**Author:** Garret Butler

**Parameters:** char\* process\_name, class\_type process\_class, int process\_priority

**Return:** Pointer to a struct pcb

-The function is responsible for setting up a pcb(process control block) with its specific attributes (process\_name, process\_class, process\_priority). It also checks to see if the priority is in legal range 0-9 and that its not 0 when process class = SYSTEM.

**Error:** None

#### **1.6.7) struct pcb\* pcb\_find(char\* to\_find)**

**Author:** Garret Butler

**Parameter:** char\* to\_find

**Return:** Pointer to a struct pcb

-The function is responsible for searching a pcb(process control block) with a specific name in 4 different queues. It then goes through the 4 different Queues and uses a function mentioned previously, the search\_queue function to find the corresponding pcb. Then if the pcb is found, it will return a pointer to that specific pcb, if it isn't found in any of the 4 queues it returns a NULL.

**Error:** None

#### **1.6.8) int pcb\_insert(struct pcb\* to\_insertPtr)**

**Author:** Garret Butler

**Parameter:** struct pcb\* to\_insertPtr

**Return:** Int value, 0 for success, 1 for NULL

-This function is responsible for inserting a specific pcb(process control block) into the correct queue based on its execution state(exe\_state) and dispatch state(dispatch\_state). It does this by checking if the pcb is acceptable. Then, by combining the execution state and dispatch state it checks what queue to add the pcb into. Then it just calls a function mentioned previously, enqueue to add it into the appropriate queue.

**Error:** None

#### **1.6.9) int pcb\_remove(struct pcb\* to\_removePtr)**

**Author:** Garret Butler

**Parameter:** struct pcb\* to\_removePtr

**Return:** Int value, 0 for success, 1 for NULL/fail

-This function is responsible for removing a pcb(process control block) From a specific queue based on the dispatch state(dispatch\_state) and execution state(exe\_state). It does this, similarly to the pcb\_insert function by first checking if the pcb is acceptable, then it determines the queue where it has to be removed from.

**Error:** None

#### **1.6.10) struct process\_queue\* get\_ready\_queue()**

**Author:** Garret Butler

**Parameter:** None

**Return:** Pointer to ready\_queue

- This function returns a pointer to the ready\_queue

**Error:** None

#### **1.6.11) struct process\_queue\* get\_blocked\_queue()**

**Author:** Garret Butler

**Parameter:** None

**Return:** Pointer to blocked\_queue

- This function returns a pointer to the blocked\_queue

**Error:** None

#### **1.6.12) struct process\_queue\* get\_sus\_ready\_queue()**

**Author:** Garret Butler

**Parameter:** None

**Return:** Pointer to sus\_ready\_queue

- This function returns a pointer to the sus\_ready\_queue

**Error:** None

#### **1.6.13) struct process\_queue\* get\_sus\_blocked\_queue()**

**Author:** Garret Butler

**Parameter:** None

**Return:** Pointer to sus\_blocked\_queue

- This function returns a pointer to the sus\_blocked\_queue

**Error:** None

### **1.7)user/pcb.c**

#### **1.7.1) void log\_info(char \*message)**

**Author:** Rama Al-Omar

**Parameters:** char\* message

**Return:** None

-The function is responsible for logging a message to the terminal.

**Error:** None

### **1.7.2) int delete\_pcb(void)**

**Author:** Rama Al-Omar

**Parameters:** None

**Return:** Int value, 0 for success,

- 1 for error(attempting to delete a system process),
- 2 for error(attempting to delete a non-existent pcb),
- 2 for error(pcb could not be removed from queue)

- The function will prompt the user to write the name of what pcb(process control block) he wants to delete. Then it will find the name of said PCB and check if it exists and if it's a system process. It will then remove it from the queue and the memory associated with it is now freed. It also has different error messages according to the specific error.

#### **Errors:**

- If the process class is a system: "Error: System processes cannot be deleted." Return -1
- If PCB cannot be removed from the queue: "Error: System processes cannot be deleted." Return -2
- If the PCB with the given name does not exist: "Error: The PCB you are trying to delete does not exist." Return -3

### **1.7.3) int block\_pcb(void)**

**Author:** Rama Al-Omar

**Parameters:** None

**Return:** Int value, 0 for success,

- 1 for error(attempting to block a pcb/ manipulate system processes),
- 1 for error(pcb to block does not exist/already blocked),
- 2 for error(pcb could not be removed from queue),
- 3 for error(pcb could not be inserted into the queue)
- 4 for error for non-existent pcb

- The function will prompt the user to write the name of what pcb(process control block) they want to block. It will then find the name of said pcb and check if it exists and if it hasn't been blocked yet. In the case it isn't blocked, it will then block the pcb by changing the dispatch state as well as the execution state and insert it into the correct queue. It also has different error messages according to the specific error.

#### **Errors:**

- If PCB's class is a system: "Error: System PCBs cannot be blocked manually." Return -1
- If PCB cannot be removed from the queue: "Error: PCB could not be removed from queue." Return -2

- If PCB cannot be inserted into the queue: "Error: PCB could not be inserted into queue." Return -3
- If the PCB with the given name does not exist: "Error: The PCB you are trying to block does not exist or is already blocked." Return -4

-

#### **1.7.4) int unblock\_pcb(void)**

**Author:** Rama Al-Omar

**Parameters:** None

**Return:** Int value, 0 for success,

- 1 for error(pcb to unblock does not exist/already unblocked),
- 2 for error(pcb could not be removed from queue),
- 3 for error(pcb could not be inserted into the queue)

- The function will prompt the user to write the name of what pcb(process control block) they want to unblock. It will then find the name of said pcb and check if it exists and is blocked. It will then unblock said pcb by changing the dispatch state as well as the execution state and insert it into the ready queue. There's also error messages according to the specific error.

#### **Errors:**

- If PCB cannot be removed from the queue: "Error: PCB could not be removed from queue." Return -1
- If PCB cannot be inserted into the queue: "Error: PCB could not be inserted into queue." Return -2
- If the PCB with the given name does not exist: "Error: The PCB you are trying to unblock does not exist or is not blocked." Return -3

#### **1.7.5) int suspend\_pcb(void)**

**Author:** Rama Al-Omar

**Parameters:** None

**Return:** Int value, 0 for success,

- 1 for error(attempting to suspend a system process),
- 2 for error(pcb to suspend does not exist/ is already suspended),
- 2 for error(pcb could not be removed from queue),
- 3 for error(pcb could not be inserted into the queue)

- The function will prompt the user to write the name of what pcb(process control block) they want to suspend. It will then find the name of said pcb and check if it exists and is not suspended. It will then suspend said pcb by changing the dispatch state as well as the execution state and insert it into the corresponding queue. There's also error messages according to the specific error.

#### **Errors:**

- If PCB's class is system: "Error: System processes cannot be suspended." Return -1
- If PCB cannot be removed from the queue: "Error: PCB could not be removed from queue." Return -2

- If PCB cannot be inserted into the queue: "Error: PCB could not be inserted into queue." Return -3
- If the PCB with the given name does not exist: "Error: The PCB you are trying to suspend does not exist or is not suspended." Return -4

#### **1.7.6) int resume\_pcb(void)**

**Author:** Rama Al-Omar

**Parameters:**None

**Return:**Int value, 0 for success,

- 1 for error(pcb to resume does not exist/ is not suspended),
- 2 for error(pcb could not be removed from queue),
- 3 for error(pcb could not be inserted into the queue)

-The function will prompt the user to write the name of what pcb(process control block) they want to resume. It will then find the name of said pcb and check if it exists and is suspended. It will then resume said pcb by changing the dispatch state as well as the execution state and insert it into the corresponding queue.

There's also error messages according to the specific error.

#### **Errors:**

- If PCB cannot be removed from the queue: "Error: PCB could not be removed from queue." Return -1
- If PCB cannot be inserted into the queue: "Error: PCB could not be inserted into queue." Return -2
- If the PCB with the given name does not exist: "Error: The PCB you are trying to resume does not exist or is not suspended." Return -3

#### **1.7.7) int set\_pcb\_priority(void)**

**Author:** Rama Al-Omar

**Parameters:**None

**Return:**Int value, 0 for success,

- 1 for error(invalid priority),
- 2 for error(non-existent pcb)

-The function will prompt the user to write the name of what pcb(process control block) they want and for the priority. It will then find the name of said pcb and check if it exists and validates the imputed priority. It will then set the new priority for the pcb and requeue it. There's also error messages according to the specific error.

#### **Errors:**

- If priority is invalid: "Error: Priority must be between 0 and 9. Priority can only be 0 for system processes." Return -1
- If the PCB with the given name does not exist: "Error: The PCB you are trying to set the priority for does not exist." Return -2

#### **1.7.8) int show\_pcb(void)**

**Author:** Connor Groizard, Malone Ingham

**Parameters:**None

**Return:**Int value, 0 for success

-1 for error(when process not found)

-The function will prompt the user to write the name of what pcb(process control block) they want to show. It will then find the name of said pcb and display the information related to it(name, class, priority, state and suspend state).

There's also an error message if the pcb is not found.

**Errors:**

- If PCB with the given name does not exist: "Error: Process not found." Return -1

### **1.7.9) void show\_ready(void)**

**Author:** Connor Groizard, Malone Ingham

**Parameters:** None

**Return:** None

-The function will display all the PCBs that are in a ready state. It will then get the pointer to the ready queue and traverse the ready queue displaying the pcb information, including name, priority, state and suspended state.

There's also a message at the end showing a list of ready PCBs.

**Error:** None

### **1.7.10) void show\_blocked(void)**

**Author:** Connor Groizard, Malone Ingham

**Parameters:** None

**Return:** None

-The function will display all PCBs that are in a blocked state. It will then get the pointer to the blocked queue and traverse the blocked queue displaying the pcb information, including name, priority, state and suspended state.

There's also a message at the end showing a list of blocked PCBs.

**Error:** None

### **1.7.11) void show\_all(void)**

**Author:** Connor Groizard, Malone Ingham

**Parameters:** None

**Return:** None

-The function will display all PCBs not caring about their state. It will then get the pointers(ready, blocked, suspended ready, suspended blocked) and traverse each queue displaying the information(name, class, priority, state and suspended state for every pcb in the queue.

**Error:** None

## **1.8)sys call.c**

### **1.8.1) struct context\* sys\_call(struct context \*current\_context)**

**Author:** Garrett Butler, Malone Ingham

**Parameters:** struct context \*current\_context

**Return:** Int value, 0 for success,

-Takes the parameter that is a pointer to a struct that represents the context of the current process and returns the pointer to the context of the process being currently loaded. Two system call operations are used (EXIT & IDLE) to decide to delete the running PCB or to save the context of the current PCB.

**Error:**

-If the operation code is anything other than EXIT or IDLE, it sets the return value to -1 and doesn't load any new context.

-If there is an error freeing memory, a value of -1 is returned.