

# Programmer's Manual

R3/4

10/21/2021

By:

Will Ferrell, Jarett Hardek, Anthony Lowery, Emily Waechter

# Table of Contents

<b>void version():</b>	<b>3</b>
<b>void help():</b>	<b>3</b>
<b>int shutdown():</b>	<b>3</b>
<b>int comhand():</b>	<b>4</b>
<b>int print(char* str):</b>	<b>4</b>
<b>int println(char* str):</b>	<b>4</b>
<b>int *polling(char *buffer, int *count):</b>	<b>4</b>
<b>int gettime():</b>	<b>5</b>
<b>int settime(char *time):</b>	<b>5</b>
<b>int check_time_str(char* time_str):</b>	<b>6</b>
<b>int getdate():</b>	<b>6</b>
<b>int setdate():</b>	<b>6</b>
<b>int checkDate(char* date):</b>	<b>6</b>
<b>int intToBCD(int val):</b>	<b>7</b>
<b>int BCDToInt(int val):</b>	<b>7</b>
<b>int BCDtoStr(int val, char *str):</b>	<b>7</b>
<b>int StrtoBCD(int val, char *str):</b>	<b>7</b>
<b>void fifo_enqueue(queue *q, PCB *pcb):</b>	<b>8</b>
<b>void priority_enqueue(queue *q, PCB *pcb):</b>	<b>8</b>
<b>PCB* AllocatePCB():</b>	<b>8</b>
<b>void FreePCB():</b>	<b>8</b>
<b>PCB* FindPCB(char* processName):</b>	<b>9</b>
<b>void InsertPCB(PCB *pcb):</b>	<b>9</b>
<b>void RemovePCB():</b>	<b>10</b>
<b>void CreatePCB(char* processName, unsigned char processClass, int priority):</b>	<b>10</b>
<b>void DeletePCB(char* processName):</b>	<b>10</b>

<b>void BlockPCB(char* processName):</b>	<b>10</b>
<b>void UnblockPCB(char* processName):</b>	<b>11</b>
<b>void SuspendPCB(char* processName):</b>	<b>11</b>
<b>void ResumePCB(char* processName):</b>	<b>11</b>
<b>void SetPCBPRIORITY(char* processName, int priority):</b>	<b>11</b>
<b>void ShowPCB(char* processName):</b>	<b>12</b>
<b>void ShowReady():</b>	<b>12</b>
<b>void ShowBlocked():</b>	<b>13</b>
<b>void ShowAll():</b>	<b>13</b>
<b>sys_call_isr():</b>	<b>13</b>
<b>sys_call():</b>	<b>13</b>
<b>void yield():</b>	<b>13</b>
<b>void loadr3():</b>	<b>14</b>
<b>void createAlarmProcess():</b>	<b>14</b>
<b>void addAlarm(char* message, int alarmTime):</b>	<b>14</b>

## **void version():**

- Prints the current version of MPX and its completion date
- Void function, prints the version and date to the terminal

## **void help():**

- Provides usage instructions for each command available to the user
- Void function, prints each command on a new line

## **int shutdown():**

- Exits command handler loop, shuts down the MPX machine
- Execution returns to kmain()
- Requires confirmation by users

**Return** - (integer): that is assigned to “quit”, the main command handler control loop variable

## **int comhand():**

- Collects input from the user and validates that it is valid
- Controls the design of the user interface
- Menu driven, listing each command implemented in the MPX

**Return** - (integer-pointer): arbitrarily returns 0 to terminate the command handler

## **int print(char\* str):**

- Prints a string to the terminal

**Parameter** - str (char pointer)

**Return** - (integer): arbitrarily returns 0 to terminate the command handler

## **int println(char\* str):**

- Prints a string on a new line to the terminal

**Parameter** - str (char pointer)

**Return** - (integer): arbitrarily returns 0 to terminate the command handler

### **int \*polling(char \*buffer, int \*count):**

-Gathers keyboard input from the user and passes it into the command buffer which is stored as a char pointer

-It validates each keystroke including all alphanumeric characters, backslash, forward slash, comma, and period. It also handles special keys such as backspace, delete, the home and end key, and the arrow keys

**Parameter** - buffer (char-pointer): the data that is stored in the command buffer (usually empty)

**Parameter** - count (integer-pointer): the value of the size of the buffer (this is the value that remains unchanged and returns)

**Return** - (integer-pointer): the value of the size of the buffer

### **int gettime():**

-Prints the time that was accessed through the outb function calls bcdtoStr on the time to convert from BCD to a char and place it in the time char array

**Return** - (integer): arbitrarily returns 0

### **int settime(char \*time):**

-The time has already been inputted after printing a prompt in the command handler

-Calls check\_time\_str to make sure the time entered was valid  
sets the time using the outB and StrtoBCD function to convert the char to a BCD

**Parameter** - time (char pointer)

**Return** - (integer): 0 if time format is valid or 1 if the time is invalid

### **int check\_time\_str(char\* time\_str):**

-Makes sure the time is of valid length and valid format in military time  
-If 0 is returned the settime function continues if 1 is returned a invalid format message is returned

**Parameter** - time\_str (char pointer)

**Return** - (integer): 0 if time format is valid or 1 if the time is invalid

### **int getdate():**

-Prints the date that was accessed through the outb function  
-Calls bcdtoStr on the date to convert from BCD to a char and place it in the date char array

**Return** - (integer): arbitrarily returns 0

### **int setdate():**

-Reads in the date entered by the user after printing a prompt  
-Calls checkDate to make sure the date entered was valid  
-Sets the date using the outB and StrtoBCD function to convert the char to a BCD

**Return** - (integer): arbitrarily returns 0

### **int checkDate(char\* date):**

-Makes sure the date is valid

-If 0 is returned the setdate function continues if 1 is returned a invalid format message is returned

**Parameter** - date (char pointer)

**Return** - (integer): 0 if time format is valid or 1 if the time is invalid

### **int intToBCD(int val):**

-Converts an int value to a BCD

**Parameter** - val (integer)

**Return** - a BCD

### **int BCDToInt(int val):**

-Converts a BCD to an int value

**Parameter** - val (integer)

**Return** - an int

**int BCDtoStr(int val, char \*str):**

**Parameter** - val (integer): BCD value

**Parameter** - str (char array pointer)

**Return** - 0 but puts the BCD into the char array pointer

**int StrtoBCD(int val, char \*str):**

-Converts a char pointer to binary coded decimal

**Parameter** - val (integer): BCD value

**Parameter** - str (char array pointer)

**Return** - the BCD of the number that was in the char array

**void fifo\_enqueue(queue \*q, PCB \*pcb):**

-puts a new pcb into a fifo queue

**Parameter** - q (queue pointer)

**Parameter** - pcb (PCB- process control block pointer)

**void priority\_enqueue(queue \*q, PCB \*pcb):**

-puts a new pcb into a priority queue

**Parameter** - q (queue pointer)

**Parameter** - pcb (PCB- process control block pointer)



## **PCB\* AllocatePCB():**

-allocates memory for a new PCB, and performs initialization

**Return** - PCB Pointer if successful, or NULL if an error occurs during allocation

## **void FreePCB():**

-frees all memory for a given PCB

**Return** - Success or Error Code

## **PCB\* SetupPCB(char\* processName, unsigned char processClass, int priority):**

-calls AllocatePCB() to create an empty pcb, performs initialization, and sets it to ready not suspended

**Parameter** - processName (char pointer)

**Parameter** - processClass (unsigned char)

**Parameter** - priority (integer)

**Return** - PCB Pointer if successful, or NULL if an error occurs

**PCB\* FindPCB(char\* processName):**

-searches all queues for a pcb with a given process name

**Parameter** - processName (char pointer)

**Return** - PCB Pointer if successful, or NULL if an error occurs

**void InsertPCB(PCB \*pcb):**

-inserts a PCB into the appropriate queue

**Parameter** - pcb (PCB pointer)

**void RemovePCB():**

-removes a PCB from its current queue

**Parameter** - pcb (PCB pointer)

**void CreatePCB(char\* processName, unsigned char processClass, int priority):**

-calls SetupPCB() and inserts the PCB into the appropriate queue

**Parameter** - processName (char pointer)

**Parameter** - processClass (unsigned char)

**Parameter** - priority (integer)

**void DeletePCB(char\* processName):**

-removes the pcb from the appropriate queue, and free all associated memory

**Parameter** - processName (char pointer)

**void BlockPCB(char\* processName):**

-Finds the given PCB, sets it to blocked, then reinserts it into its queue

**Parameter** - processName (char pointer)

**void UnblockPCB(char\* processName):**

-Finds the given PCB, sets it to unblocked, then reinserts it into its queue

**Parameter** - processName (char pointer)

**void SuspendPCB(char\* processName):**

-Finds the given PCB, sets it to suspended, then reinserts it into its queue

**Parameter** - processName (char pointer)

**void ResumePCB(char\* processName):**

-Finds the given PCB, sets it to not suspended, then reinserts it into its queue

**Parameter** - processName (char pointer)

**void SetPCBPriority(char\* processName, int priority):**

-sets the priority of the pcb

**Parameter** - processName (char pointer)

**Parameter** - priority (integer)

**void ShowPCB(char\* processName):**

-Displays the process name, class, state, suspended status and priority of all PCB's in the ready queue

**Parameter** - processName (char pointer)

**void ShowReady():**

-Displays process name, class, state, suspended status, and priority

## **void ShowBlocked():**

-Displays the process name, class, state, suspended status and priority of all PCB's in the blocked queue

## **void ShowAll():**

-Displays the process name, class, state, suspended status and priority of all PCB's in both the ready and blocked queues

## **sys\_call\_isr():**

-the system call interrupt service routine is written in assembly language. This pushes registers to the stack, calls sys\_call, and pops the registers from the stack

## **sys\_call(context\* register):**

-Prepares MPX for the next ready process to begin/resume execution.

**Parameter** - register (context pointer)

## **void yield():**

-Causes the command handler to yield to other processes, if any processes are in the ready queue.

**void loadr3():**

-Loads all r3 processes into memory in a suspended ready state at any priority.

**void createAlarmProcess():**

-Creates a PCB for the alarm.

**void addAlarm(char\* message, int alarmTime):**

-Creates a new alarm that contains a specified message and outputs the message at a given time

**Parameter** - message (char pointer)

**Parameter** - alarmTime (char pointer)

