nexTTeam mpx Module 2

Programmer's Manual

Team Members

Teddy Hale
Jessica Hammersla
Mya Junkin
Philip Wenger

Table of Contents

iii. setupPCB
iv. removePCB
v. setPriority
vi. findPCB
vii. showPCB
viii. showReady
ix. showBlocked
x. showAllProcesses
xi. printOnePCB
xii. block
xiii. unblock
xiv. freePCB
xv. suspend
xvi. resume

Module 1

commandhandler.h

```
// Displays the current version being used for the MPX
void version();

// Displays instructions for how to use each command
void help();

// Shutdown the MPX and terminate
int shutdown();

// Prompts the user for time input and sets the time
void settimeWrapper();

// Prompts the user for date input and sets the date
void setdateWrapper();

// Processes user input and find the correct function based off user request
int comhand();
```

date.h

```
// Get the date from the registers.  Display the date.
void getdate();


// Set the date in the registers. User supplies the month, day, and year to set.
void setDate(int month, int day, int year);
```

polling.h

```
// Will accept user input.  Processes input and displays the buffer.
int poll(char * buffer, int * count);
```

time.h

```
// Will get the time.  Displays the time.
void gettime();

// Accepts an unsigned char and converts from BCD to an integer
int bcdToInt(unsigned char value);

// Sets the time based off user input.
void setTime(int hours, int minutes, int seconds);

// Accepts an integer and converts from an integer to a BCD
unsigned char intToBcd(int data);

// Accepts an integer and converts from an integer to ASCII
char * intToAscii(int integer);
```

Module 2

commandhandler.h

```
// Allows the user to create a process
void createPCBWrapper();

// Allows the user to remove a process
void deletePCBWrapper();

// Prompts the user for the name and priority value
void setPriorityWrapper();

// Prompts the user for a PCB name
void showPCBWrapper();

// Prompts the user for a PCB name (and finds that pcb to verify its valid)
void blockWrapper();

// Prompts the user for a PCB name (and finds that pcb to verify its valid)
void unblockWrapper();

// Displays the previous ten commands used by the user
void history();

// Prompts the user for a PCB name than sets it to suspended
void suspendWrapper();

// Prompts the user for a PCB name than sets it to not suspended
void resumeWrapper();
```

pcb.h

```
// Inserts PCB into the correct queue.
void insertPCB(pcb* Pcb);
```

```
// Allocates space for the PCB.
pcb* allocatePCB();

// Set up the values for a PCB.
pcb* setupPCB(char *name, int classCode, int priorityCode);

// Remove the PCB from a queue.
int removePCB(pcb* process);

// Change the priority for a PCB and move into the correct queue.
void setPriority(char *name, int priorityNum);

// Finds a PCB with the specified name.  Return a pointer to the PCB if found.
pcb* findPCB(char *PcbName);

// Show the PCB with the specified name. Display name, state, if blocked, and priority.
void showPCB(char *name);

// Show all PCBs in ready queue.  Display name, state, if blocked, and priority.
void showReady();

// Show all PCBs in blocked queue. Display name, state, if blocked, and priority.
void showBlocked();

// Show all PCBs. Display name, state, if blocked, and priority.
void showAllProcesses();

// Print one PCB with a specified name. Display name, state, if blocked, and priority.
void printOnePCB(pcb* Pcb);

// Finds a PCB sets it's state to blocked and reinserts it into the right queue
void block(pcb* PCB);
```

```
// Finds a PCB sets it's state to unblocked and reinserts it into the right queue
void unblock(pcb* PCB);

// Frees space for the PCB
int freePCB(pcb* PCB);

// Frees space for the PCB
void suspend(pcb* PCB);

// Frees space for the PCB
void resume(pcb* PCB);
```