

Four of a Kind

Programmer's Manual

September 2021



Table of Contents

commandhandler.c	2
serial.c	6
polling_helper.c	6
Pcb_internal.c	7
Pcb_commands.c	9

commandhandler.c

Syntax: `void commandhandler()`

Description: Interprets commands entered by the user and calls the corresponding function

Parameters: none

Syntax: `void help()`

Description: Displays the list of available commands and what they do

Parameters: none

Syntax: `void shutdown()`

Description: Sends shutdown signal to the machine

Parameters: none

Syntax: `void version()`

Description: Displays the current version and last updated date

Parameters: none

Syntax: `void error()`

Description: Prints error message when invalid command is entered

Parameters: none

Syntax: `int getDate()`

Description: Retrieves the current date of the operating system

Parameters: none

Syntax: `void SetDate(int year, int month, int day)`

Description: Sets the date of the operating system

Parameters: **Year** - the year entered by the user

Month - the month entered by the user

Day - the day entered by the user

Syntax: `void setYear(int year)`

Description: Sets the current year of the operating system

Parameters: **Year** - the year entered by the user

Syntax: `int getYear()`

Description: Gets the current year of the operating system

Parameters: none

Syntax: `void setMonth(int month)`

Description: Sets the current month of the operating system

Parameters: **Month** - the month entered by the user

Syntax: `int getMonth()`

Description: Gets the current month of the operating system

Parameters: none

Syntax: `void setDay(int day)`

Description: Sets the current day of the operating system

Parameters: **Day** - the day entered by the user

Syntax: `int getDay()`

Description: Gets the current day of the operating system

Parameters: none

Syntax: `void setTime(int hours, int minutes, int seconds)`

Description: Sets the time of the operating system

Parameters: **Hours** - the hour entered by the user

Minutes - the minutes entered by the user

Seconds - the seconds entered by the user

Syntax: `int getTime()`

Description: Retrieves the current time of the operating system

Parameters: none

Syntax: `int getHours()`

Description: Retrieves the current hour of the operating system

Parameters: none

Syntax: `void setHours(int hours)`

Description: Sets the current hour of the operating system

Parameters: **Hours** - the hour entered by the user

Syntax: `int getMins()`

Description: Gets the current minute of the operating system

Parameters: none

Syntax: `void setMin(int min)`

Description: Sets the current minute of the operating system

Parameters: **Min** - the minute entered by the user

Syntax: `int getSeconds()`

Description: Gets the current second of the operating system

Parameters: none

Syntax: `void setSec(int seconds)`

Description: sets the current seconds of the operating system

Parameters: `Seconds` - the seconds entered by the user

Syntax: `char *itoa(int num, char buffer[])`

Description: Binary coded digit converter. Converts the time to the BCD format

Parameters: `Num` - the integer that will be converted to char

`Buffer` - the char array that will hold the converted character

Syntax: `void reverse(char buffer[])`

Description: Reverses a character array

Parameters: `Buffer` - the character array that will be reversed

Syntax: `void clear()`

Description: Clears the terminal screen

Parameters: none

Syntax: `void menu()`

Description: Prompts the user with a menu of actions they can perform

Parameters: none

Syntax: `int PCB_exit()`

Description: Asks and allows the user to exit from creating PCB commands

Parameters: none

Syntax: `void PCB_menu()`

Description: Prompts the user with a menu of actions they can perform

Parameters: none

serial.c

Syntax: `int *polling(char *buffer, int *count)`

Description: Calls on the helper function when a letter is found in the register

Parameters: `*Buffer` - the current user input

`*Count` - keeps track of where the cursor is

polling_helper.c

Syntax: `int special_keys(char *buffer, int *count, char letter, int* sizePtr, int *cursorPtr)`

Description: Deals with special keys entered, like arrow keys

Parameters: `*Buffer` - user input from terminal

`*Count` - how full the buffer is

`Letter` - the letter entered in the terminal

`*SizePtr` - pointer to the size of the buffer

`*CursorPtr` - where the cursor is in the buffer

Syntax: `void backspace(char *buffer, int *count, int* sizePtr, int *cursorPtr)`

Description: Enables user to delete in the terminal

Parameters: `*Buffer` - user input from terminal

`*Count` - how full the buffer is

`*SizePtr` - pointer to the size of the buffer

*CursorPtr - where the cursor is in the buffer

pcb_internal.c

Syntax: int freePCB(pcb *toBeFreed)

Description: Deals with freeing up space for the process

Parameters: *toBeFreed - which pcb to free

Syntax: pcb* findPCB (char *name)

Description: Finds a certain pcb with the name entered

Parameters: *name - user input from terminal on which pcb it is

Syntax: pcb* allocatePCB()

Description: Allocates memory for this function

Parameters: None

Syntax: setupPCB(char *name, int class, int priority)

Description: Enables user to setup and initialize a pcb

Parameters: *name - user input from terminal

int class - class number

Int priority- the priority of the process

Syntax: void insertPCB(pcb *process)

Description: takes in a process and inserts it into a queue based on priority

Parameters: *process - user input

Syntax: int removePCB (pcb *process)

Description: Removes the pcb with the name entered

Parameters: *name - user input from terminal on which pcb it is

pcb_commands.c

Syntax: `int createPCB(char *name, int class, int priority)`

Description: Deals with creating a new PCB

Parameters: `*name` - user input from terminal

`int class` - class number

`int priority` - the priority of the process

Syntax: `int deletePCB (char *name)`

Description: Finds a certain pcb with the name entered and deletes it

Parameters: `*name` - has to be a valid name of an already created pcb

Syntax `int blockPCB (char *name)`

Description: Takes the pcb and puts it in the blocked queue

Parameters: `*name` - has to be a valid name of an already created pcb

Syntax `int unblockPCB (char *name)`

Description: Takes the pcb and puts it in the ready queue

Parameters: `*name` - has to be a valid name of an already created pcb

Syntax:**Syntax:** `void insertPCB(pcb *process)`

Description: takes in a process and inserts it into a queue based on priority

Parameters: `*process` - user input

Syntax: `int suspendPCB (pcb *process)`

Description: Puts the pcb with the name entered in the suspended state

Parameters: `*name` - user input from terminal on which pcb it is

Syntax: `int resumePCB (pcb *process)`

Description: Takes the pcb with the name entered out of the suspended state

Parameters: *name - user input from terminal on which pcb it is

Syntax: int suspendPCB (pcb *process)

Description: Puts the pcb with the name entered in the suspended state

Parameters: *name - user input from terminal on which pcb it is

Syntax: int setPriority(char *name, int priority)

Description: Deals with setting a priority for a PCB

Parameters: *name - user input from terminal

Int priority- the priority of the process

Syntax: int showPCB(char* name)

Description: shows the information and details of the specified pcb

Parameters: *name - user input from terminal on which pcb it is

Syntax: void showReady ()

Description: shows all the PCBs in the ready state

Parameters: None

Syntax: void showBlocked ()

Description: shows all the PCBs in the blocked state

Parameters: None

Syntax: void showSuspendedReady ()

Description: shows all the PCBs in the Suspended ready state

Parameters: None

Syntax: void showSuspendedBlocked ()

Description: shows all the PCBs in the Suspended blocked state

Parameters: None

Syntax: `void showAll()`

Description: shows all the PCBs created in every state

Parameters: None

Syntax: `int error_name_check(char* name)`

Description: checks to see if the name entered is valid

Parameters: `*name` - user input from terminal on which pcb it is

Syntax: `void printPCB (pcb *process)`

Description: Will print all the information of the pcb

Parameters: `*name` - user input from terminal on which pcb it is