WestVirginiaUniversity
BENJAMIN M. STATLER COLLEGE OF ENGINEERING AND MINERAL RESOURCES

Marauders OS Programmer’s Manual

Authors: *Nathan Mullins, Riley Stauffer*

Table of Contents

1. Serial.h

1.1 int serial_poll(device dev, char *buffer, size_t len)
Author:
Nathan Mullins
Description
polls a device dev for data and will store it in an array one character at a time until a new line (enter key) is encountered or the buffer is full. Returns the number of bytes read from dev, or a negative number if an error has occurred.

Parameters
device dev - a communication port for serial communication limited to predefined devices COM1, COM2, COM3, COM4, which represent addresses for serial communication ports.
char* buffer - user provided buffer that stores chars being polled from dev. Size is predefined.
size_t len - length of buffer, allowable amount of characters to enter buffer without overflow.

1.2 clear_input_buffer(device dev)
Author:
Nathan Mullins
Description
Clears the specified device of characters to flush unwanted/extra output characters, ideally before writing then reading output to the device. Does not return anything.

Parameters
device dev - a communication port for serial communication limited to predefined devices COM1, CMO2, COM3, COM4, which represent addresses for serial communication ports.

2. itoa.h

2.1 char* itoa(int value, char* destination, int base)

Author:
Nathan Mullins
Description
Converts an integer to a string value with a specified base (ie. binary, hex, decimal, etc.)

Function takes in an integer of base 10, an array to store the converted integer, and a numerical base used to represent the value as a string. Returns the converted string into the destination array.

Parameters

int value - integer to convert to string. Integers must be non-negative and not exceed ((2^31)-1), since an int is 4 bytes, or 32 bits.

char* destination - array to store string. Size of destination must not exceed 50 bytes of storage.

int base - numerical base used to represent the value as a string. Base must be between and including 2-36 (after 36, the number of unique symbols required to represent the digits exceeds the number of standard alphanumeric characters (0-9, A-Z) available)

3. doubly_linked_list.h

3.1 void insertFront(struct Node** head, char* data)

Author:

Nathan Mullins

Description

Inserts a new node at the front/head of a doubly linked list. If the list is not empty, moves nodes over and places the new node at the beginning. Returns nothing.

Parameters

struct Node** head - points to first/front node within a doubly linked list. head must be a pointer to a pointer

char* data - array of data to be placed in the front node. data must be of type char* and data length must be less than the length of the buffer.

3.2 void insertAfter(struct Node* prev_node, char* data)

Author:

Nathan Mullins

Description

Inserts a new node after the specified node prev_node passed into the function into the doubly linked list. If the previous node is null/does not exist, it returns immediately from function. Otherwise, returns nothing.

Parameters

struct Node* prev_node - points to the node that the new node will be placed after. prev_node must be a pointer of type struct.

char* data - array of data to be placed in a new node. data must be of type char* and data length must be less than the length of the buffer.

3.3 void insertEnd(struct Node** head, char* data)
Author:
Nathan Mullins
Description
Inserts a new node at the end of the doubly linked list. If the list is empty, places the new node at the front of the list. Otherwise, traverses through the list until the next node is null, and places the new node in.

Parameters
struct Node** head - points to the node that will be placed at the end of the list. must be a pointer of type struct.

char* data - array of data to be placed in a new node. data must be of type char* and data length must be less than the length of the buffer.

3.4 deleteNode(struct Node** head, struct Node* del_node)
Author:
Nathan Mullins
Description
Deletes the specified node del_node from the doubly linked list. If the list is empty or the del_node is empty, immediately returns because deletion is not possible. Otherwise, removes del_node and corrects pointers of previous and next nodes in the list.

Parameters
struct Node** head - points to first/front node within a doubly linked list. head must be a pointer to a pointer

struct Node* del_node - points to the node that will be deleted. must be a pointer of type struct.

3.5 struct Node
Author:
Nathan Mullins
Description

Structure/layout of each node in a doubly linked list. Each node will contain a pointer to the next node, a pointer to the previous node, and data associated with the node.

Parameters

char* data - array of data associated with a node. data must be of type char* and data length must be less than the length of the buffer.

struct Node* next - points to the next node in a list. must be a pointer.

struct Node* prev - points to the previous node in a list. must be a pointer.

4. comhand.h

4.1 void comhand(void)

Author:

Ian Jackson

Description

Responsible for handling user input from the terminal and executing various commands based on that input. It begins by displaying a welcome message, listens for user input indefinitely, parses the input to identify specific commands, and then executes the corresponding command or displays an error message if the command is not recognized. The supported commands include "version," "shutdown," "help," "getdate," "setdate," "gettime," "settime," and "clear," each with its own specific functionality.

Parameters

N/A

5. help.h

5.1 void help(const char *cmd)

Author:

Ian Jackson

Description

Provides usage instructions/ description for all available commands given to the user. Does not return anything.

Parameters

const char* cmd - provides specific instruction/description about the parameter if it is a command that exists. Parameter cmd must be of length smaller than the defined buffer size.

6. time.h

6.1 void get_time(void)
Author:
Riley Stauffer
Description:
Displays the current time in the form HH:MM:SS. Note, the time is displayed in GMT by default
Parameters:
N/A
6.2 void get_date(void)
Author:
Grant Stumpf
Description:
Displays the current date in the form MM-DD-YY.
Parameters:
N/A

6.3 void set_time(const char *command)

Author:
Riley Stauffer
Description
Changes the internal time of the system. Parses input into hours, minutes, and seconds and converts those into integers. Converts the integers from decimal to BCD. Disables interrupts and writes values directly to the time registers of the system. Re-enables interrupts.
Parameters
const char *command - user input time. If the user writes "setdate HH:MM:SS" the time will be isolated and passed to the function.

6.4 int isValidTimeFormat(const char* input)

Author:
Riley Stauffer
Description
Verifies user input of the set_date(const char *command) function. This function passes the truncated user command from comhand.c. The input will be verified as a valid input, HH:MM:SS if the length is 8, there are ":" in the second and fifth positions, HH<24, MM<60, and SS<60.
Parameters
const char* input - user input time, automatically truncated if the time is preceded by "settime."

6.5 void set_date(uint8_t day, uint8_t month, uint8_t year)
Author:
Grant Stumpf
Description
Changes the internal date of the system by disabling and enabling interrupts and writing directly to the system's time registers. Converts inputs from decimal to BCD before writing to the registers.
Parameters
uint8_t day - day (DD) parsed from user input, uint8_t month - month (MM) parsed from user input, uint8_t year - year (YY) parsed from user input.

6.6 int hexToDec(uint8_t hex)
Author:
Grant Stumpf
Description
Converts an integer in binary coded decimal (BCD) to decimal. Returns decimal value integer.
Parameters
uint8_t hex - input of BCD value before conversion

6.7 uint8_t decToHex(int decimal)
Author:
Riley Stauffer
Description
Converts an integer in decimal to BCD. Returns uint_8t of input in BCD.
Parameters
int decimal - input of decimal value before conversion

7. version.h

7.1 void version(void)
Author:
Grant Stumpf
Description
prints the most updated/current version of the MarauderOS along with the last compilation date. Function is called by the user in the terminal by typing "version" when running MarauderOS.

Parameters

version() takes no parameters and does not return anything.