==========

PROGRAMMER'S MANUAL

==========

This is the programmers manual page for the MPX OS Version R1. It contains information necessary to manipulate the code and therefore the system.

This manual will be organized by file. The file name will be the section, and the contents will be the function header, description, parameters, and any returned values.

==========

SERIAL.C

==========

FUNCTION CALL:

serial_poll();

DESCRIPTION:

Continuously polls (checks) the LSR (Line Status Register) for data and stores it one bit at a time in to a user-instantiated buffer, until either the buffer is full or a null terminator is encountered

PARAMETERS

device dev - device to read from
char *buffer - the user-provided buffer to write into
size_t len - size of the buffer

RETURN

size_t bytes_read - the amount of bytes read

COMMHAND.C

==========

FUNCTION CALL:

commhand();

DESCRIPTION:

Checks buffer from polling function to see if user input matches desired function input. If it does, proceed with command. If not, exit.

PARAMETERS
void

RETURN
void

ERRORS
"Invalid input, please try again."
User's input does not match any of the commands used to trigger the functions. Immediately allows the user to attempt to put the input command indefinitely.
"Buffer Overflow."
User's input exceeded the amount of space allocated in the buffer. Immediately allow the user to attempt to input the command indefinitely.

FUNCTION CALL:
help();

DESCRIPTION:
Provides instructions to the user upon request on how to operate the terminal.

PARAMETERS
void

RETURN
void

ERRORS
none

FUNCTION CALL:
shutdown

DESCRIPTION:
Exits the loop initiated in the commhand() function upon confirmation from the user. Returns execution to kmain().

PARAMETERS
	void

RETURN
	void

ERRORS
	none

FUNCTION CALL:
	get_time();

DESCRIPTION:
	Reads the binary coded decimal stored in the Real Time Clock Register 0x71.

PARAMETERS
	void

RETURN
	void

ERRORS
	none

FUNCTION CALL:
	set_time();

DESCRIPTION:
	Disables interrupts, writes a new value to the RTC register 0x71 using outb(), then re-enables interrupts

PARAMETERS
	void

RETURN

void

ERRORS

“Input buffer overflow.”

User entered a time that caused overflow in the buffer. User has to re-enter the command and the time correctly. System immediately allows the user to try again indefinitely.

“Invalid input format.”

User entered a time that was not HH:MM formatting, the hours or minutes were not within the range of 00-24 and 00-59 respectively.

FUNCTION CALL:

get_date();

DESCRIPTION:

Reads date value stored in system register

PARAMETERS

void

RETURN

void

ERRORS

none

FUNCTION CALL:

set_date();

DESCRIPTION:

Writes new value to date value stored in register

PARAMETERS

void

RETURN

void

ERRORS

"Input buffer overflow."

User entered input that exceeded the amount of space in the buffer. System forces the user to re-enter the command and the time correctly, the cycle happens indefinitely.

"Invalid date format. Please use DD/MM/YY."

User entered a date where the day, month and year were not within the set bounds of 01-31, 01-12 or 00-99. System assumes the year starts at 2000.

FUNCTION CALL:

create_pcb(const char *name, int class, int priority);

DESCRIPTION:

Creates a new PCB queue node and places it in the appropriate location in the queue.

PARAMETERS

const char *name
int class
int priority

RETURN

void

ERRORS

"Name for PCB already exists."

User attempted to create a PCB with the name of a process already in queue.

"Invalid PCB class."

User attempted to create a PCB with a class that is invalid.

"Invalid PCB priority"

User attempted to create a PCB with a priority that was invalid.

FUNCTION CALL:

delete_pcb(const char *name)

DESCRIPTION:

Finds the input process name and removes it from the queue using pcb_remove().

PARAMETERS

const char *name

RETURN

void

ERRORS

"Invalid PCB name."

User attempted to delete a PCB with a name that did not fit into the requirements given for PCB names.

"PCB not found."

User attempted to delete a PCB with a name that fit the requirements, but did not match the name of a PCB in the queue.

"System processes cannot be deleted."

System cannot delete the process, immediately will return to the menu for the user to re-enter their desired commands indefinitely.

"Failed to free PCB memory."

System cannot free the memory, immediately will return to the menu for the user to re-enter their desired commands indefinitely.

FUNCTION CALL:

block_pcb(const char *name);

DESCRIPTION:

Puts the input process into a blocked state, and then moves it to the blocked queue.

PARAMETERS

const char *name

RETURN

void

ERRORS

"Invalid process name."

User attempted to block a PCB with a name that did not fit into the requirements given for PCB names.

"Process not found."

User attempted to block a PCB with a name that fit the requirements, but did not match the name of a PCB in the queue.

FUNCTION CALL:

unblock_pcb(const char *name);

DESCRIPTION:

Takes input process, finds it in the blocked queue and moves it into the ready queue.

PARAMETERS

const char *name

RETURN

void

ERRORS

“Invalid process name.”

User attempted to unblock a PCB with a name that did not fit into the requirements given for PCB names.

“Process not found.”

User attempted to unblock a PCB with a name that fit the requirements, but did not match the name of a PCB in the queue.

FUNCTION CALL:

suspend_pcb(const char* name);

DESCRIPTION:

Suspends process by putting it in suspended queue and updating state number to match suspended state.

PARAMETERS

const char* name

RETURN

void

ERRORS

“Invalid process name.”

User attempted to suspend a PCB with a name that did not fit into the requirements given for PCB names.

“Process not found.”

User attempted to suspend a PCB with a name that fit the requirements, but did not match the name of a PCB in the queue.

FUNCTION CALL:

resume_pcb(const char* name);

DESCRIPTION:

Places the requested process into the ready state from being in the suspended queue.

PARAMETERS

const char* name

RETURN

void

ERRORS

"Invalid process name."

User attempted to resume a PCB with a name that did not fit into the requirements given for PCB names.

"Process not found."

User attempted to resume a PCB with a name that fit the requirements, but did not match the name of a PCB in the queue.

FUNCTION CALL:

set_pcb_priority(const char* name);

DESCRIPTION:

Allows the user to set the PCB priority level between 0 and 9, and places the process appropriately in the queue.

PARAMETERS

const char* name

int new_priority

RETURN

void

ERRORS

"Invalid PCB name."

User attempted to resume a PCB with a name that did not fit into the requirements given for PCB names.

"Invalid priority, please assign a priority in range 0-9."

User attempted to assign a priority that was outside the range.

FUNCTION CALL:

show_pcb(const char* name);

DESCRIPTION:

Displays the requested process's name, class, state, suspended status and priority.

PARAMETERS

const char* name

RETURN

void

ERRORS

"Invalid PCB name."

User attempted to resume a PCB with a name that did not fit into the requirements given for PCB names.

"Invalid priority, please assign a priority in range 0-9."

User attempted to assign a priority that was outside the range.

FUNCTION CALL:

show_ready(const char* name);

DESCRIPTION:

Displays the following information of all the processes in the ready state: name, class, state, suspended status and priority.

PARAMETERS

none

RETURN

void

ERRORS

none

FUNCTION CALL:

show_blocked(const char* name);

DESCRIPTION:

Displays the following information of all the processes in the blocked state: name, class, state, suspended status and priority.

PARAMETERS

none

RETURN

void

ERRORS

none

FUNCTION CALL:

show_all(const char* name);

DESCRIPTION:

Displays the following information about all of the created processes, no matter their state: name, class, state, suspended status and priority.

PARAMETERS

none

RETURN

void

ERRORS

none
