
Programmer's Manual

This is the programmer manual for the MPX OS, documentation for various versions are supplemented below

Author's Notes

The manual proceeds on the premise that the reader has a general understanding of technology

-If not an experienced user, please refer to the user's manual

Command_Handler.C

FUNCTION CALL:

command_handler(void)

DESCRIPTION

Gives the user a menu and 7 options to choose from to perform various Functions such as setting the current date and time, managing processes and yielding cpu usage

PARAMETERS

VOID

RETURN

VOID

FUNCTION CALL:

help _command

DESCRIPTION

Provides the user with another menu asking which commands the user needs help with. Displays useful information about each command

PARAMETERS

VOID

RETURN

VOID

FUNCTION CALL:

Sht_dwn

DESCRIPTION

A function in the code that changes the shutdown value from the preset 1 value to 0, breaking the while loop

PARAMETERS

VOID

RETURN

RETURNS INTEGER TO CHANGE PRESET STATE OF SHUT DOWN FROM 1 TO 0 AND CLOSE COMMHAND

FUNCTION CALL:

help _command

DESCRIPTION

Provides the user with another menu asking which commands the user needs help with. Displays useful information about each command

PARAMETERS

VOID

RETURN

VOID

FUNCTION CALL:

get_version

DESCRIPTION

Gets the user the current version number and compilation date

PARAMETERS

VOID

RETURN

VOID

USER_COMMANDS.C

FUNCTION CALL:

upper_to_lower

DESCRIPTION

Converts capital letters to lowercase letters

PARAMETERS

A POINTER TO THE STRING THAT IS INTENDED TO BE LOWERCASED

RETURN

A CHARACTER POINTER TO THE LOWERCASE STRING

FUNCTION CALL:

`get_time`

DESCRIPTION

Returns the user the time that is currently set on the clock by accessing index registries and assign seconds, minutes and hours to index pulls

PARAMETERS

VOID

RETURN

VOID

FUNCTION CALL:

set_time

DESCRIPTION

Lets the user set the time that the system is using by writing to corresponding time registers

PARAMETERS

VOID

RETURN

VOID

FUNCTION CALL:

get_date

DESCRIPTION

Lets the user get the date that the system uses by shifting bits

PARAMETERS

VOID

RETURN

VOID

FUNCTION CALL:

`set_date`
DESCRIPTION

Lets the user set the date that the system uses by converting to BCD and writing to the day, month year registry

PARAMETERS

VOID

RETURN

VOID

FUNCTION CALL:

`itoa`

DESCRIPTION

Converts an integer to a null terminated string

PARAMETERS

- `n` number that it takes
- `str[]` string that the number gets converted to
- `base` base that is being converted from

RETURN

Returns the newly converted string.

FUNCTION CALL:

`delete_pcb`

DESCRIPTION:

Deletes a non-system, user given, process control block

PARAMETERS:

- `pcb_name`: Name of the process

RETURN

VOID

FUNCTION CALL

`block_pcb`

DESCRIPTION

Moves a user specified process to the blocked state

PARAMETERS

- `pcb_name`: Name of the process

RETURN

VOID

FUNCTION CALL

`unblock_pcb`

DESCRIPTION

Moves a user specified blocked process to the ready state

PARAMETERS

- `pcb_name`: Name of the process

RETURN

VOID

FUNCTION CALL:

suspend_pcb

DESCRIPTION

Puts a user defined process into a suspended state

PARAMETERS

- `pcb_name`: Name of the process

RETURN

VOID

FUNCTION CALL:

resume_pcb

DESCRIPTION:

Takes a user defined process in a suspended state, and unsuspends

it PARAMETERS:

- `pcb_name`: Name of the process

RETURN

VOID

FUNCTION CALL:

set_pcb_prio

DESCRIPTION:

Changes a user defined process' priority to a new priority defined by the user

PARAMETERS

- `pcb_name`: Name of the process
- `priority`: New priority for the process

RETURN

VOID

FUNCTION CALL:

show_pcb

DESCRIPTION:

Prints off a user defined process' name, class, state, status, and priority

PARAMETERS:

- pcb_name: Name of the process

RETURN

VOID

FUNCTION CALL:

show_ready

DESCRIPTION:

Prints out all processes in the ready queue along with characteristics for pcb

PARAMETERS:

None

RETURN:

VOID

FUNCTION CALL:

show_blocked

DESCRIPTION

Prints out all processes in the blocked queue

PARAMETERS:

NONE

RETURN

VOID

FUNCTION CALL:

show_all

DESCRIPTION

Prints out all current processes in all queues

PARAMETERS

NONE

RETURN

VOID

FUNCTION CALL:

loadR3PCB

DESCRIPTION

Each process is loaded and is queued in a non-suspended ready state,

With a name and prio of user's choosing

PARAMETERS

(char* name = desired process name to be loaded , int function = the function to be loaded in)

RETURN

VOID

FUNCTION CALL:

loadR3

DESCRIPTION

Searches for each R3 test process and if none are loaded into a queue then the

program iterates through and adds them

PARAMETERS

VOID

RETURN

VOID

FUNCTION CALL:

createProc

DESCRIPTION

Creates a process with a given name, address location, class and priority

PARAMETERS

(char *name = desired name for process, int function = location, User or system class with given priority)

RETURN

VOID

FUNCTION CALL:

createAlarm

DESCRIPTION

Creates an alarm and prompts user for a message, hours minutes and seconds to raise the interrupt at a given time

PARAMETERS

VOID

RETURN

VOID

FUNCTION CALL:

initAlarm

DESCRIPTION

Initializes the alarm

PARAMETERS

(char *message=message to be sent, char time[9] = desired time saved in an array)

RETURN

VOID

SERIAL.C

FUNCTION CALL:

serial_poll()

DESCRIPTION

Data waits and “polls” the buffer receives data from the device and performs actions on what is the data received

PARAMETERS

- `dev` is the device to read from
- `buffer` is the user-provided buffer
- `len` is the size of the user-provided buffer
- Returns the number of bytes read from the device, or a negative number on

RETURN

Returns the number of bytes read from the device, or a negative number on error -----

PCB.C

FUNCTION CALL

pcb_allocate

DESCRIPTION

allocates memory to a new process control block structure

PARAMETERS:

NONE

RETURNS

- `pcbInit`: memory allocated process

FUNCTION CALL:

pcb_free

DESCRIPTION:

Frees all memory from a given process

PARAMETERS:

- `*pcb`: process control block struct to free memory from

RETURN

- `i`: Returns a 0 on success, or returns any other number and an error message if there is an error

FUNCTION CALL

pcb_setup

DESCRIPTION:

Sets up a new PCB using the name, class, and queue priority

PARAMETERS

- `name`: Name of the process
- `class`: Class of the PCB, 0 for System Process, 1 for User Process
- `priority`: priority to of the PCB

RETURN

- `newpcb`: Returns new process control block structure

FUNCTION CALL

`find_pcb`

DESCRIPTION:

Finds a process that matches the user defined name

PARAMETERS:

- `desiredName`: Name of the process to find

RETURN:

- `newPtr`: pointer to the found process structure
- `NULL`: returns NULL on error or if the process is not found

FUNCTION CALL

`pcb_remove`

DESCRIPTION

Removes a given process structure from its respective queue

PARAMETERS:

- `p`: Process structure to remove

RETURN

- `0`: Returned on success
- `1`: Returned on error

HEAP_MANAGER.C

FUNCTION CALL:

`initialize_heap`

DESCRIPTION:

Initializes the memory heap, done on startup. Creates a new memory control block from a specific number of bytes to allocate. Automatically makes sure all the memory is free.

PARAMETERS:

`size_t size`: amount of memory to initialize, given in BYTES

RETURN:

VOID

FUNCTION CALL:

allocate_memory

DESCRIPTION:

Allocates memory of a given size in bytes. If memory is not available, then an error is returned.

PARAMETERS:

size_t size: amount of memory to allocate, given in BYTES

RETURN:

void* address: void pointer to the starting memory address where its allocated

FUNCTION CALL:

free_memory

DESCRIPTION:

Frees the memory of a given starting address. If memory is not allocated, then an error is returned. When the memory freed is less than the allocated, it will split or merge free blocks of memory together.

PARAMETERS:

void* block: address of the allocated memory to free

RETURN:

int 0 on success
int 1 on failure

FUNCTION CALL:

split

DESCRIPTION:

"Splits" an MCB so there is still an amount of free/allocated memory left after allocate_memory and free_memory are run respectively.

PARAMETERS:

mcb* block: The mcb to split
int size: the size of the block

RETURNS:

NULL: Error or Block was not split
MCB: MCB is split and returned

FUNCTION CALL:

Merge

DESCRIPTION:

Merges two blocks of free memory so there isn't fragmentation or leakage.

PARAMETERS:

mcb* block: The mcb that will merge

RETURNS

int 0: On Success

int 1: On Failure

FUNCTION CALL:

mcb_remove

DESCRIPTION:

Removes a memory block from the allocated or free lists

PARAMETERS:

struct mcb m: The entire structure that will be removed

RETURN:

int 0: On Success

int 1: On Failure

MEMORY_COMMANDS.C

FUNCTION CALL:

memory_alloc

DESCRIPTION:

User command that will allocate a defined size of memory.

PARAMETERS:

int size: User input size to allocate

RETURN:

void* address: starting address of the allocated memory

NULL: On error

FUNCTION CALL:

memory_free

DESCRIPTION:

User command that will free the memory at a given address.

PARAMETERS:

void* address: user entered address(in hexadecimal)

RETURN:

VOID

FUNCTION CALL:

show_mcb

DESCRIPTION:

Helper command that will print off the size and address of a mcb.

PARAMETERS:

mcb* mcb: MCB that will be shown

RETURN:

VOID

FUNCTION CALL:

show_alloc_mem

DESCRIPTION:

User command that will show the amount of memory allocated, and it's starting address.

PARAMMETERS:

NONE

RETURN:

VOID

FUNCTION CALL:

show_free_mem

DESCRIPTION:

User command that will show the amount of free memory

PARAMETERS:

NONE

RETURN:

VOID
