Programmer's Manual

Group 2: Jacob Dunning, Dana Foual, Arpan Kumar, Kyle Kuminkoski

Table of Contents

R1

comhand - int comhand()
Processes user input and calls functions to complete the desired command. Menu driven.

Parameters:
N/A

Returns:
N/A

polling - int *polling(char *buffer, int *count)
Collects user input from keyboard. Data is collected on a character-by-character basis and stored in the COM1 BASE register (0x3F8).

Parameters:
Buffer - Character buffer
Count - Character count tracker

Returns:
Count

help - void help()
Provides usage instructions for each command.

Parameters:
N/A

Returns:
N/A

version - void version()
Displays current version of the MPX and the completion date.

Parameters:
N/A

Returns:
N/A

shutdown - int shutdown()

Exits command handler loop. Asks the user for confirmation before shutting down the system.

Parameters:
N/A

Returns:
N/A

setDate - int setDate()
Takes user input and changes the date on the system to the user inputted date.

Parameters:
N/A

Returns:
N/A

getDate - int getDate()
Displays the date stored in the system

Parameters:
N/A

Returns:
Date

setTime - int setTime()
Takes user input and changes the time on the system to the user inputted time.

Parameters:
N/A

Returns:
N/A

getTime - int getTime()
Displays the time stored in the system.

Parameters:

N/A

Returns:
Time in HH:mm:ss format

abs - int abs(int num)
Computes the absolute value.

Parameters:
Num- integer to perform absolute value

Returns:
Absolute value of the number

swap - void swap(char *x, char *y)
Swaps the position of two characters.

Parameters:
x - character 1
y - character 2

Returns:
N/A

reverse - char* reverse(char str[], int i, int j);
Reverses a character array.

Parameters:
str[] - character array
i - first element in array
j - last element in array

Returns:
Reversed char array

itoa - char* itoa(int value, char* buffer, int base)
Converts an integer to base 2, 8, 10, or 16. Converts to char array then adds to buffer.

Parameters:
value - integer

buffer -
base - base user want to convert to

Returns:
Char array

clear - int clear()
Clears contents of the entire page and leaves a new line to continue with next command

Parameters:
N/A

Returns
Integer value

R2

AllocatePCB - pcb* allocatePCB(void)
Allocates memory for a new PCB using sys_alloc_mem(). Possibly including the task to perform any reasonable initialization.

Parameters:
N/A

Returns:
PCB pointer - success
NULL - error during allocation

FreePCB - void freePCB(pcb* pcb)
Frees all memory associated with a given PCB

Parameters:
PCB pointer to the pcb to be freed

Returns:
Success or error code

SetupPCB - pcb* setupPCB(char *name, int priority, processClass c)
Create an empty PCB, initializes the information and sets the PCB state to ready

Parameters:
Process name
Process Class
Process priority

Returns:
PCB pointer - success
NULL - error or invalid parameters

FindPCB - pcb* findPCB(char *name)
Searches all queues for a process with a given name

Parameters:
Process name

Returns:

PCB pointer - success
NULL - pcb not found

Find - pcb* find(char *name, pcb_queue *queue)
Helper function for findPCB, finds pcb in the queue

Parameters:
Process name
Process queue

Returns:
PCB

InsertPCB - int insertPCB(pcb* pcb)
Inserts a PCB into the appropriate queue

Parameters:
PCB pointer

Returns:
Success or error message

RemovePCB - int removePCB(pcb* pcb)
Removes a PCB from the queue in which it is currently stored

Parameters:
Pointer to a PCB

Returns:
Success or error message

CreatePCB - int createPCB(char* name,)
Calls SetupPCB and insert the PCB in the appropriate queue

Parameters:
Process name
Process class
Process priority

Returns:

Success or error message

DeletePCB - int deletePCB(char* name)
Removes a PCB from the appropriate queue and then frees all associated memory. This method finds the pcb, unlinks it from the appropriate queue, and then frees it.

Parameters:
Process name

Returns:
Success or error message

BlockPCB - int blockPCB(char* name)
Finds a PCB and sets its state to blocked, then reinserts it into the appropriate queue.

Parameters:
Process name

Returns:
Success or error message

UnblockPCB - int unblockPCB(char* name)
Places a PCB in the unblocked state and reinserts it into the appropriate queue

Parameters:
Process name

Returns:
Success or error message

SuspendedPCB - int suspendPCB(char* name)
Places a PCB in the suspended state and reinserts it into the appropriate queue

Parameters:
Process name

Returns:
Success or error message

ResumePCB - int resumePCB(char* name)
Places a PCB in the not suspended state and reinserts it into the appropriate queue

Parameters:
Process name

Returns:
Success or error message

SetPCBPriority - int setPCBPriority(char* name, int newPriority)
Sets a PCB's priority and reinserts the process into the correct place in the correct queue

Parameters:
Process name
New priority

Returns:
Success or error message

ShowPCB - void showPCB(char* name)
Displays the following information for a PCB: Process Name, Class, State, Suspended Status, Priority

Parameters:
Process name

Returns:
PCB information

ShowReady - void showReady()
Displays the following information for each PCB in the ready queue: Process Name, Class, State, Suspended Status, Priority

Parameters:
N/A
Returns:
PCB information in the ready queue

ShowBlocked - void showBlocked()

Displays the following information for each PCB in the blocked queue: Process Name, Class, State, Suspended Status, Priority

Parameters:
N/A

Returns:
PCB information in the blocked queue

ShowAll - void showAll()
Displays the following information for each PCB in the blocked and ready queue: Process Name, Class, State, Suspended Status, Priority

Parameters:
N/A

Returns:
PCB information in the blocked and ready queue

Sprintf - void sprintf(char* input)
Send formatted output to the string, stores the output on char buffer.

Parameters:
Character input

Returns:
N/A

ProcessType - int processType (char* c)
Takes user input and processes it, helper method for createPCB that processes class type

Parameters:
Character input, class type

Returns:
Different int depending on input for further processing

struct pcb

char name[30] - name of pcb
classType[20] - pcb type
Int priority - pcb priority value
processState state - state of pcb
Struct pcb *next, *prev - next and previous pointer
Unsigned char stack[STACK_SIZE] - stack size 1024
Unsigned char *top, *base - pointer to top and base of stack

R3

Sys_call - u32int* sys_call(context *registers)
Prepares mpx for next ready process to begin/resume execution

Parameters:
Context registers

Returns:

Yield - void yield()
Command handler yields to other processes, any processes in ready queue is executed

Parameters:
N/A

Returns:
N/A

Loadr3 - void loadr3()
Loads all R3 processes

Parameters:
N/A

Returns:
N/A

makeProc1 - void makeProc1()
Makes R3 processes 1

Parameters:
N/A
Returns:
N/A

makeProc2 - void makeProc2()
Makes R3 processes 2

Parameters:
N/A
Returns:
N/A

makeProc3 - void makeProc3()
Makes R3 processes 3

Parameters:
N/A
Returns:
N/A

makeProc4 - void makeProc4()
Makes R3 processes 4

Parameters:
N/A
Returns:
N/A

makeProc5 - void makeProc5()
Makes R3 processes 5

Parameters:
N/A
Returns:
N/A

nextProc - pcb* nextProc()
Runs next process

Parameters:
N/A

Returns:
N/A

struct context

u32int gs, fs, es, ds - data segment registers
u32int edi, esi, ebp, esp, ebx, ecx, eax - 32-bit general purpose registers
u32int eip, cs, eflags - instruction pointer, code segment, 32 bit flags registers

R4

startComhandProc() - void startComhandProc()
Adds command handler to ready queue

Parameters:
N/A

Returns:
N/A

startsIdleProc() - void startsIdleProc()
Adds idle to ready queue

Parameters:
N/A

Returns:
N/A

Alarm - void alarm()
Set an alarm with a specific message and time that it should be printed

Parameters:
N/A

Returns:
N/A

R5

initializeHeap() - void initializeHeap(u32int size)
Allocates all the memory available for MPX

Parameters:
U32int - size

Returns:
N/A

allocateMemory() - u32int allocate(u32int size)
Allocates memory from the heap

Parameters:
u32int - size

Returns:
int - address of memory block

freeMemory() - int freeMemory(void* bA)
Frees a block of memory that was previously allocated

Parameters:
Void* - beginning address

Returns:
Success or error message

showAllocated() - void showAllocated()
Shows the address and size of the block in the allocated list

Parameters:
N/A

Return:
Memory block information

showFree() - void showFree()
Shows the address and size of the block in the free list

Parameters:
N/A

Return:
Memory block information

isEmpty() - int isEmpty()
Returns true or false based on whether the heap contains only free memory

Parameters:
N/A

Return:
1 - true
0 - false

Insert() - int insertCMCB(cmcb* insert)
Inserts a cmcb to either the allocated or free list

Parameters:
cmcb* - insert

Return:
N/A

Remove() - int removeCMCB(cmcb* remove)
Removes cmcb from either the allocated or free list

Parameters:
cmcb* - cmcb to be removed

Return:
N/A

struct cmcb

Int allocated - allocated or free list
U32int size - size to be allocated
U32int beginA - Beginning address
Cmcb *next, *prev - pointer to next and previous cmcb

struct list

Cmcb *head - pointer to the head of the list

R6

Com_open - int com_open(int *eflag_p, int baud_rate)
Initializes the serial port

Parameters:
Eflag_p - Pointer to an integer event flag
Baud_rate - Int value representing desired baud rate

Returns:
N/A

Com_close - int com_close(void)
Called at the end of the session of serial port use

Parameters:
N/A

Returns:
Success or error codes

Com_read - int com_read(char *buf_p, int *count_p)
Obtains input characters and loads them into the requestor's buffer

Parameters:
Buf_p - far pointer to the starting address of the buffer to receive input characters
Count_p - address of integer count value indicating number of characters to be read

Returns:
Success or error codes

Com_write - int com_write(char *buf_p, int *count_p)
Initiates the transfer of a block of data to the serial port

Parameters:
Buf_p - far pointer to the starting address of the buffer to receive input characters
Count_p - address of integer count value indicating number of characters to be read

Returns:
Success or error codes

Top_handler - void top_handler();
Serial port interrupt handler. Vector transfers initially to the first level handler, which is responsible for determining the exact cause of the interrupt and performing general processing. This in turns selects and calls the specific second-level handler appropriate for the specific interrupt.

Parameters:
N/A

Returns:
N/A

Struct dcb
Int open - flag indicating whether opened or closed
Int events - flag indicating completion of operation
Dcb_status status - indicates what status the dcb is
Unsigned char* in - input buffer, stores info from device requested by the application
Int in_x - current index
Int in_s - size of 'in' buffer
Unsigned char* out - output buffer, stored info for device requested by application
Int out_x - current index
Int out_s - size of 'out' buffer
Unsigned char ring [16] - ring buffer
Int ring_inx - current in index
Int ring_outx - current out index
Int ring_s - size of buffer