Four of a Kind

User Manual

November 2021

User Commands

Version()- to call on this function the user will type in version and the terminal will print the current version of MPX and the completion date. This will be updated for every module completion.

Gettime()- This function gets the current time set in the terminal. A user simply needs to call the function and it will show the hour, minute and second of the current time when called.

Settime(int hrs, int min, int sec)- the user is able to call this function to set the current time in the terminal. The parameters will take in the hour, minute and second that the user inputs and sets that as the current time.

Getdate()- This function gets the current date set in the terminal. A user will call the function by typing getdate() and it will show the date including the month, day and year.

Setdate(int month, int day, int year)- the user is able to call this function to set the current date in the terminal. It will have three parameters to get the information from the user and then set those requests as the current date.

Help() - This function when called will pop up a screen with all the commands a user has access to and what each command can do.

Shutdown()- this function exits the command handler loop and will return to kmain() and the system will halt. However before it does all that it will ask for confirmation from the user asking if they are sure they want to shut down and the user must enter Y for yes or N for no before the function completes.

AllocatePCB()- This function will use sys_alloc_mem() to allocate memory for a new PCB, possibly including the stack, and perform any reasonable initialization. It will return the pcb pointer if successful.

FreePCB()- This function will use sys_free_mem() to free all memory associated with a given PCB (the stack, the PCB itself, etc.).

SetupPCB()- This function will call AllocatePCB() to create an empty PCB, initializes the PCB information, sets the PCB state to ready, not suspended. It will return PCB Pointer when successful

FindPCB()- This function will search all queues for a process with a given name. It will return pcb pointer if successful.

InsertPCB()- This function will insert a PCB into the appropriate queue.

RemovePCB()- This function will remove a PCB from the queue in which it is currently stored.

Suspend()- This function will suspend a pcb and place it in the suspended block state

Resume()- This function will take the pcb from the suspended state and put it back into the ready queue

Set Priority(int prio)- This function will set a priority for the pcb that the user gives.

Show PCB(char *name)- This function will show all the information of the pcb

Show all Processes()- This function will show all the processes no matter what state

Show Ready Processes()- This function will show all the PCBs in the ready state

Show Blocked Processes()- This function will show all the PCBs in the blocked state

Delete PCB()- The Delete PCB command will remove a PCB from the appropriate queue and then free all associated memory. This method will need to find the pcb, unlink it from the appropriate queue, and then free it

sys_call_isr()- This will push all the general purpose register to the stack and return from interrupt

sys_call()- This declares a PCB as a global variable and checks to see if sys call has been called before.◦If sys_call has not been called, save a reference to old (the caller's) context in a global variable. Otherwise, return the context

Yield()- This function is temporary only in R3. It will cause the commhand to yield to other processes

loadr3()- This function will load all R3 processes into memory in a suspended ready

setAlarm()- This function will allow a user to choose to set an alarm with whatever message they want and choose what time for it to go off.

idle()- This function will be used for dispatching if no other processes are available to execute

showFree()- This function will traverse through the list and show the address and size of all the free blocks.

showAllocated()- This function will traverse through the list and show the address and size of all the free blocks.

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

com_write(char *buf_p, int *count_p)-

com_open(int baud_rate)- Initializes the DCB, sets the new interrupt handler address and enables all necessary interrupts

com_close()- Ends the serial port use by clearing the open indicator and disabling the PIC mask register

output_handler()- It will ignore the interrupt if the status is not writing, otherwise it gets the next character from the requestors output buffer and keeps going until all characters have been transferred. Then it resets the status to idle

input_handler()- Reads the character from the input register, if status is not reading then it stores the character into the ring buffer until its full, if status is reading then its stores the character into the requestors input buffer then sets status to idle

set_int(int bit, int on)- Initializes the DCB, sets the new interrupt handler address and enables all necessary interrupts

top_handler()- Will figure out what type of interrupt this is and then call the output or input handler

Error Messages for R6

com_open error codes

INVALID_FLAG_PTR -101

INVALID_BAUD_DIVISOR -102
PORT_ALREADY_OPEN -103

com_close error codes
PORT_NOT_OPEN -201

com_read error codes
READ_PORT_NOT_OPEN -301
INVALID_BUFFER_ADDRESS -302
READ_INVALID_COUNT -303
DEVICE_BUSY -304

com_write error codes
WRITE_PORT_NOT_OPEN -401
WRITE_INVALID_BUFFER_ADDRESS -402
WRITE_INVALID_COUNT -403
WRITE_DEVICE_BUSY -404

Temporary Commands for R2 – will be removed for R3/R4

Create PCB()-The Create PCB command will call SetupPCB() and insert the PCB in the appropriate queue

Block()- This function will find a PCB sets its state to blocked and reinserts it into the appropriate queue

Unblock()- This function will place a PCB in the unblocked state and reinserts it into the appropriate queue

Temporary Commands for R5

init_heap (*u32int size***)**- This function allocates all the memory available for your MPX. It takes in an integer parameter that will be the size of the heap in bytes.

allocateMemory (*u32int size***)** - This function is called to allocate memory from the heap. The integer parameter that it takes on will be the amount of bytes you want to allocate from the heap.

freeMemory(*cmcb* toBFreed***)** – This function will free a particular block of memory that was previously allocated. The parameter is using a pointer to an address in memory.

IsEmpty() - This function will return true or false if the heap is empty or if the heap only has free memory