G5 Operating System

Programmer Manual

Group 5
Jose Vitale
Nate Huff
Shivan Prasad
Stephen Mabry

Functions

comhand()

- Parameters

  No parameters

- Return Values

  Void return type - takes input and produces display messages

- Implementation / Application

  Runs and processes serial polling information to be passed into the buffer. Parses the input to call the appropriate command.

keyfromstring()

-Parameters

Takes in a char* key

-Return Values

Returns a port for t_symstruct

-Implementation / Application

Takes a string input and checks it against the t_symstruct table.

strcat()

-Parameters

Two const char*'s to combine together

-Return Values

A concatenated string from the two input strings

-Implementation / Application

Concatenates two strings together while retaining memory allocation

Strcmp()

-Parameters

Two strings to be compared

-Return Values

An integer based on the success or failure of the comparison

-Implementation / Application

Returns a 0 if the strings are identical, otherwise returns the difference in the indexes of the characters

Strcpy()

-Parameters

A string to be output, as well as a string to be copied

-Return Values

A string that has been copied from an input string

-Implementation / Application

Copies a string that has been input, mostly for comparison and mutation purposes

Strlen()

-Parameters

A string to be measured

-Return Values

A size variable for the inputed string

-Implementation / Application

Used to find the length of a string

Strtok()

-Parameters

Input strings intended to be tokenized and combined

-Return Values

A string comprised of tokenized input from the input strings

-Implementation / Application

Intended to tokenize string inputs

toLowerCase()

-Parameters

A pointer to a string to be converted to all lowercase letters

-Return Values

A copy of the string inputed in all lowercase

-Implementation / Application

Turns a string into all lowercase for case insensitive comparison purposes

createReady()

-Parameters

None

-Return Values

None

-Implementation / Application

Instantiates and makes ready the Ready queue to accept PCBs

createBlocked()

-Parameters

None

-Return Values

None

-Implementation / Application

Instantiates and makes ready the Blocked queue to accept PCBs

pcb_allocate()

-Parameters

None

-Return Values

Returns a PCB that has had appropriately sized memory allocated for it

-Implementation / Application

Allocates the memory necessary for a pcb to be created

pcb_free()

-Parameters

A pointer to the PCB that will be allocated

-Return Values

An integer value that indicates a success or failure

-Implementation / Application

Allocates memory for a new PCB to be created, and initializes the PCB

pcb_setup()

-Parameters

- char* - the name of the process
- Int - the class of the process with 0 being a user application and 1 being a system process, and the priority (0-9) of the process

-Return Values

Returns a pointer to the initialized PCB on a success, NULL on an error in almost any category

-Implementation / Application

Allocates a new PCB and initializes it with the parameters into the Ready state

pcb_find()

-Parameters

The name of the process being looked for

-Return Values

Returns a pointer to the sought PCB if found, NULL otherwise

-Implementation / Application

Searches all queues for a PCB with the passed in name

pcb_insert()

-Parameters

A pointer to the PCB being enqueued

-Return Values

None

-Implementation / Application

Inserts a PCB into the appropriate queue, based on the available fields

push()

-Parameters

- queue* - a pointer to the queue that will be pushed to
- Pcb - a pointer to the PCB that will be added to the passed in queue

-Return Values

None

-Implementation / Application

Pushes a specified PCB to the top of the specified queue

pcb_remove()

-Parameters

A pointer to the PCB to dequeue

-Return Values

An integer that represents either a fail or success state

-Implementation / Application

Removes a PCB from its current queue, but does not free any associated memory or data structures

removeQ()

-Parameters

- Queue - A pointer to the queue that will be altered
- PCB - A pointer to the PCB being removed from the Queue

-Return Values

Returns an integer representative of a success or failure state

-Implementation / Application

Removes a PCB from a queue and frees the memory associated simultaneously

isEmpty()

-Parameters

A pointer to the PCB that is the head of the queue

-Return Values

An integer that represents a fail or success state

-Implementation / Application

Determines whether a queue is empty or not

serial_poll()

- Parameters
    - Device dev - an input or output device that can read or display data to or from the user
    - char* / const char* buffer - a string of input or output to be handled by the comhandler after being processed by serial poll
    - Size_t len - The variable containing the size of the input being passed in or processed out
- Return Values
    - -1 - represents a failed state
    - 0+ - represents a unique state based on the input/output that can mean different things
- Implementation / Application

    Processes input from the keyboard to determine whether a command is being entered, a key is being pressed to manipulate the cursor, or a character is being deleted. Input is appropriately executed from here. For outputting, the serial polling function relies on calls to the comhandler.

disp()

- Parameters

  char* - outputs the passed string to the console

  Int - converts the int to a string to be outputted

- Return Values

  Void return type - takes input and produces display messages

- Implementation / Application

  Used to display simple strings or integers without formatting required. Intended to be used by other functions to show information or prompts to the user

Atoi()

-Parameters

A string to be converted to an integer value

-Return Values

An integer representation of the imputed string’s ascii value

-Implementation / Application

Used to convert characters to integers

Itoa()

-Parameters

An integer to be converted to a string

-Return Values

A character that is a numeric value

-Implementation / Application

Used to convert an integer to a numeric character

Isalpha()

-Parameters

An integer value of an ascii equivalent

-Return Values

Returns an integer that represents either a success or a failure

-Implementation / Application

Determines if a characters ascii value makes it an alphabetical letter

my_isdigit()

-Parameters

A character to be evaluated as a number

-Return Values

Returns true if it is a digit between 0-9 inclusive, false otherwise

-Implementation/Application

Used to determine characters whether or not characters within strings are integers or not.

isnumber()

-Parameters

A string of character to be evaluated as containing integers

-Return Values

Returns true if string only contains integer values

-Implementation/Application

Used to determine if a string can be substituted with an equivalent integer value

AvailablePCB()

-Parameters

None

-Return Values

PCB pointer to the next available PCB

-Implementation / Application

This is a helper method for sys_call(). It checks the front of the ready queue, if empty returns NULL. Otherwise it saves the next process, removes it from the queue, and returns a pointer to said process.

sys_call()

-Parameters

Cur - Context pointer of the current process

-Return Values

Returns a context pointer to the process being loaded

-Implementation / Application

Based on the context there are two operations that can happen. IDLE or EXIT, if the eax register is not set to either of those two, the register is set to -1 and the current context gets returned. There are two global pointers, one to the currently executing process and one pointing to the first context when sys_call() is called.

In the case of IDLE, the next available PCB is stored in a temporary variable using availablePCB(). If the process is a system process or NULL, the current context is returned. Else, the context of the current PCB is saved by updating the executing stackPtr, the PCB is added back into the queue, and the context of the next process is returned.

In the case of EXIT, the currently running PCB is deleted. If there are any available PCBs the next is loaded as IDLE, otherwise load the first context.

loadAlarm()

-Parameters

None

-Return Values

None

-Implementation / Application

Creates and sets context for the alarm process, which is then inserted into the ready queue and set to IDLE. The EIP of the process points to the **'alarm()'** function.

setAlarm()

-Parameters

Time structure to set alarm and message string to display.

-Return Values

None

- Implementation / Application

It will idle the alarm process until the current time is greater than the alarm time, then it will display the alarm message and exit the process. If the alarm is set to a time less or equal to the current time, the alarm will rollover to the next day.

displayAlarm()

-Parameters

Takes in alarm time and the time the alarm was set at.

-Return Values

None

-Implementation / Application

Displays alarm time and the time that it was set at.

kmain()

-Parameters

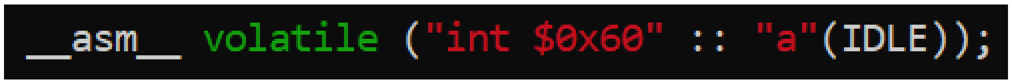
None

-Return Values

None

-Implementation / Application

This method isn’t created just altered to accommodate the IDLE processes. It creates two system processes when initializing the MPX modules. Command Handler is created as a process with the highest priority level (0). System IDLE Process is a process running at the lowest priority level (9). Finally the given code is uncommented to call Command Handler.



```
__asm__ volatile ("int $0x60" :: "a"(IDLE));
```

initalize_heap()

-Parameters

size_T size - The total size of the heap

-Return Values

None

-Implementation / Application

The function initializes the heap to the specified size. It creates the initial Memory Control Block (MCB) for the heap, sets its address, size, and status, and adds it to the MCB list. The function should be called before using allocate_memory() or free_memory() to ensure proper heap management.

allocate_memory()

-Parameters

size_T Size - the size of the block the user would like to allocate, not including the accompanying MCB.

-Return Values

void* : A pointer to the allocated memory block.

-Implementation / Application

The function is used to allocate a memory block to the specified size. The allocated memory block will be managed by a Memory Control Block (MCB). The function will search for an available MCB with enough memory and, if found, will allocate the requested memory block.

free_memory()

-Parameters

void* address- the address of the memory block to be freed

-Return Values

None

-Implementation / Application

The function is used to free a previously allocated memory block. The function searches for the corresponding MCB using the provided address, if found, it will free the memory block and update the MCB status.

Commands

vercom (Version Command)

- Parameters

  No input parameters

- Return Values

  Void return type - Outputs information

- Implementation / Application

  Displays on screen the current working version of G5 OS and the date it was published

shutdown

- Parameters

  No input parameters

- Return Values

  No return values, displays to console

- Implementation / Application

  Asks for confirmation from the user to shutdown, if confirmed then exits ComHand to kmain, allowing the system to shutdown. If the user objects to shutting down, returns back to the command line input.

gtime (Get time)

- Parameters

  No input parameters

- Return Values

  No return values, displays to console

- Implementation / Application

  Displays to the user the current time that has previously been set by the user

stime (Set time)

- Parameters

  No input parameters

- Return Values

  No return values

- Implementation / Application

  Allows the user to set the time and date based on a twenty-four hour clock

gdate (Get date)

- Parameters

  No input parameters

- Return Values

  No return values - displays to console

- Implementation / Application

  Returns the current date, month, and year to the user

sdate (Set date)

- Parameters

  No input parameters

- Return Values

  No return values

- Implementation / Application

  Allows the user to set the date, month and year

man (Manual)

- Parameters

  No input parameters

- Return Values

  Void return type - Outputs information

- Implementation / Application

  Displays to the screen the full list of implemented commands available to the user

createPCB()

-Parameters

None, takes in a name, class, and priority from the user post execution from the terminal.

-Return Values

None

-Implementation / Application

This function creates a Process Control Block (PCB) by calling pcbSetup() and then inserts it into the appropriate queue using pcbInsert(). The function prompts the user for the name, class, and priority of the PCB after being executed from the terminal.

deletePCB()

-Parameters

None, takes in the name of a PCB to be deleted.

-Return Values

None

-Implementation / Application

This function finds the requested PCB by its name and removes it from its queue. The PCB is deleted from the system.

blockPCB()

-Parameters

None, takes in the name of a PCB to be added to the blocked state

-Return Values

None

-Implementation / Application

This function adds a PCB to the blocked state and inserts it into the appropriate blocked queue.

unblockPCB()

-Parameters

None, takes in the name of a PCB to be added to the unblocked state

-Return Values

None

-Implementation / Application

Adds a PCB to the unblocked state and into an appropriate queue.

suspendPCB()

-Parameters

None, takes in the name of a process from the terminal to be suspended

-Return Values

None

-Implementation / Application

Puts a process in the suspended state and moves it to the appropriate queue.

resumePCB()

-Parameters

None, takes in the name of a process from the terminal to be suspended

-Return Values

None

-Implementation / Application

Puts a process in the not suspended state and moves it to the appropriate queue.

setPriority()

-Parameters

None, takes in a process name and a new priority to be assigned to the given PCB from the console

-Return Values

None

-Implementation / Application

Changes a process's priority and moves it to the appropriate place in the appropriate queue.

showPCB()

-Parameters

None, takes in a process name from the console

-Return Values

None

-Implementation / Application

Displays a process's name and all information related to the PCB.

showReady()

-Parameters

None

-Return Values

None

-Implementation / Application

Shows all information for all processes in the ready state.

showBlocked()

-Parameters

None

-Return Values

None

-Implementation / Application

Shows all information for all processes in the blocked state.

showAll()

-Parameters

None

-Return Values

None

-Implementation / Application

Shows all information for all PCBs regardless of state or priority.

loadR3()

-Parameters

None

-Return Values

None, displays to console

-Implementation / Application

Loads the R3 test processes from <processes.h> into a queued non-suspended, ready state. The names of the processes are “process_#” (# being the process number). The context of each process is saved at the top of the PCB stack.

alarm()

-Parameters

None, prompts user to input the alarm time and alarm message

-Return Values

None, displays to console

-Implementation / Application

Prompts the user to input the alert time with the format “hh:mm:ss” and a message to display when the alarm is set to go off. Validates the input time and sets the alarm. Validation is done upon the format of the time to make sure the separators are correctly given and to make sure the hours/minutes/seconds are valid values. If an invalid time is given an error message will display.

allocateMem()

-Parameters

None, prompts the user to input the size of the allocation request (in decimal)

-Return Values

Prints the console a confirmation message that the allocation was successful

-Implementation / Application

Prompts the user for input. Sanitize and validate the input. Then calls the allocate_mem() function to allocate the memory with the inputted size.

freeMem()

-Parameters

None, prompts the user to input the address of the memory block that is going to be freed

-Return Values

None

-Implementation / Application

Prompts the user for input. Sanitize and validate the input that it is a valid address. Then calls the free_memory() function with the inputted address

showAllocated()

-Parameters

None

-Return Values

None, displays the allocated mcbList to the console

-Implementation / Application

Iterates through the mcbList only displaying the MCB's that are allocated (status = 1)

showFree()

-Parameters

None

-Return Values

None, displays the free mcbList to the console

-Implementation / Application

Iterates through the mcbList only displaying the MCB’s that are free (status = 0)

Structures

T_symstruct

- Values

    Char* key - the name of a command able to be executed by the comhand

    Int val - the integer representation of the function location corresponding to the key

- Purpose

    Creates a table of locations that have a key and a physical address to be accessed by the command handler.

Queue

- Values

    Pcb front* - the pcb that comes directly before the current node, null if head

Pcb rear* - the pcb that comes immediately after the current node, null if tail

- Purpose

  A priority queue based around a linked list that has a pointer to its previous and its next pcb in the queue.

PCB (Process Control Block)

- Values

  char* pID - The name of the process

  Int state - 0 for ready, 1 for blocked

  Int class - 0 for user application, 1 for system process

  Int priority - 0(highest) - 9(lowest)

  Int status - 0 for suspended, 1 for not suspended

  pcb* next - a pointer to the next pcb in the queue

- Purpose

  To classify a process to be executed by the cpu and to store data related to how and when it should be executed.

Time

- Values

  hh - Hours

mm - Minutes

ss - Seconds

- Purpose

  Used to pass in alarm time to the setAlarm() function.

MCB (Memory Control Block)

- Values

  void* address- the base address of usable memory in the block (first bit after the MCB)

  size_T size - The size of the memory block, not including the MCB

  Int status - 0 for free block, 1 for allocated block

  mcb* next - a pointer to the next mcb in the mcbList

  mcb* prev - a pointer to the previous mcb in the mcbList

- Purpose

  The purpose of the memory control block is to display the start address and size of each memory block. It is positioned immediately before each memory block. While containing doubly linked pointers for the mcbList.

mcbList

- Purpose

This is the list of MCB’s to keep track of all the MCBs. Using a single list makes traversing significantly easier, and it is the approach the group took.