Codestorm OS

Programmers Manual

Creators:

Clayton Grubick,
Andrew Harper,
Jacob Liebau,
Christian Straub

Version 2.1

November 10th 2022

West Virginia University

Table of Contents

Module One:

Serial.c:

int *polling(char *buffer, int *count)

- Description: The polling function gathers keyboard input via the serial port using the technique of polling. The function handles alphanumerics, backspace, delete, arrow keys, carriage returns and new-lines.
- Parameter 1(*buffer): This pointer variable is used to store each character handled by the polling function.
- Parameter 2(*count): This pointer points to the size of the buffer.
- Return: This function returns count which is the size of the buffer.

Com_hand.c:

int comhand()

- Description: The command handler function is the primary way users interact with the operating system. This function uses the input gathered from polling and determines which command was entered by the user. It also provides a user interface that runs the command entered by the user. These commands are help, quit, version, time, and date.

Date.c:

int print_date()

- Description: Gets the dates for month/day/year from the registers, the register values are defined in date.h a swell as the function prototypes. Prints the date using sys_req(WRITE…);
- Return: Returns an integer

int print_time()

- Description: Gets the dates for the hours/minutes from the registers, prints the time using sys_req(WRITE,....);
- Return: Returns an integer

unsigned char decimal_to_bcd(int dec)

- Description: Converts an int into an unsigned char, this is done to allow the register values for the clock and date to be changed since they are stored in BCD.
- Return: returns an unsigned char

int set_date(char date[30]){

- Description: Sets the date after receiving a date string from com_hand.c then it processes the date into a string, int, and finally BCD and uses outb to write to the registers.
- Return: Returns an int, uses to check whether the date was successfully changed or not, if the return is an int of 0 it has failed in setting date, if the result is 1 it succeeded in setting date

int set_time(char time[30]){

- Description: Sets the time after getting a string from the com_hand.c then it processes the time into a string, int and finally BCD and uses outb to write to the registers.
- Return: Returns an int, uses to check whether the time was successfully changed or not, if the return is an int of 0 it has failed in setting time, if the result is 1 it succeeded in setting dtimeate

int bcd_to_decimal(unsigned char bcd)

- Description: Take in an unsigned char and convert to an integer, by taking the upper and lower nibbles of the byte, and converting each to an int.
- Return: Returns an int of the BCD

void swap(char *x, char *y)

- Description: This function swaps the char values pointed at by x and y.
- Parameter 1(*x): This pointer points at the first value to be swapped.
- Parameter 2(*y): This pointer points at the second value to be swapped.

char* reverse(char *buffer, int i, int j)

- Description: This function reverses the contents of the buffer array sent to it.
- Parameter 1(*buffer): This is a pointer to the array that is being reversed.
- Parameter 2(i): This int is used as an index for reversing the array. This int starts at the beginning of the array
- Parameter 3(j): This int is used as an index for reversing the array. This int starts at the end of the array.
- Return: This function returns the reversed array.

char* itoa(int value, char* buffer, int base)

- Description: This function converts an integer into a string.
- Parameter 1(value): The integer that is being converted into a string.
- Parameter 2(buffer): The pointer to the array that stores the string that was converted.
- Parameter 3(base): The conversion base.
- Return: This function returns the converted string.

Help.c:

int help()

- Description: This function writes the contents of the help command to the console.

Module Two

Pcb.c:

Struct pcb* allocate_pcb()

- Description: Uses sys_alloc_mem() to allocate memory for a new PCB including the stack.
- Returns: This returns a pointer to a newly allocated PCB.

Int free_pcb(struct pcb* pcbtofree)

- Description: Uses sys_free_mem() to free all memory associated with a PCB
- Parameter(pcbtofree): A pointer to the PCB to free.
- Returns: This returns integer 0 to indicate success, otherwise error.

Struct pcb* setup_pcb(char buff[30])

- Description: Uses allocate_pcb() to allocate a new PCB, initializes it with data provided and sets the state to Ready, Not-Suspended.
- Parameters: The process name, process class, and the process priority. The format to enter is as follows, Name,Class,Priority
- Returns: A pointer to the initialized PCB on success, or returns NULL on error allocating, initializing, or invalid parameter.

Struct pcb* find_pcb(char *find_name)

- Description: Locates the name of a process inside of any queue.
- Parameter(find_name): The name of the process that the user is trying to find.
- Return: This returns a pointer to the name of the process if successful. Otherwise, it returns NULL if the name of the process does not exist.

Void init_queues()

- Description: Initializes each of the four queues heads, and tails to NULL and sets the pcb count to 0

Int insert_pcb(struct pcb* pcb_insert)

- Description: Inserts a process into the appropriate queue based on its state and priority.
- Parameter(pcb_insert): A pointer to the PCB to enqueue.

Int remove_pcb(struct pcb* target)

- Description: Removes a PCB from its current queue
- Parameter(target): A pointer to the PCB to remove.
- Return: This returns 1 if successful and 0 if it is unsuccessful

Int block_pcb(char *block_name)

- Description: Moves a PCB to the blocked queue.
- Parameter(block_name): The name of the process the user wants to have blocked.
- Return: This returns 1 if successfully blocked and 0 if it is unsuccessfully blocked.

Int unblock_pcb(char *unblock_name)

- Description: Moves a PCB to the ready queue.
- Parameter(unblock_name): The name of the process the user wants to have unblocked (ready).
- Return: This returns 1 if successfully unblocked and 0 if it is unsuccessfully unblocked.

Int is_empty(struct queue* queuePtr)

- Description: Checks whether a queue is empty or not returns a value depending on this.
- Parameter(queuePtr): A pointer to a queue
- Return: Returns a 1 if the queue is empty and a 0 if it contains any PCBs

Void showpcb(struct pcb *pcbtoshow)

- Description: Displays a process's: Name, Class, State, Suspended Status, Priority.
- Parameter(pcbtoshow): A pointer to the PCB to display.

Void suspend_pcb()

- Description: Changes a PCB to the suspended state and moves it to the appropriate queue

Void resume_pcb(char *name)

- Description: Changes a PCB to the not suspended state and moves it to the appropriate queue.
- Parameter(name): A pointer to the name of the PCB to resume.

Void show_ready_pcb()

- Description: For all PCBs in the ready queue it displays the process's name, class, state, suspended status, and priority.

Void show_blocked_pcb()

- Description: For all PCBs in the blocked queue it displays the process's name, class, state, suspended status, and priority.

Void show_all_pcb()

- Description: For all PCBs in every queue it displays the process's name, class, state, suspended status, and priority.

Void setpriority(char *name, char *priority)

- Description: Changes a process's priority, and moves it to the appropriate place in the appropriate queue. Prints a message on success.
- Parameter 1(name): The name of the existing process to set priority of.
- Parameter 2(priority): and the value to set new priority to ranging from 0-9.
- Error Checking: This prints error message when name or priority is not valid.

Module Three:

Interrupts.c

Uint32_t* sys_call(context* registers)

- Description: Performs a context switch when called.
- Parameter(registers): A pointer to a struct representing the context.
- Return: Returns a 32-bit integer that is the stack pointer of the new process.

Void show_all_pcb()

- Description: For all PCBs in every queue it displays the process's name, class, state, suspended status, and priority.

Procsr.c

Void proc1()

- Description: Process to be loaded into memory and queued in a non-suspended ready state with specified name and priority.

Void proc2()

- Description: Process to be loaded into memory and queued in a non-suspended ready state with specified name and priority.

Void proc3()

- Description: Process to be loaded into memory and queued in a non-suspended ready state with specified name and priority.

Void proc4()

- Description: Process to be loaded into memory and queued in a non-suspended ready state with specified name and priority.

Void proc5()

- Description: Process to be loaded into memory and queued in a non-suspended ready state with specified name and priority.

Usercom.c

Struct pcb* load_proc1()

- Description: Creates a new user PCB named "p1", with a priority of 1. It also sets the context for the PCB as well.
- Return: It returns the created PCB.

Struct pcb* load_proc2()

- Description: Creates a new user PCB named "p2", with a priority of 1. It also sets the context for the PCB as well.

- Return: It returns the created PCB.

Struct pcb* load_proc3()

- Description: Creates a new user PCB named “p3”, with a priority of 1. It also sets the context for the PCB as well.
- Return: It returns the created PCB.

Struct pcb* load_proc4()

- Description: Creates a new user PCB named “p4”, with a priority of 1. It also sets the context for the PCB as well.
- Return: It returns the created PCB.

Struct pcb* load_proc5()

- Description: Creates a new user PCB named “p5”, with a priority of 1. It also sets the context for the PCB as well.
- Return: It returns the created PCB.

Module Five:

mcb.c:

Void init_heap (size_t bytes)

- Description: Allocates all memory available to the memory manager as a single large block.
- Parameter(bytes): The requested size, plus room for the MCB.
- Error Checking: This prints an error message when the heap cannot be initialized.

Void alloc_mem (uint_32t num_bytes)

- Description: Allocates memory from the heap.
- Parameter(num_bytes): The requested size.
- Return: Returns void pointer to the allocated memory. Can also return NULL, or print an error message.

Int free_mem (void* target_addr)

- Description: Frees the allocated memory.
- Parameter(target_addr): A pointer to the block that is set to be freed.
- Return: Returns an int representing either a 0 (success) or a 1 (failure).

Void show_all_free()

- Description: Displays the free MCB’s and their addresses.

Void show_all_allocated()

- Description: Displays the allocated MCB’s and their addresses.