Apr 11, 2025

Programmer's Manual

CS 450: WStem

Table of Contents:

Introduction

This manual is intended for programmers working on or viewing the WStem CS 450 project. It provides an overview of the data types and functions created by the team.

R1

int serial_poll(device dev, char *buffer, size_t len)

Author: Kylie Burke
Description: Gets keyboard input and stores it in a buffer or processes
the input. This function returns the amount of bytes read into the Buffer.

void shiftLeft(device dev, char *buffer, int mode)

Author: Kylie Burke
Description: This method shifts the characters in the buffer to the left to remove one from a certain index. This is determined by the int mode that uses this method differently for backspace and delete keys.

void print(const char* msg)

Author: Kylie Burke

Description: This method prints a constant string to the terminal. For example, it is used to print times and dates for the user to see.

char* itoa(int n)

Author: Kylie Burke

Description: This method takes in an integer (between the values of -2,147,483,648 to +2,147,483,647) and returns the number as a string. This can be used for time and dates to assist with printing to the terminal with print().

int isDigit(char c)

Author: Kylie Burke

Description: Checks if a character is a number between 0-9.

int isNumeric(char* str)

Author: Kylie Burke

Description: Checks if a string consists of only integers.

char* concat(const char* str1, const char* str2)

Author: Olivia Gottlieb

Description: Combines two strings into one by concatenating one string to the end of another string. This concatenated string is returned. Dynamic allocation was utilized for pointer privileges in order to increment the characters and return the output.

int get_hour()

Author: Olivia Gottlieb

Description: This method gets the decimal representation of the hour and puts it to the terminal by reading in the hour and pointing to the correct place in memory. Finally, returning the converted decimal minutes.

Apr 11, 2025

int get_min()

Author: Olivia Gottlieb

Description: This method gets the decimal representation of the minute and puts it to the terminal by reading in the minutes and pointing to the correct place in memory. Finally, returning the converted decimal minutes.

int get_sec()

Author: Olivia Gottlieb

Description: This method gets the decimal representation of the second; puts it to the terminal by reading in the seconds and pointing to the correct place in memory. Finally, returning the converted decimal seconds.

char* get_time()

Author: Olivia Gottlieb

Description: This method gets the time in hours, minutes, and seconds. Calls get_hour(), get_min(), get_sec() and converts them to strings. Concatenates the hour, minutes, and seconds and returns the string formatting in HH:MM:SS layout.

void set_time(int hour, int min, int sec)

Author: Olivia Gottlieb

Description: This method sets the time in hours, minutes, and seconds by putting them in registers for future accessibility. It writes an index to the RTC index register with outb(), and writes a new value to that index with the user input.

int days_in_month(int month, int year)

Author: Olivia Gottlieb

Description: This method sets the amount of days allocated to each month. It uses a switch case to decipher the amount of days in a given month and year if it is February, which is later used.

Month	Days	
January	31	
February	28	Leap years 29
March	31	
April	30	
May	31	
June	30	
July	31	
August	31	
September	30	
October	31	
November	30	
December	31	

int get_year()

Author: Olivia Gottlieb

Description: This method gets a decimal representation of the year and puts it to the terminal. It uses the year in binary, reads it in using outb(), and converts it to decimal.

int get_month ()

Author: Olivia Gottlieb

Description: This method gets a decimal representation of the month and puts it to the terminal. It uses the month in binary, reads it in using outb(), and converts it to decimal.

int get_day ()

Author: Olivia Gottlieb

Description: This method gets a decimal representation of the day and puts it to the terminal. It uses the day in binary, reads it in using outb(), and converts it to decimal.

char* get_date ()

Author: Olivia Gottlieb

Description: This method sets the date in years, months, and days by putting each value in separate registers for future accessibility. It uses the previous get_year(), get_month(), get_day() functions, converts the integers to strings, concatenates the strings, and returns the concatenated string date.

void set_date ()

Author: Olivia Gottlieb

Description: This method sets the amount of days allocated to each month. It writes an index to the RTC index register with outb(), and writes a new value to that index with the user input.

void get_time_cmd ()

Author: Olivia Gottlieb

Description: This method is the user level of get_time() where interaction with the user for getting the current time in UTC occurs. Finally, prints to the terminal.

void set_time_cmd ()

Author: Olivia Gottlieb

Description: This method is the user level of the set_time() where interaction with the user for setting/allowing input of a time in HH:MM:SS format occurs.

void get_date_cmd ()

Author: Olivia Gottlieb

Description: This method is the user level of the get_date() where interaction with the user for getting the current date occurs.

void set_date_cmd ()

Author: Olivia Gottlieb

Description: This method is the user level of the set_date() where interaction with the user for setting/allowing input of a date in DD/MM/YY format occurs.

void version_command(void)

Author: Nada Mikky

Description: Prints the current version of MPX on the terminal. It uses the sys_req call to write a fixed version string to COM1. It helps users quickly verify which version and build of MPX is running

void help_command(const char *command)

Author: Nada Mikky

Description: Displays help instructions for using MPX commands. When no specific command is provided, it prints a complete list of commands with simple descriptions.

void shutdown_command()

Author: Nada Mikky

Description: Asks the user if they want to shut down MPX. It prints a confirmation prompt and reads the response. If the answer is yes, it prints a shutdown message and stops the system's main loop

void comhand(void)

Author: Nada Mikky

Description: This function is the main command loop for MPX. It continuously prompts the user for commands, reads input, and checks the command. Depending on the input, it calls the correct function or displays an error message.

R2

void itoa_pcb(int num, char *str) NOT USED AS OF R4

Author: Nada Mikky

Description: Helper function that converts an integer into a string representation and stores the result in the provided character buffer.

void print_pcb(struct pcb *PCB)

Author: Nada Mikky

Description: Prints the details of the given PCB, including its name, state, and priority, to the terminal using sys_req. If the PCB pointer is NULL, it outputs an error message.

void help_command(const char *command)

Author: Nada Mikky

Description: Displays help instructions for using the available pcb commands. When no specific command is provided, it prints a complete list of commands with simple descriptions.

void create_pcb(const char *name, int class, int priority)

Author: Nada Mikky

Description: Creates a new Process Control Block (PCB) with the specified name, class, and priority. Validates inputs (name, class, and priority), ensures the PCB name is unique, and inserts the new PCB into the appropriate process queue.

void delete_pcb(const char *name)

Author: Nada Mikky

Description: Deletes a PCB identified by the given name. It verifies that the PCB exists and is not a system process, then removes it from its queue and frees its allocated memory.

void block_pcb(const char *name)

Author: Nada Mikky

Description: Blocks the PCB with the specified name by removing it from its current queue and inserting it into the blocked queue. It also checks that the PCB exists and is not already blocked.

unblock_pcb(const char *name)

Author: Nada Mikky

Description: Unblocks the PCB identified by the given name by moving it from the blocked queue to the ready queue. Validates that the PCB is currently blocked before proceeding.

void suspend_pcb(const char *name)

Author: Nada Mikky

Description: Suspends a PCB with the provided name. Ensures the PCB exists, is not a system process, and is not already suspended, then marks it as suspended and reinserts it into the proper suspended queue.

void resume_pcb(const char *name)

Author: Nada Mikky

Description: Resumes a previously suspended PCB identified by name by marking it as not suspended and reinserting it into the ready queue. Validates that the PCB exists and is in a suspended state.

void set_pcb_priority(const char *name, int priority)

Author: Nada Mikky

Description: Changes the priority of the PCB with the specified name. Validates the new priority and, if the PCB is in the ready state, repositions it in the queue based on the new priority value.

void show_pcb(const char *name)

Author: Nada Mikky

Description: Displays detailed information (such as name, state, and priority) for the PCB identified by the given name by the user.

void show_ready(void)

Author: Nada Mikky

Description: Iterates through and displays all PCBs in the ready queue.

void show_blocked(void)

Author: Nada Mikky

Description: Iterates through and displays all PCBs in the blocked queue

void show_all(void)

Author: Nada Mikky

Description: Displays all PCBs by first listing those in the ready queue and then those in the blocked queue.

struct pcb

Author: Olivia Gottlieb

Description: Creates the pcb structure where the crucial variables including the pid (process id), name of the process, stack size, a pointer to the stack, state, priority, suspension updates, class, and prev/next pointers are initialized.

struct pcb* allocate(void)

Author: Olivia Gottlieb

Description: This method allocates memory to the associated new pcb structure by initializing a new pcb with the name, class, and priority using sys_alloc_mem(). If memory fails to allocate because of some error with either the name being null or the priority is not within the valid range (0-9), then it returns NULL. It then allocates memory for the PCB and initializes its fields, including setting the process state to READY, marking it as NOT_SUSPENDED, and assigning the given priority and class. The method returns a pointer to the created pcb or NULL if there is a problem with the allocation.

int pcb_free(struct pcb* new_pcb)

Author: Olivia Gottlieb

Description: This method returns the new pcb structure, unless if the pointer is null, then return -1.

struct pcb* pcb_setup(const char *name, int priority, int class)

Author: Olivia Gottlieb

Description: This method initializes a new process control block with the specified name, priority, and class. If there is a problem with the input because of some error with either the name being null or the priority is not within the valid range (0-9), then it returns NULL. If all inputs are valid, it allocates memory for a new PCB using pcb_allocate(). Then, it initializes the PCB's fields by marking it at NOT_SUSPENDED, setting its state to READY, and assigning the given priority and class. This method returns a pointer to the new PCB.

struct pcb* pcb_find(const char *name)

Author: Olivia Gottlieb

Description: This method attempts to search for a pcb by a provided name in all process queues, including ready, blocked, suspended ready, and suspended blocked queues. It iterates through each queue, comparing the name of each PCB to the given name using strcmp(). If a match is found, it returns a pointer to the PCB. Otherwise, the method returns NULL.

struct pcb_insert(struct pcb* new_pcb)

Author: Olivia Gottlieb

Description: Depending on the state and suspension status, this method will insert a pcb into the appropriate queue. First, it will determine which queue the pcb belongs to (ready, blocked, suspended ready, or suspended blocked), unless the queue is empty, in which case, the pcb is placed in the first elemental position. In all other cases, the method inserts the pcb in the correct position based on priority. Next and prev pointers of nodes are taken into account throughout this process considering the properties of doubly linked lists.

struct pcb_remove(struct pcb* new_pcb)

Author: Olivia Gottlieb

Description: This method removes a PCB from the current queue, but it does not free the allocated memory. It unlinks the requested pcb from its queue by fixing the next and prev pointers to exhibit the updates to the queue. If the PCB is the first element in the queue, the queue's head is updated accordingly. The method returns 0 on success and -1 if the input PCB is NULL.

char* get_PCB_name()

Author: Kylie Burke

Description: This method gets the PCB name from the user by using I/O. It will return a validated name as a string.

int get_PCB_class()

Author: Kylie Burke

Description: This method gets the PCB class from the user by using I/O. It will return either a 0 for a system process or a 1 for a user process.

int get_PCB_priority()

Author: Kylie Burke

Description: This method gets the PCB priority from the user by using I/O. It will return an integer between 0 (high) and 9 (low).

int get_century(void)

Author: Kylie Burke

Description: This method gets the century from the century register and returns it in decimal format.

R3

sys_call_isr

Author: Kylie Burke

Description: Saves the content of general purpose registers, calls sys_call, and restores the content of the general purpose registers.

struct context

Author: Kylie Burke

Description: Holds the context of currently running processing including Segment Registers, Status Control Registers, and General Purpose Registers.

void load_proc1_suspended()

Author: Nada Mikky

Description: Creates a PCB, sets it as READY, and assigns execution to proc1

void load_proc2_suspended()

Author: Nada Mikky

Description: Creates a PCB, sets it as READY, and assigns execution to proc2

void load_proc3_suspended()

Author: Nada Mikky

Description: Creates a PCB, sets it as READY, and assigns execution to process 3

void load_proc4_suspended()

Author: Nada Mikky

Description: Creates a PCB, sets it as READY, and assigns execution to proc4

void load_proc5_suspended()

Author: Nada Mikky

Description: Creates a PCB, sets it as READY, and assigns execution to proc5

void yield(void) (REMOVED AS OF R4)

Author: Nada Mikky

Description: causes command handler to yield the CPU

void load_r3()

Author: Nada Mikky

Description: Loads all R3 processes into memory in a READY state

void load_r3_suspended()

Author: Nada Mikky

Description: Loads all R3 processes into memory in a suspended state

void load_proc1()

Author: Nada Mikky

Description: Creates a PCB for proc1, sets it as READY, assigns execution to proc1, and initializes its context.

void load_proc2()

Author: Nada Mikky

Description: Creates a PCB for proc2, sets it as READY, assigns execution to proc2, and initializes its context.

void load_proc3()

Author: Nada Mikky

Description: Creates a PCB for proc3, sets it as READY, assigns execution to proc3, and initializes its context.

void load_proc4()

Author: Nada Mikky

Description: Creates a PCB for proc4, sets it as READY, assigns execution to proc4, and initializes its context.

void load_proc5()

Author: Nada Mikky

Description: Creates a PCB for proc5, sets it as READY, assigns execution to proc5, and initializes its context.

void load_r3_user()

Author: Nada Mikky

Description: Loads all R3 processes into memory in a READY state, initializes their PCBs, and confirms successful loading.

context *sys_call(context *cur_context)

Author: Olivia Gottlieb

Description: Manages system calls by context switching. Handles cases like switching processes to IDLE and replacing it with the next ready process in queue, terminating a currently running process if it has been completed, and system calls for reading and writing, but returning errors if the input is incorrect form.

R4

void alarm(void)

Author: Nada Mikky and Olivia Gottlieb

Description: Calls the `setAlarm` function

int checkTime()

Author: Nada Mikky and Olivia Gottlieb

Description: Compares the specified alarm time with the current system time.

void setAlarm(char *time, char *msg)

Author: Nada Mikky and Olivia Gottlieb

Description: Sets an alarm by creating a new alarm process. It reads the time and message from the user input, then invokes the setupAlarm function to initialize the alarm with the provided message.

void startAlarm(char alarmCount)

Author: Nada Mikky and Olivia Gottlieb

Description: Handles the process of starting the alarm. It prints a message to COM1 and then exits the process. This function is called when the alarm process is triggered.

void setupAlarm(int hr, int min, int sec, char *msg)

Author: Nada Mikky and Olivia Gottlieb

Description: Creates and sets up an alarm by initializing an Alarm struct with the provided time values. It then creates a new process control block (PCB) for the alarm and sets the alarm process to a ready state. The alarm process is then inserted into the process queue, and execution continues at the startAlarm function once the alarm time is triggered.

void load_comhand(void)

Author: Kylie Burke

Description: Creates a system process for the command handler (comhand) with the highest priority (0).

void load_sys_idle(void)

Author: Kylie Burke

Description: Creates a system process for the system idle process (sys_idle_process) with the lowest priority (0).

int isdigit(char c) (REMOVED AS OF R5; see isDigit() in R1)

Author: Olivia Gottlieb

Description: Checks if the alarm input is a valid number and within range.

int get_numerical_input (const char *prompt, int min_val, int max_val, int length) (Previously get_alarm_input)

Author: Olivia Gottlieb

Description: prompts the user for hour minute and second input to set an alarm. This function collects and validates the input ensuring it is a numerical value that is within range.

R5

void itoa(char *buff, int n) -*updated***-**

Author: Nada Mikky

Description: Converts a base 10 integer to its string representation and stores it in the provided buffer.

char *concat(const char *str1, const char *str2, char *buffer) -*updated***-**

Author: Nada Mikky

Description: Concatenates two null-terminated strings (`str1` and `str2`) into the provided pre-allocated buffer. This avoids dynamic memory allocation and helps prevent memory leaks.

void allocate_mem(size_t allocateSize)

Author: Nada Mikky

Description: Allocates a block of memory from the heap with the specified size and calls allocate_memory() to do so.

void free_mem(void *address)

Author: Nada Mikky

Description: Frees a previously allocated block of heap memory.

void show_allocated_memory(void)

Author: Nada Mikky

Description: Displays details of all currently allocated memory blocks in the heap.

void show_free_memory(void)

Author: Nada Mikky

Description: Displays details of all currently free memory blocks in the heap.

int mathPower(int num, int expo)

Author: Nada Mikky

Description:. Calculates the power of a number and returns the result of a number raised to the power of an exponent

int decimalToHex(unsigned int dec);

Author: Nada Mikky

Description:. Convert a decimal number to hexadecimal and print it. Returns 0 on success.

int hexToDecimal(char *hex, unsigned int hexSize);

Author: Nada Mikky

Description: Converts a hexadecimal string to a decimal number.

void load_process(int num, int priority)

Author: Kylie Burke

Description: Loads a R3 test process individually in a suspended and ready state. During comhand, the user is prompted for which process to load and at what priority.

void initialize_heap(size_t heapSize)

Author: Kylie Burke

Description: Initializes a heap of memory as one, large contiguous block that will be allocated in the future.

void* allocate_memory(size_t allocateSize)

Author: Kylie Burke
Description: Allocates a block of memory out of the available free memory. It finds the first block that can fit the data and creates an MCB for the newly allocated memory and another MCB for the remaining free memory affected.

int free_memory(void *ptr)

Author: Kylie Burke
Description: Receives the pointer of the start address of the MCB to free. It removes it from the allocated memory and adds it back to the free list. If the newly freed memory has neighboring free memory, it combines it to make one contiguous block.

struct mcb *get_free_memory_list(void)

Author: Kylie Burke
Description: Gets the free memory list.

struct mcb *get_allocated_memory_list(void)

Author: Kylie Burke
Description: Gets the allocated memory list.

void set_free_memory_list(struct mcb *list)

Author: Kylie Burke
Description: Sets the free memory list to the list of MCBs provided in the parameter.

void set_allocated_memory_list(struct mcb *list)

Author: Kylie Burke
Description: Sets the allocated memory list to the list of MCBs provided in the parameter.

void* sys_alloc_zeroed_mem(size_t num, size_t size)

Author: Kylie Burke

Description: Instead of using sys_alloc_mem, this is used to make sure the pointer created with sys_alloc_mem, the length (num), and the size of the type being put into the memory is empty prior to use. If it isn't, it clears the pointer and returns that.

R6

void enqueue_iocb(struct iocb *iocb, struct iocb **head)

Author: Olivia Gottlieb

Description: Enqueuing uses the first come first served methodology which implies the queue is in order of arrival requests. Each device has its own queue, and if a device is busy, the request gets enqueued and the process is blocked temporarily.

struct iocb* dequeue_iocb(struct iocb **head)

Author: Olivia Gottlieb

Description: Dequeuing pulls the next ready-to-run IOCB from the front of the queue which shifts the pointers to the right one place allowing for a seamless transition for the next-received IOCB to be implemented when the device is available.

void block_pcb(const char *name) *-removed for R6-*

Author: Nada Mikky

Description: Blocks the PCB with the specified name by removing it from its current queue and inserting it into the blocked queue. It also checks that the PCB exists and is not already blocked.

unblock_pcb(const char *name) *-removed for R6-*

Author: Nada Mikky

Description: Unblocks the PCB identified by the given name by moving it from the blocked queue to the ready queue. Validates that the PCB is currently blocked before proceeding.

void check_event_flags()

Author: Nada Mikky

Description: Checks and updates the event flags for all devices. Responsible for monitoring the event flags of devices and updating their status based on the current state of I/O operations

struct dcb *get_dcb_by_device(int device_id)

Author: Nada Mikky

Description: Returns the DCB (Device Control Block) for a given device ID. If the device is invalid or uninitialized, returns NULL

int serial_open(device *dev, int speed)

Author: Nada Mikky

Description: A function that opens a serial port. It initializes the serial port and prepares it for communication.

int serial_close(device dev)

Author: Nada Mikky

Description: Closes the given serial port by disabling interrupts, freeing its DCB, and clearing it from the global DCB table

context *sys_call(context *cur_context) -*updated*-

Author: Nada Mikky

Description: Handles system calls like IDLE, EXIT and I/O. Updates include I/O scheduling, device validation and read/write operations.

struct iocb

Author: Kylie Burke

Description: It stores which process made the request, what device it's for, the operation type (read or write), the buffer and its size, and a pointer to the next request in the queue. Used by the I/O scheduler to manage and track pending I/O tasks.

struct iocb **headQCOM1 = NULL;
struct iocb **headQCOM2 = NULL;
struct iocb **headQCOM3 = NULL;
struct iocb **headQCOM4 = NULL;

Author: Kylie Burke
Description: Structs used to manage the I/O Control Blocks (IOCBs) for each device

void io_scheduler(op_code operation, device dev, char *buffer, size_t size, struct pcb* pcb)

Author: Kylie Burke
Description: Handles I/O system calls. If the device is free, the operation starts right away. If not, the request is queued and the process is blocked.

void iocb_free(struct iocb *iocb)

Author: Kylie Burke
Description: Frees the memory used by a completed or canceled I/O request.

struct iocb **getHeadQ(device dev)

Author: Kylie Burke
Description: Returns the head pointer for the queue of the given device (COM1- COM4)

void allocateIOCBHeads(void)

Author: Kylie Burke
Description: Sets up memory for each device's I/O queue head so requests can be added

void freeIOCBHeads(void)

Author: Kylie Burke

Description: cleans up the memory used for the I/O queue heads when they're no longer needed.

int serial_read(device dev, char *buf, size_t len);

Author: Kylie Burke

Description: Starts a read operation on the given device. Transfers characters from the ring buffer to the user buffer and sets the event flag when done.

int serial_write(device dev, char *buf, size_t len)

Author: Kylie Burke

*Description:*Starts a write operation by sending the first character and enabling write interrupts. Updates the DCB with buffer info.

void serial_interrupt(void)

Author: Kylie Burke

Description: Handles serial port interrupts. It checks what kind of interrupt happened and passes it to the right function either for input or output

void serial_input_interrupt(struct dcb *dcb)

Author: Kylie Burke

Description: Handles incoming characters. If reading, places them in the user buffer; otherwise, stores them in the ring buffer.

void serial_output_interrupt(struct dcb *dcb)

Author: Kylie Burke

Description: Sends characters from the output buffer one at a time. When finished, marks the operation complete and disables write interrupts.

int device_id_to_int(device dev)

Author: Kylie Burke

Description: Converts a device enum (COM1 -COM4) to its corresponding integer index. Returns -1 for invalid devices.

int valid_device(device dev)

Author: Kylie Burke

Description: Checks if a device is a valid COM port. Returns 0 if valid, and -1 if not

device device_int_to_dev(int i)

Author: Kylie Burke

Description: Converts an integer to its corresponding device enum. Returns -1 if the input is invalid

int pollChar(struct dcb *dcb, char c)

Author: Kylie Burke

Description: Processes a single input character: handles newlines, backspaces, and normal characters. Updates the input buffer accordingly.

void freeDCB(struct dcb *dcb)

Author: Kylie Burke

Description: Frees the memory allocated for a DCB if it exists.

void clearAllPCBs(void)

Author: Kylie Burke

Description: Clears all PCBs from the ready, blocked, suspended ready, and suspended blocked queues.

void version_help(void)

Author: Kylie Burke

Description: prints out the version command description and usage for the user

void help_help(void)
Author: Kylie Burke
Description: prints out the description of the help command and how to use it

void shutdown_help(void)
Author: Kylie Burke
Description: Displays the description and usage of the shutdown command

void get_time_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the get time command to display the current system time.

void set_time_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the set time command for modifying the system time.

void get_date_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the get date command to display the current system date

void set_date_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the set_date command for modifying the system date.

void delete_pcb_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the delete_pcb command to remove a specified PCB

void suspend_pcb_help(void);
Author: Kylie Burke

Description: Prints the description and usage of the suspend pcb command to suspend a specified PCB

void resume_pcb_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the resume pcb command

void set_pcb_priority_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the set pcb priority command to change a PCB’s priority.

void show_pcb_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the show pcb command to display a specific PCB’s information

void show_ready_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the show ready command

void show_blocked_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the show blocked command

void show_all_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the show all command to display all PCBs

void load_r3_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the load r3 command to load R3 processes

void load_r3_suspended_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the load r3 suspended command

void load_proc1_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the load proc 1 command

void load_proc2_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the load proc 2 command

void load_proc3_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the load proc 3 command

void load_proc4_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the load proc 4 command

void load_proc5_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the load proc 5 command

void alarm_help(void);
Author: Kylie Burke
Description: Prints out the description and usage of the set alarm command, which sets an alarm

void show_allocated_memory_help(void);
Author: Kylie Burke

Apr 11, 2025

Description: Prints out the description and usage of the show allocated memory command to display all allocated memory blocks

void show_free_memory_help(void);

Author: Kylie Burke
Description: Prints out the description and usage of the show free memory command to display memory blocks

void notes_help(void);

Author: Kylie Burke
Description: prints out notes about troubleshooting tips and important system behavior reminders for the user