Programming Assignment #2

Due Jun 21 by 5pm **Points** 50 **Submitting** a file upload
Available after Jun 8 at 10pm

This page last modified 8 Jun, 2021

You shall implement a Microshell, "MSH". MSH shall provide the following functions:

1. Provide a shell-like interface for launching new programs:

When MSH starts running, it will print a prompt "cssc0000% " (replace cssc0000 with your username) and then wait for the user to type in a file name. Note, MSH will NOT have any shell built-in functions (such as cd, setenv, printenv, bg, fg, etc), with the exception of #5 (below).

2. When the user enters a filename and hits the "enter" key, MSH then reads the filename entered and determines if the file is an executable file. If it isn't, MSH will print a useful error message to the user and then return to #1 above (display a new prompt and wait for user input). If it is an executable file, then MSH will create a new process and run this program in the new process.
3. Note – you will need to determine whether the filename is a fully qualified path name or if the file needs to be searched for (fully qualified pathnames begin with a slash (i.e. "/"), use the correct variant of 'exec' to do this.
1. MSH will support users creating two processes with a pipe for the two processes to use to communicate with each other. For example, the following is legal input:

"cssc0000% ls | sort"

In this example, MSH will create two processes and a pipe for them to use for communication between the two processes. In this example, the first process (ls program) would send its output to the second process (sort program) which would read from the pipe.

5. Okay, there is one builtin function you need to implement (actually three, counting the pipe in the previous requirement and the launch a new program), if the user types "exit" (then hits the "enter" key), then your program shall gracefully close itself down and terminate.

NOTE - The ONLY valid input formats are the following (gracefully reject/don't accept all others):

- msh% executablefilename
- msh% executablefilename argument <- (only one argument, no more than one)
- msh% executablefilename | executablefilename
- msh% executablefilename | executablefilename | executablefilename <- (any number of pipes may be constructed on the command line)
- msh% exit

Your program will be tested by compiling it and executing it on edoras using these features and some bad input.

Your program shall be written such that it compiles and executes cleanly when using the gcc/g++ compiler on edoras. You shall create a sub-directory named "a2" in your home directory. In a2, you shall place your source files (multiple source files are required), your header file, your Makefile, and a README file (follow instructions from assignment #1 for the README file). Additionally, identify in your README file who worked on which lines of code in this project (if you used Agile/Pair programming state who was writing and who was providing input for each function/method). Your source files SHALL CONTAIN sufficient comments for making the source easy to read. Points will be taken off for poorly (or non) commented source. Your main() should be a small function (it should look like a "table of contents" for your program). Name the executable "msh".

- Create ~/a2 by hand.
- Create multiple c/c++ source files, an include file, a Makefile, and a README file. Put them into ~/a2.
- The Makefile shall create an executable by the name of msh in the same directory (~/a2).
- The system call "system()" will NOT be allowed

TURNING IN YOUR WORK (only one of you turns in the project! Make sure both names and REDIDs are in all files!):

Make sure that all of your files (all source files, Makefile, README file, test files, etc) are in the a2 sub- directory of the class account

Additionally, create a tarball (tar file) or zip file and attach that file (upload it) under Assignment Submission in Assignment #2 in Canvas.

SUGGESTION: check out my sample source files and Makefile posted on Blackboard.

SAMPLE OUTPUT: Note, this is an interactive program, you can enter commands like you do in your bash shell you logged into on edoras, this sample tests key features of this assignment: msh_sample.txt

msh rubric

Criteria	Ratings		Pts
README File	**0 pts** **Minimum Points** README exists but contains minimal info	**5 pts** **Full Points** README file includes content as specified in the README format doc	5 pts
Makefile	**0 pts** **Minimum Points** Minimal or no Makefile	**5 pts** **Full Points** Makefile correctly compiles the project using at least two rules: (1) make the project and (2) clean up files	5 pts
Command Line fn	**0 pts** **Minimum Points** Displays username and waits but fails to perform any work (correctly)	**15 pts** **Full Points** Program displays username, waits for command, parses correctly and returns to wait for next command. The exit builtin works correctly	15 pts
Creates new processes	**0 pts** **Minimum Points** Maybe uses fork() but probably not correctly, doesn't use exec correctly	**15 pts** **Full Points** Correctly uses fork() and the appropriate variant of exec to create child process(es).	15 pts
Creates IPC	**0 pts** **Minimum Points** May call one of the pipe() system calls but does not do it correctly or able to send output of one process to input of another	**10 pts** **Full Points** Correctly uses one of the pipe() system calls to send output of one process to input of another	10 pts
			Total Points: 50

Submission

Submitted!
Jun 21 at 4:54pm
Submission Details
Download a2.zip
Download README-1.txt

Grade: 50 (50 pts possible)
Graded Anonymously: no
View Rubric Evaluation

Comments:
cssc4206 redid: 827228799
Dong Lin, Jun 21 at 4:54pm