CST 334: Operating Systems

Dr. Glenn Bruns

# MSH 2

**Instructions**. Extend your msh code of last week’s homework as follows:

* your code should continue to support the commands 'exit', 'help', and 'today' from MSH 1. Please modify the 'help' command to match the output shown in the example below.
* run Linux commands that are entered by the user. You must do this by using fork() and execvp() as shown in lecture and in OSTEP Figure 5.3. After a command is executed, the prompt should be displayed as usual.

It should be obvious, but your msh code must never (this week or any other) use the 'system' command, or any other command to execute shell commands. That would defeat the purpose of the assignment.

In OSTEP Figure 5.3, the code runs only command 'wc'. Your MSH code should run whatever command is entered on the command line.

Here is some sample output to help you understand what your program should do:

$ ./msh

msh> ls

temp.txt README.txt

msh> help

enter Linux commands, or ‘exit’ to exit

msh> foo 1

msh: foo: No such file or directory

msh> today

01/02/2018

msh> exit

$

Note the error message reported when ‘foo’ was entered. execvp() will return an error message if it couldn’t run the command entered by the user. How do you know this? execvp() will only return if there was a problem running the command that was passed to it. Here is the code I use to print an error message if execvp returns:

printf("msh: %s: %s\n", toks[0], strerror(errno));

Here toks is an array I used to store the “tokens” entered at the command line. strerror() and errno are defined by Linux. For more information, look for errno in the man page for exec, and look at the man page for strerror.

Your code needs to handle options provided on the command line. For example, you should be able to support the command ls -l -t

I have provided file msh2.c as your starter code. This code is a solution to MSH 1. You can find msh2.c in /home/CLASSES/brunsglenn/cst334/hw/hw3 on mlc104.

**Testing your code**. On mlc104, the directory /home/CLASSES/brunsglenn/cst334/hw/hw3 contains six test files test1.sh, test2.sh, …, test5.sh. Copy these to the directory where you are working on msh2.c. Each test should give exit status 0, like this:

$ ./test1.sh

$ echo $?

0

You need to run test1.sh first, as it will compile your code and produce binary file 'msh' that is used by the other tests. The directory also contains a Makefile. If you enter the command 'make', the target 'tests' in Makefile will run, causing each test to run. If you enter the command 'make clean', temporary files created by testing will be deleted.

**Submitting**: Submit your msh2.c file on iLearn. Here is information on [copying files to/from mlc104](https://docs.google.com/document/d/1TY6waCCHFkSBYCWKsHUelg9KH2yjXS2VOAyZLIPu93U/edit?usp=sharing).

**Grading**: This assignment is worth 60 points: 10 points for each of 5 unit tests, and 10 points for clean code (formatting, commenting, etc.).

Please use our Slack #assignments channel for questions to clarify what is being asked.