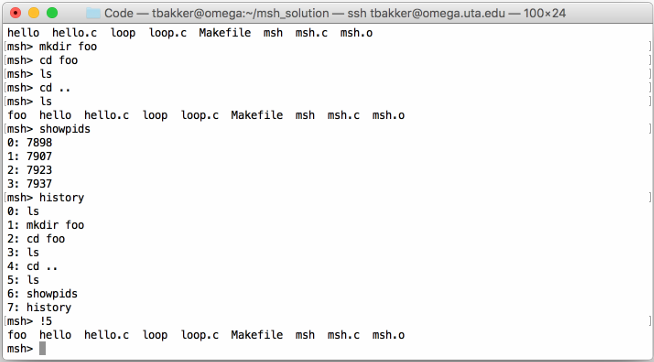
Requirement 10: Your shell shall support the pidhistory command to list the PIDs of the last 15 processes spawned by your shell. If there have been less than 15 processes spawned, then it shall print only those process PIDs

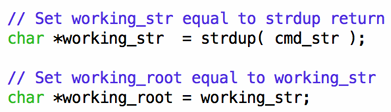
Requirement 11: Your shell shall support the history command which will list the last 15 commands entered by the user. Typing !n, where n is a number between 0 and 14 will result in your shell re-running the nth command. If the nth command does not exist then your shell will state “Command not in history.” The output shall be a list of numbers 0 through n-1 and their commands, each on a separate line, single spaced.



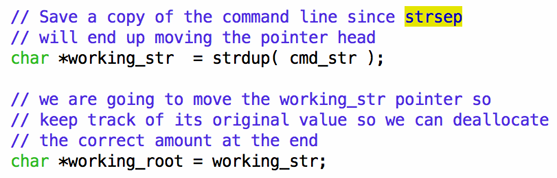
If there are less than 15 commands in the history only list the commands the user has entered up to that point.

Requirement 16: All code must be well commented. This means descriptive comments that tell the intent of the code, not just what the code is executing.

The following are poor comments:



The following explains the intent:



When in doubt over comment your code.

Requirement 19: Each function should have a header that describes its name, any parameters expected, any return values, as well as a description of what the function does.

Requirement 20: Remove all extraneous debug output before submission. The only output shall be the output of the commands entered or the shell prompt.

Administrative

This assignment must be coded in C. Any other language will result in 0 points. Your programs will be compiled and graded on omega.uta.edu. Code that does not compile on the omega.uta.edu with: gcc msh.c -o msh —std=c99 will result in a 0.

Your program, msh.c is to be turned in via Canvas. Submission time is determined by the Canvas system time. You may submit your programs as often as you wish. Only your last submission will be graded.

Hints

Read the man pages for the following: fork, exec, exit, print, fgets, strtok, strsep, strcmp, wait, and pause.

Use fork and one of the exec family as discussed in class to execute the command and call wait to wait for the child to complete. If the command is “cd” then use chdir() instead of exec. Note, chdir() must be called from the parent.

If you see garbage in any of your commands or parameters, try using the functions memset() or bzero() to clear out your input string and token array before and/or after you are done using them. Also, verify you are NULL terminating your strings. There are examples on the course GitHub repository for this assignment that show how to use execl and execvp.