CSC 415 Operating Systems Principles
Fall 2018 Anthony J Souza

***Homework #3 – Building a Simple Shell (Based on Nutt, Kernel Projects for Linux, Addison-Wesley, 2001, Lab #2), or Simple Programming with Files, Processes and Pipes.***

For this assignment you will implement your own Shell or Command Line Interpreter (e.g. to replace /bin/bash for simple interactions with the Linux Kernel.). Your shell will be character-oriented, and will fork off processes to execute user commands. Your shell should read lines of user input into a 256-byte buffer, then parse and execute the commands (be sure to clear the buffer between successive commands!) It should be possible for the user to specify the command to execute by giving an absolute path to the file containing the executable (e.g. ./hw1); or to use path expansion to locate the file containing the executable by using the environment PATH variable to construct a series of absolute paths and executing the first file found in this way (note that the execvp() command performs this processing automatically, you do not need to program this yourself!) Your code should parse the input string and separate it into a collection of sub-strings (stored in myargv[]) along with a count of the number of strings encountered (stored in myargc). Note that piped commands will require multiple argc/argv instances!

Your shell should support the following functions:

- Execute a single command with up to four command line arguments (including command line arguments with associated flags). For example:
  Myshell> ls –l
  Myshell> cat myfile
  Myshell> ls –al /usr/src/linux
- Execute a command in background. For example:
  Myshell> ls -l &
  Myshell> ls –al /usr/src/linux &
- Redirect the standard output of a command to a file. For example:
  Myshell> ls -l > outfile
  Myshell> ls -l >> outfile
  Myshell> ls –al /usr/src/linux > outfile2
  Myshell> ls –al /usr/src/linux >>outfile2
- Redirect the standard input of a command to come from a file. For example:
  Myshell> grep disk < outfile
  Myshell> grep linux < outfile2
- Execute multiple commands connected by a single shell pipe. For example:
  Myshell> ls –al /usr/src/linux | grep linux
- Execute the cd and pwd commands
  Myshell> cd some_path
  Myshell> pwd

**NOTE** that in most Linux distros CD is not a program like ls but rather is called a shell built-in. Built-ins are shell commands that are implemented in the shell and no some external binary. For giving your shell the cd and pwd commands, you need to implement these functions in the shell with your code, even if it is provided by your OS. This can be done with the chdir() and getcwd() functions in unistd.h

CSC 415 Operating Systems Principles
Fall 2018 Anthony J Souza

Suggested implementation strategy to implement a shell with multiple command line arguments (using iterative refinement):

1. Implement shell0 to initialize variables and then go into an infinite loop until stdin detects EOF (i.e. the user enters CTL-D). Each iteration through the loop, the shell program should prompt the user for input, read the input & echo it back to the user and then print out another prompt.
2. Implement shell1 by extending shell0 to parse the user's line of input into a series of strings that are stored into a ***char myargv*** ** array, along with a count of the number of strings in an ***int myargc*** variable. Print out the contents of myargv and myargc after you have parsed the comment. Allow the user to enter an 'exit' command to terminate the shell (instead of typing CTL-D and sending a kill signal to terminate execution).
3. Implement shell2 by extending shell1 to create a child process to execute the command (passing in myargv/myargc as arguments) and then wait for the child process to complete execution.

See the lecture slides and/or the course web site for a summary of what to submit with your homework assignment. *This assignment will be given a maximum of 150 homework points.*

**To get points for EC, your shell needs to work first. Broken shells are not eligible for EC.**

**Extra Credit (15 Points):** Implement your shell so that any combination of the shell functions above can be used in a single command line.

**Extra Credit (10-Points):** Implement the shell so the current working directory is shown on the prompt. For example:

OLD PROMPT:
myshell >>
NEW PROMPT:
myshell ~/ >>
myshell ~/hw3 >>
myshell ~/hw3/build >>
myshell /etc >>
myshell /etc/apach2 >>

Note: ~/ is only shown when in the CURRENT user's home directory.

**Extra Credit(20):** Add the functionality to your shell to store a history of all commands executed. This includes the ability to scroll through this history as well and rerun previously entered commands. See your basic Linux shell for examples. This also requires the use of the up and down arrow keys to scroll through the list.