Assignment – 1

Building a Shell

We will first build a simple shell, much like the bash shell of Linux. A shell takes in user input, forks one or more child processes using the fork system call, calls exec from these children to execute user commands, and reaps the dead children using the wait system call. Learn about the fork system call and all variants of the wait and exec system calls before you begin this lab.

Q1. Build a simple linux-like shell using C. On executing your code, the shell should start as:

MTL458:~$

where "~" is the path where we are currently at. The shell should terminate on pressing Ctrl+C. An incorrect number of arguments or incorrect command format should print an error in the shell. The shell must not crash and should simply move on to the next command. The shell should execute simple commands (commands which are readily available as executables and can be invoked using exec system calls) like ls, cat, echo, sleep. For now, you may assume that the user input does not include any extra functionalities like background execution, pipes, or I/O redirection.

(5 marks)

Q2. Implement the "cd" command. Treat the directory from where the shell script is run as home. You should be able to go both forward and backward in directories including multiple steps. E.g. If we go to "dir1" from the home directory we should see the following.
MTL458:~/dir1$

(3 marks)

Q3. Implement the "history" command. It should display the last 5 commands executed on the shell. If there's only one command executed since the start of the shell, it should show only one command in history and so on.

(2 marks)

Submission Instructions: Submit a zip file with folder name <Entry_No>_A1 (eg: 2016MT60648_A1). The folder should contain your C code file named shell.c