CS 620 HW 2 9/13/2013 Due date: 9/30/2013

Instructions:

1. Deadline: 11:00pm on Monday, September 30th.

2. Late policy: 1 day late: 5 points deducted; 2 days late: 10 points deducted; 3 days late: 20 points deducted; greater than 3 days: will not be graded.

3. Create a make file to build your executable file. The name of your executable must be **myshell**. Submit all your **source** files (no object or executable files), **readme** file and **makefile**.

   The *readme* file should contain any information that would help with the grading of your assignment.

4. The makefile (all lowercase) must generate the binary file myshell. A sample makefile would be:

```
# CS620
myshell:   myshell.c myshell.h
           gcc -Wall myshell.c -o myshell
myhistory: myhistory.c myhistory.h
           gcc -Wall myhistory.c -o myhistory
```

   The program *myshell* is then generated by typing *make* at the command line prompt. The *myhistory* is explained in Section 2.

   Note: the third and fifth line in the above makefile must begin with a tab.

5. Electronic submission: You assignment is due by 11:00 pm, 9/30/2013. Type in the following command from your cisunix account.

   $\sim cs620/submit\ 2\ file1\ file2$ ....

   2 is the assignment number. Note that if you want to resubmit, then you need to use 2a, 2b, 2c,..... for assignment number (not 2).

6. You can lookup UNIX programming textbooks or the internet for information on fork(), execvp(), and wait() system calls. Use the perror() system call to print error messages when using system calls. For example:

```
if (fork() < 0) {
     perror("Failed to execute fork command");
     exit(1);
 }
```

7. The shell code is available on-line and in various Unix programming texts. You can lookup the code of shell programs in textbooks and on-line. However, you must then write the shell script yourself. You are not permitted to copy code.

8. The assignment consists of 2 sections. The first section is worth 70 points and the second section is worth 30 points.

1 Basic Shell

Maximum grade: 70 points
Submit as myshell.c, myshell.h

The objective is to learn how the system calls *fork(), execvp(), wait()* are used by shell to execute programs. The shell is the fundamental user interface to an operating system. You have to write a basic shell that only has to execute external commands like

myshell>ls -l

Note that the command line prompt for myshell is *myshell>*. Your shell need not handle internal commands, pipes, redirection, background execution, or wild card * expansion. Therefore, treat the following as non-recognized commands:

myshell>ls -l | more
myshell> Command not found: ls -l | more
myshell> cat file > junk junk3
myshell> Command not found: cat file > junk junk3
myshell> ddddddddddd
myshell> Command not found: ddddddddddd

Your shell is not expected to execute internal commands such as cd.

myshell> cd
myshell> Command not found: cd

The sample output for *myshell* gives a list of commands that *myshell* should handle. Exit from *myshell* with the **quit** command.

2 History feature

Maximum grade: 30 points
Submit as myhistory.c, myhistory.h

Once you get the basic shell to work, copy the myshell.c program to myhistory.c. Modify myhistory.c to include the history feature, which allows users to see the most recently entered commands. In order to see what history does, type history at your command prompt.

The history command is not part of standard Linux commands, so you cannot just exec the history command. You have to write code for history. Implement history as a global two dimensional array: char history[100][80]. The sample file shows the history command output.