Homework 3: Simple Shell Program[1]

Due Date: February 20, 2018

A simple shell is a basic shell program that supports commands with I/O re-direction. Once, a user enters a command string (ended with a return key), your program should parse the command and need to handle I/O redirection signified by > for output to a file or < for input from a file. (no and pipe, you will do these in the OS class). The program has also 3 built-in commands that `cd`, `pwd`, and `exit`. The built-in functions are not executed by forking and executing an executable. Instead, the shell process executes them itself. All other command must be executed in a child process.

To implement the cd command, your shell should get the value of its current working directory (cwd) by calling getcwd() on start-up. When the user enters the cd command, you must change the current working directory by calling chdir().

What to Hand in

Submit a tar file using the following command

```
%tar cvf p1.tar README typescript your_codes #*.c *.C *.h Makefile
```

To extract you can use "tar xvf p1.tar"

1. A README file with:
    1. Your name and your partner's name
    2. If you have not fully implemented all shell functionality then list the parts that work (and how to test them if it is not obvious) so that you can be sure to receive credit for the parts you do have working.
2. All the **source** files needed to compile, run and test your code (Makefile, .c or c++ files, optional test scripts). Do not submit object or executable files.
3. Output from your testing of your shell program. Make sure to demonstrate:
    1. simple Unix commands
    2. built-in commands
    3. I/O redirection
    4. error conditions

    You can capture the screen output of shell program using `script`

```
> script
Script started, file is typescript
 % ./myshell
  myshell % ls
  foo.txt        myshell
  myshell % exit
  good bye
 % exit
exit
```

[1] This project is originally authored by Prof Tia Newhall and excerpted from http://www.cs.swarthmore.edu/~newhall/cs45/f01/cs45.html . I partially modified it for our class.

```
Script done, file is typescript
```

Useful Unix System Calls

getenv: get the value of an environment variable

```
path = getenv("PATH");
cwd  = getenv("PWD");
```

chdir: change the current working directory (use this to implement cd)

fork-join: create a new child process and wait for it to exit:

```
if (fork() == 0) { // the child process
} else { // the parent process
    pid = wait(&status);
}
```

execv: overlay a new process image on calling process

```
execvp( full_path_name, command_argv_list, 0);
```

access: check to see if a file is accessible in some way.

```
access(full_path_name_of_file, X_OK | F_OK);
```

open, close, dup: for I/O Redirection

```
// to re-direct stdout to file foo
int fid = open(foo, O_WRONLY|O_CREAT);
close(1);
close(fid);
```

Test:

Your program should be able to handle the following examples of commands. However, your program will be tested with more than these commands (including error handling cases).

```
myshell % ls  > ls.out    # redirect ls's stdout to file ls.out
myshell % cat foo.txt
myshell % wc  <  ls.out
myshell % cd /usr/bin
myshell % ls
myshell % cd ../
myshell % pwd
myshell % /usr/bin/ps
myshell % cd
myshell % find . -name foo1.txt
myshell % wc foo1.txt
myshell % exit
```