PURPOSE: The purpose of this lab is to allow students to learn a user interface aspect of a UNIX shell. Student will work with process management and some basic system calls.

Important note: please use sp1, sp2, sp3, or atoz servers for this lab.

UNIX Shell

We will start with 3 built-in commands:

- **cd** (change directory),
- **pwd** (print working directory), and
- **exit** (exit shell).

The built-in functions are **not** executed by forking and executing an executable. Instead, the shell process executes them itself.

Your shell is basically an interactive loop:

- it repeatedly prints a prompt "**csc60msh >** ",
- parses the input, executes the command specified on that line of input,
- and waits for the command to finish.

FILES TO COPY:

To get the files you need, first move to your class folder by typing: **cd csc60**
The following command will create a directory named **lab6** and put all the needed files into it below your csc60 directory.
Type: **cp -R /gaia/home/faculty/bielr/classfiles_csc60/lab6 .**
Spaces needed: (1) After the **cp** ↑ Don't miss the space & dot.
(2) After the **-R**
(3) After the directory name at the end & before the dot.

Please follow these steps:

- Review the source codes, compile, and execute the programs. Examine the output texts to understand the behavior of each program.
- I have provided you with a simple shell template (**lab6.c**) that you can work from. You build your program from it. Read the template closely to identify the key components and understand its execution flow. You can compile the shell using the following command: **gcc *lab6.c***
- **Function main.** Handling **built-in Commands**: There are three special cases where your shell should execute a command directly itself instead of running a separate process.
  - *First*, if the user enters **"exit"** as a command, the shell should terminate.
  - *Second*, if the user enters **"cd dir"**, you should change the current directory to "dir" by using the *chdir* system call. If the user simply types **"cd"** (no dir specified), change to the user's home directory. The $HOME environment stores the desired path; use *getenv("HOME")* to obtain this.
  - *Third*, if the user enters **"pwd"**, print the current working directory. This can be obtained with *getcwd()* function.
  - Additionally, we have to deal with the fact that a user might just type an Enter Key with no command.

Pseudo Code

```
/*-------------------------------------------------------*/
int main (int argc, char **argv)
{
    while (1)
   {
          int childPid;
          char *cmdLine;
          print the prompt();      /* i.e. csc60mshell > , Use printf*/

          cmdLine= readCommandLine(); /* use fgets not gets, provided */

          argc = parseCommand(cmdLine); /* use provided function parseline */

          /* code to print out the argc and the agrv list to make sure it all came in*/
          Print a line. Ex: "Argc = %i"
          loop starting at zero, thru less than agrc, increment by one.
              print each argv[loop counter]

          /* Start processing the built-in commands */
          if ( argc compare equal to zero)
               /* a command was not entered */
               continue to end of while-loop

          // next deal with the built-in commands
          //   Use strcmp to do the test
          //   after each command, do a continue to end of while-loop

          if ("exit")
              issue an exit call
          else if ("pwd")
              declare a char variable to hold the path of MAX_PATH_LENGTH
              do a getcwd
              print the path
          else if ("cd")
             declare a char variable dir
              if the argc is 1
                    use the getenv call  with "HOME" and return the value from the call to dir
              else
                    dir gets assigned the value of argv[1]
              do a call to chdir(dir) with error checking. Message = "error changing directory"

   }       /* end of the while loop
}          /* end of main
```

Resources

Useful Unix System Calls:

getenv/setenv: get/setenv the value of an environment variable
path = getenv("PATH");
cwd = getenv("PWD");
setenv("PWD", tempbuf, 1);

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

C Library functions:

```
#include <string.h>
String compare:
    int strcmp(const char *s1, const char *s2);
    strcmp(argv[0],"cd")
    strcmp(argv[0],"exit")
    strcmp(argv[0],"pwd")
    strcmp(….,">")
    strcmp(….,"<")

print a system error message:
    perror("Shell Program error");

Input of characters and strings:
    fgets(cmdline, MAXLINE, stdin);
```

Compilation & Building your program

The use of *gcc* is just fine. If you want to have the output go elsewhere from a.out, type:

gcc –o name-of-executable name-of-source-code

Partnership

Students may form a group of 2 students (maximum) to work on this lab. As usual, please always contact your instructor for questions or clarification.

Hints

Writing your shell in a simple manner is a matter of finding the relevant library routines and calling them properly. Please see the resources section above.

Keep versions of your code. This is in case you need to go back to your older version due to an unforeseen bug/issue.

I hope you enjoy this project….or at least learn from it ….and don't lose too much of your sanity.

Marks Distribution

Lab 6 with the built-in commands is work 25 points.

Deliverables

Your source file(s):

1. lab6.c
2. YourNameLab6.txt
    - Your program's output test (with various test cases). Please use the UNIX **script** command to capture your program's output. (Review Lab 2 for directions on using **script**.)

- Submit your two files to **SacCT**

All files should include the names of students involved in this assignment

Preparing your script file:

Run the program, and enter in sequence:

the Enter Key,
a Space, and then the Enter Key,
pwd,
cd ..
cd
exit

If all worked, submit your script file. (The script file will NOT contain the contents of lab6.c.)