INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COMPUTING LAB I, AUTUMN 2016

Assignment 2: DEVELOPING A CUSTOM LINUX SHELL

July 27, 2016

1 INTRODUCTION

The `command shell` is the interface between the user and the OS in Linux. Just like how a user interacts with the computer system through the graphical user interface, the shell is yet another such interface, but is one of the most preferred interfaces among the developers for reasons obvious to the developer community. As computer engineers, it is imperative for us to understand how the shell works. This assignment will give you an insight into the internals of Linux `shell commands` and how the shell executes them in turn making you further inquisitive of the internals of an OS.

2 PROBLEM STATEMENT

To understand the working of a `command shell` involving aspects such as *creating a process, loading a program* and also get an insight into the */proc directory*.
Write a C program which will act like a *mini command shell*. It will have a prompt and will accept the following commands and perform the required actions. The commands listed below should be coded as individually executable programs themselves, and should be able to accept command line arguments.
The shell will load these commands through the combination of `fork+exec` functions and execute them:-

1. **mypwd** - Print the present working directory to *STDOUT*.

2. **mymkdir** - Creates a directory.
   Sample Inputs:
   Single directory : `mymkdir` dir1
   Multiple directories : `mymkdir` dir1 dir2 dir2/dir3
   Absolute Path : `mymkdir` /rootpath/subdir/subdir2

3. **mycd** - Change Current Working Directory *(CWD)* to specified directory.

4. **myrm** - Remove a directory or a file. If a directory is to be removed, the files inside the directory should be deleted prior to deletion of the directory. *Sample Inputs:*
   Remove file(s) : `myrm` file1 file2 file3
   Remove empty directory : `myrm` -d dir1
   Remove a directory and its contents : `myrm` -r dir2

5. **mymv** - Move a *file/Directory* from one location to another. Delete the original directory after moving the contents. *Sample Inputs:*
   Move file(s) : `mymv` sourceFile targetFile
   Move Directory(s) : `mymv` sourceDirectory targetDirectory

6. **myps** - List all processes for the *current user* without regards to the present terminal.
   Options : -a *List all processes running on the system.*

7. **myls** - List the Directory contents. Output should be same as `ls -l` on a linux shell.

8. **mytail** - Print the last 'n' lines of input file.
   Options :

   - **<N>** *Number of lines from end to print.*
   - **<filename>** *Filename of the input file.*

   Sample Input : `mytail -10 file1.txt`

2.1 Details

The above given commands need to be programmed with standard C functions for Unix. Usage of `exec family/system()` along with bash commands will not be awarded any marks. All the commands should be able to handle absolute and relative paths as well as the combination of both wherever applicable as in a real shell interface.

2.2 Outputs

Write your own shell to execute the above commands. The shell should work as follows:-

1. Show a prompt and wait for user input.
2. On receiving the input, the shell should fork a child process
   Sample Input : `[/home/user]myshell$ mymkdir /home/user/newDir`
3. In the child process, call exec family functions to load your program corresponding to the input command.

2.3 Code Format

```
void myshell()
{
            ...
            ...

    while(1)
    {
        printf(PROMPT);
        cmd_len = getline(&buf,&buf_len,stdin);

            ...
            ...
        if(strcmp(cmd,"myexit")==0)
                exit(1);

        If( (pid = fork ()) == 0)
        {
            if (strcmp(cmd,"mypwd")==0)
            {
                execlp (“mypwd”,0);
            }

            else if(strcmp(cmd,"myls")==0)
            {
                execlp (“myls”,0);
            }
            ...
            ...

        }
        else
            waitpid(pid, &status, 0);
      ...
      ...
}
```

3 Deliverables

There should be separate C files for each implemmention of commands along with the main implementation file.

You need to compress above files as *assign*1_<*Roll_No*>.*tar.gz* and submit this single compressed file in moodle by the deadline.
Deadline: 4 August 2016, 2:00 pm IST Enjoy Coding !!