ASSIGNMENT 2 : DEVELOPING A CUSTOM LINUX SHELL

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.

Objective

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.

Description

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:-

(8 * 10 = 80 marks)

(a)   mypwd - Print the present working directory to `STDOUT`.

(b)   mymkdir Creates a directory

   Sample Inputs:
   Single directory `mymkdir dir1`
   Multiple directories `mymkdir dir1 dir2 dir2/dir3`
   Absolute Path `mymkdir  /rootpath/subdir/subdir2`

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

(d)   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`

(e)   mymv - Move a file/Directory from one location to another. Delete the original directory after moving the contents.
Sample Inputs:

```
mymv sourceDirectory  targetDirectory
mymv sourceFile  targetFile
```

(f)   myps - List all processes for the current user without regards to the present terminal.

Options:
-a   List all processes running on the system.

(g)   myls - List the Directory contents. Output should be same as `ls -l` on a linux shell.

(h)   mytail - Print the last 'n' lines of the input file

Options:
<N>   Number of lines from the end to print
<filename>   Filename of the input file

Sample Input:
```
mytail -10 file1.txt
```

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.

Write your own shell to execute the above commands. The shell should work as follows:-   **(20 marks)**

1. Show a prompt and wait for user input.
2. On receiving the input, the shell should fork a child process
   a. 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.

The following is a skeleton code for your shell program:-

```c
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 f(strcmp(cmd,"myls")==0)
            {
            execlp ("myls",0);
            }
            ...
            ...

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

Deliverables

The C programs (ideally there should be 9 source files) along with the Makefile compressed in a .tar file format <rollno>_a2.tar.gz. It is mandatory that your code should follow proper indentation and commenting style. There will be deductions in the awarded marks, if you fail to do so.