Computing Lab – I

Assignment 7

12 October, 2017

Objective of this Assignment

- To develop a custom Linux shell using C
- Command shell: an interface between the user and the OS in Linux
- Motivation: gaining some understanding of how a shell works
- This is an individual assignment.

Problem statement

- Write a C program which will act like a mini command shell
  - Display a prompt
  - Accept a subset of the Linux commands and perform the required actions.
  - Each command should be coded as individually executable programs themselves, and should be able to accept command line arguments.
  - The shell will load and execute these commands through fork() and exec() system calls

Commands to be implemented

- mypwd: print the present working directory to STDOUT
- mymkdir: create a directory
  - Single directory: mymkdir dir1
  - Multiple directories: mymkdir dir1 dir2 dir3
  - With absolute path: mymkdir /home/x/testdir
- mycd: change current working directory to specified directory

Commands to be implemented

- myrm: remove a file or directory
  - Remove file: myrm file1
  - Remove directory: myrm dir1: should remove specified directory only if it is empty
  - Remove directory recursively: myrm –r dir1: should remove specified directory and all its contents
  - Removal of multiple files and directories allowed

Commands to be implemented

- mymv: move a file or directory from one location to another
  - Move file: mymv sourceFile targetFile
  - Move files: mymv sourceFile1 sourceFile2 targetDirectory
  - Move directories: mymv sourceDir1 sourceDir2

Commands to be implemented

- myls: list the contents of specified directory
  - If no directory specified, assume current working directory
  - Output should be same as that of “ls – l” on a standard Linux shell
- mycat: show contents of the specified file
  - mycat file1 displays content of file1 on STDOUT

Commands to be implemented

- mytail –n: show last n lines of the specified file on STDOUT
  - mytail -10 file1 shows the last 10 lines of file file1

- myps: list all processes for the current user

- myexit: log out (stop program execution)

Basic framework of shell

- The custom shell should
  - Display prompt and wait for user input
  - Upon receiving input command, fork()
  - Child process should use exec() to load and execute program corresponding to input command

Basic framework of custom shell

```
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);
      ...
      ...
}
```

What you should do

- All commands should be able to
  - Handle one or more command line arguments
  - Handle relative and absolute pathnames
  - Display proper error messages in case of wrong input, and show prompt again
- As done in standard Linux shell

What you should not do

- Using exec() family of system calls with bash commands will not be awarded any marks
- Shell should not terminate abruptly in case of wrong inputs (e.g., a file which does not exist)

Submission Instructions

- There should be separate C files for implementation of each command, along with the main file implementing the custom shell.

- Compress above files as assign7_<roll_no>.tar.gz and submit this single compressed file in moodle.

- **Submission Deadline – October 26, 2:00 PM IST**

Marking scheme

- Basic framework: 20%
- Commands: 30%
- Support for multiple command-line arguments, absolute and relative paths: 20%
- Support for handling errors in input and giving proper error messages: 20%
- Documentation, understandability: 10%

{>;}

HAPPY
CODING