Lab 2 : System Programming I - System Calls for File Management

August 7, 2016

Objective :

- Lab 2 is intended to provide the way to use the most common system calls in order to make input-output operations on files, as well as operations to handle files and directories in Linux

Recommended Systems/Software Requirements:

- Any flavour of Linux

References:

1. *Unix concepts and applications*, Fourth Edition, Sumitabha Das, TMH.

Theoretical Background:

You are expected to refer to the resource reading file available at the course website before starting the lab.

Problems:

1. Implement in C the following UNIX commands using System calls : *cat*, *ls* and *mv*
2. Determine the size of a file using the *lseek* command. Once you found out the size, calculate the number of blocks assigned for the file. Compare these results with the similar results obtained when using the function *stat*.
3. Write a C program that finds a file in a file-tree starting from a given directory. The name of the file for which we are searching for, as well as the name of the starting directory should be read from the command line. Optionally, the name of the file can be specified as a pattern using the '*' character.
4. Write a C program that deletes a directory with all its subfolders. The name of the directory should be read from the command line.

5. In this assignment we will start writing a command interpreter (Shell). The shell will give a prompt for the user to type in a command (from a set of commands), take the command, execute it, and then give the prompt back for the next command (i.e., actually give the functionality of a shell). Your program should do the following:

   - Give a prompt "myshell$" for the user to type in a command
   - Implement the following builtin commands:
     (a) cd $< dir >$: changes the directory to "dir"
     (b) pwd : prints the current directory
     (c) mkdir $< dir >$: creates a directory called "dir"
     (d) rmdir $< dir >$: removes the directory called "dir"
     (e) ls : lists the files in the current directory. It should support both ls without any option and with the option "-l"
     (f) exit : exits the shell

     The commands are the same as the corresponding Linux commands by the same name. Do "man" to see the descriptions. You can use the standard system calls chdir, getcwd, mkdir, rmdir, readdir etc. to implement the calls (standard C library functions are available for these; look them up). These commands are called builtin commands since your shell program will have a function corresponding to each of these commands to execute