BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI (RAJ.)

Second Semester 2017-18 CS F111 Computer Programming

LABORATORY SESSION #14

(Topics: Regular expression; Unix filters; File handling in C)

In this lab, you will learn about some other Unix commands, and to work with regular expressions. You will explore the working of commands by consulting the online manual, after getting an idea from the description provided. Thereafter you will attempt the exercises given.

The ***grep*** command searches a named input file (or standard input, if no input is mentioned) for a particular pattern of characters and displays all lines that contain that pattern. By default, it prints the matching lines. Several options can be given for this command, e.g., `grep –i "hello" f1.txt` prints out all lines of f1.txt containing the pattern "hello", regardless of whether uppercase or lowercase letters. Explore the man pages to find out what options can be used.

The ***head*** command, by default, prints the top ten lines of the specified file. You can also modify how many lines you want to print by mentioning an argument, e.g., `head -7 f1.txt` prints to standard output the first seven lines of the file.

The ***tail*** is the complementary of the head command. It prints the last ten lines of the specified file.

Regular expressions and pattern matching using `grep`:

Regular expression helps us search and manipulate the text in a flexible and concise way, based on patterns. The following table gives the meaning of special symbols when used with **grep** for pattern-matching:

Symbols	**Significance**
*	Matches zero or more occurrences of previous character
.	Matches a single character
[iik]	Matches a single character i,j or k.
[^ ijk]	Matches a single character which is not an i, j or k
[c1-c2]	Matches for a single character within this ASCII range represented by c1 and c2
^ pat	Matches pattern pat at beginning of line
pat$	Matches pattern pat at end of line

Practice exercises

1. Make two subdirectories **dir1** and **dir2** under the current directory, and copy the file `/etc/passwd` into **dir1** as **f1.txt** (without changing the directory). This file contains the list of all users along with their passwords (encrypted, of course!) and other details, one user per line. Then change your location to **dir1.**
2. Store the last ten lines of **f1.txt** to **f2.txt**. Then append the first 5 lines of **f1.txt** to **f2.txt**.

3. Remember piping wherein the output of one command can be “piped” as the output of a second? Piping is very useful to accomplish tasks. For instance, if you want to find out how many users who belong to the 2016 batch have accounts, you can write this command:

   ```
   grep "f2016" /etc/passwd | wc -l
   ```

   This tells the shell to pass the output of the grep output to wc –l command that counts the lines.

   Now, use a piping sequence to count how many users with the login ID beginning f2017 are currently logged into the system. Next, modify this sequence (using the –v argument of grep) to print all others (i.e., excluding those whose login IDs begin with f2017).

4. Store all *but* the first 4 lines (i.e., starting from the fifth line) of **f1.txt** to **f2.txt**. (Hint: Explore options of `tail` command referring to the online manual.)

5. Create a file **f3.txt** under the directory **dir2**, using `cat`, that contains the following text:

   Unix Linux
   Redhat Ubuntu
   Fedora Redhat Debian
   Ubuntu Linux

   Write a command sequence that prints those lines that contain the word *Ubuntu* but doesn’t contain the word *Redhat*.

6. Write in plain English what these regular expressions match. You can create files with appropriate content to check your answers.
   a. `a.*b`
   b. `..*`
   c. `^}$`

7. The file `/etc/passwd` contains, among other things, the login IDs of all users of the Linux system installed. Every line of the file contains data of one user each.

   a. Write a command sequence to view lines 51 to 100 of this file. Examine the 50 lines of the output of this command sequence. You will notice these are the login IDs of some of the first-degree users.

   b. Write the Unix command sequence to list only those lines containing details of first-degree student users whose login IDs have *at least* two occurrences of the digit 6 in them.

   c. How many first-degree student users have their ID numbers more than 500 but less than 900? Write down the Unix command sequence to do this.

   d. Write a Unix command sequence to list only those users who are not first-degree students and whose login IDs do not begin with the letter “p”. How many such users are there?

8. Write C programs to implement the barebones version of each of the following Unix commands. You will use file handling and command line arguments to do the task.
   a. `cat` b. `cp` c. `head` d. `grep` (without use of regular expressions)