BIRLA INSTTUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)
Second Semester 2019-20   CS F111 Computer Programming

LAB SESSION #2

(Unix Commands and vi Editor)

As discussed in Lab 1, to do any task in Unix, certain commands need to be executed. We have studied few Unix commands in the last Lab and today we will be discussing some other Unix commands. We will also discuss how to use vi editor for creating text files.

Grep Command:

The ***grep*** command searches a named input file (or the 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 used with grep. For instance, grep –i "hello" file1.txt prints out all lines of file1.txt containing the pattern "hello", regardless of whether uppercase or lowercase letters. Find out the meaning of the following three other options to grep using the manual page: –v, –c and –o. Try out using some examples, and then note down the meaning of these options.

Example: Consider that there is a file named demo.txt stored in your home directory

```
THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
this line is the 1st lower case line in this file.
This Line Has All Its First Character Of The Word With Upper Case.


Two lines above this line is empty.
And this is the last line.
```

Demo.txt

```
Command1:     grep "this" demo.txt
Output:              this line is the 1st lower case line in this file.
                     Two lines above this line is empty.
                     And this is the last line.
Observation:      Search is case-sensitive

Command 2:    grep -i "this" demo.txt
Output:              THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.
                     This line is the 1st lower case line in this file.
                     This Line Has All Its First Character Of The Word With Upper Case.
                     And this is the last line
Observation:  All lines containing the keyword "this" (not case-sensitive) displayed.
```

Try executing grep command with other options.

Pipe command:

The Pipe is a command in Linux that lets you use two or more commands such that output of one command serves as input to the next. In short, the output of each is processed directly as input to the next one like a pipeline. The symbol '|' denotes a pipe.

Example:

ls -l | grep ".txt" The output generated by the ls –l command is provided as input to the grep command and the grep command searches for the presence of the keyword ".txt" in the input (thus fetching all files which end with .txt)

<u>Input and Output Redirection</u>

Most Unix system commands take input from your keyboard and send the resulting output to your terminal. *A command normally reads its input from the standard input, which happens to be your keyboard by default. Similarly, a command normally writes its output to standard output, which is again your terminal by default.* However, these default streams can be changed through using input/output redirection so that the input is taken from a file, and the output goes to a file, for instance.

The output from a command normally intended for standard output can be easily diverted to a file instead. This capability is known as output redirection. If the notation **> f1** is appended to any command that normally writes its output to standard output, the output of that command will be written to file **f1** instead of your terminal. This is ***output redirection***. Just as the output of a command can be redirected to a file, so can the input of a command be redirected *from* a file. As the greater-than character **>** is used for output redirection, the less-than character **<** is used to redirect the input of a command. This is ***input redirection***.

Example:

wc –l test.c **>** text.txt → Writes the output of wc –l command in text.txt instead of terminal. The content of the text.txt gets overwritten. To avoid overwriting of text.txt **>>** is used instead of **>**.

Similarly you can observe the use of input redirection by redirecting a file to wc –l:
wc –l < test.c → The file is given as an input to wc –l, and the no. of lines in the file are printed on the screen.
Can you observe some difference in the output of the command above and "wc –l test.c"?

Attempt each of the following tasks on the Linux machine, and write down the corresponding Linux commands you used for each task in your notebook:

1. How do you make the system print out the name of <u>your home</u> directory?

You notice that a long string starting with the / character is printed, with words separated by the / character again. The starting / is called the root directory under which all other directories and sub-directories reside.

2. What is the <u>parent directory</u> of your home directory?

Just as the home directory is denoted by ~ symbol, the current directory is denoted by the symbol . (period), while the parent is denoted by .. (two periods).

3. Make a directory called **dir1** under your home directory.
   a) Change to this directory.
   b) Make three empty files: **file1 file2** and **file3** under the current directory.
      (You can either use the **touch** command, or the cat **>** *filename* operator. Try both options.)
   c) Make a sub-directory called **dir1-1** under the current directory.
4. Draw the file structure of the existent files and directories under your home directory.
5. Remove the subdirectory **dir1-1**.
6. Try now removing **dir1**
7. Store the list of all the files and directories (using ls long listing) in a file called **dirfile**.
8. Display the contents of **dirfile** on the screen using the **cat** command

9. Store in a file called **userlist** the list of users who are currently logged into the system

10. Combine the contents of **dirfile** and **userlist** and store in the file called **file1**

11. Print the number of lines, words and characters in **file1** using the **wc** command.

12. Copy the contents of **file1** into **file2**.

13. Append the following two lines into **file2**:

    This is file2.

    And I am using Linux!

Introduction to vi-editor

The VI editor is the most popular and classic text editor in the Linux family. To start the *vi* editor type the following command in the terminal: **vi filename.txt.** Note that the **filename.txt** can be any file provided by you.

Example: vi FILE.txt

If FILE.txt is an existing file, then the editor would open it for you to edit. Else, it creates FILE.txt

Three modes of operation in the vi-editor:
- Command mode
- Insert mode
- Escape mode

Command mode:

- The vi editor opens in this mode, and it only **understands commands**
- In this mode, you can, **move the cursor, cut, copy, paste the text, etc.**
- **Commands are case sensitive.** You should use the right letter case
- To enter into Command Mode from any other mode, press Esc key.

Insert mode:

- You can switch to the Insert mode from the command mode **by pressing 'i' on the keyboard.** (You may also press I, o, a, etc.)
- Once you are in Insert mode, any key would be taken as an input for the file on which you are currently working.
- To return to the command mode and execute any command, press the Esc key.

Escape mode:

- It is invoked by typing a colon [:] while vi is in Command Mode. The cursor will jump to the last line of the screen and vi will wait for a command. This mode is specifically used to perform tasks such as saving files, quitting vi and executing commands.

Vi Editing Commands:

Commands used in vi-editor for performing various tasks. You should be in the "command mode" to execute these commands. VI editor is case-sensitive so make sure you type the commands in the right letter-case. Following commands can be used in vi-editor:

- i - Insert at cursor (goes into insert mode)
- a - Write after cursor (goes into insert mode)
- A - Write at the end of line (goes into insert mode)
- ESC - Terminate insert mode
- u - Undo last change

- U - Undo all changes to the entire line
- o - Open a new line (goes into insert mode)
- dd - Delete current line (ndd deletes n lines)
- yy – copy current line
- p/P – paste the copied line one line below/above cursor
- D - Delete contents of line after the cursor
- C - Delete contents of a line after the cursor and insert new text. Press ESC key to end insertion.
- dw - Delete word
- 4dw - Delete 4 words
- cw - Change word
- x - Delete character at the cursor
- r - Replace character

Moving within a file

- k - Move cursor up
- j - Move cursor down
- h - Move cursor left
- l - Move cursor right

You need to be in the command mode to move within a file. The default keys for navigation are mentioned here; You can **also use the arrow keys on the keyboard**.

Saving and Closing the file

- Shift+zz - Save the file and quit
- :w - Save the file but keep it open
- :q/:q! - Quit without saving
- :wq - Save the file and quit
- :set nu – set line numbers

Note that the commands beginning with a colon (:) indicate the Escape mode. To save or quit, you should be in the Escape/Command mode.

<u>Beginning to use the *vi* editor</u>

14. Open **file3** using *vi* editor and add the following line of text to it: This is file3!

15. Save and exit editing **file3**. You will now be at the shell prompt.

16. Append the contents of **file2** (created previously by you) to **file3**.

17. Open **file3** using vi editor again, and insert the following line after the first line: (25 asterisks)

```
*************************
```

20. Now insert a line of 25 asterisks after the last line of **file3**, just by using few vi commands (that is, do not type out the line again).

21. Save the modified file into another file named **file4**, without exiting vi editor.

22. Now, count the number of characters in file4, without exiting vi editor. (You can escape to the shell by typing ! in the command line mode.)

23. Exit vi editor, so that none of these changes you made is recorded in file3.