Birla Institute of Technology & Science, Pilani
Computer Programming (CS F111)
Lab-2

Objectives:

1. File maintenance Command
2. Creating files using vi editor
3. File Editing in vi

INTRODUCTION

In Lab1 we have learned few basic commands like **cd** (change directory), **ls** (list), **mkdir** (make directory), and we have learned how to navigate through the directory structure. We have also seen what all files are present in the directory. In this lab session we will learn how to create files, display its contents, file maintenance commands, and file permissions. Apart from this we will learn some more UNIX commands. In UNIX there are many text editors like vim, nano, emacs, etc. We shall be using the VI editor for all the labs.

FILE MAINTENANCE COMMANDS

1. Copy a file using ***cp*** command –
   - **a.** The **cp** command copies the content of one file into another file.
   - **b.** Create a new file with a name **first.txt**

     If there is no file by the name **first.txt**, you can temporarily create one using
     ```
     [user@localhost: Lab2]$ echo "the first file" > first.txt
     ```
   - **c.** Do the following and copy the content of **first.txt** into **firstCopy.txt**
     ```
     [user@localhost: Lab2]$ cp first.txt firstCopy.txt
     ```
   - **d.** Now see the contents of the file **first.txt** and **firstCopy.txt** (Hint: use the **cat** command), you can observe that the contents of the two file are same.
   - **e.** You now have two copies of the file, each with identical contents. They are completely independent of each other and you can edit and modify either as needed.
   - **f.** To copy entire directory, you should use the -r option of cp command.
     - **i.** Come back to home directory.
     - **ii.** To copy **Lab2** directory to **Lab2copy** directory
       ```
       [user@localhost: ~]$ cp -rv Lab2 Lab2copy
       ```
       (The option v gives verbose output.)

2. Moving a file with the **mv** command **–**
   - **a.** The **mv** command is used to move files between the directories.
   - **b.** So you are in directory **Lab2**, create a new directory inside **Lab2** with a name **mvTest** (use **mkdir** command).
   - **c.** Let us move **firstCopy.txt** into **mvTest** directory
     - ➢
       ```
       [user@localhost: Lab2]$ mv firstCopy.txt mvTest
       ```
     (If the directory mvTest does not exist, then a **file** named mvTest will be created.)
   - **d.** Now go the directory **mvTest** (use **cd** command) and see that **firstCopy.txt** file is present in the directory.
   - **e.** Now come back into the directory **Lab2**, list the contents of the directory you will see that **firstCopy.txt** is not there.
   - **f.** Now try the following command
     - ➢
       ```
       [user@localhost: Lab2]$ cp first.txt mvTest
       ```

g. Now list the contents of both **Lab2** and **mvTest** directory using the **ls** command and you will be able to see that **first.txt** is present in both the directories.

h. What you can observe is that **mv** command moves the file into other directory, while **cp** command will create a copy of file and then put that copy into the other directory.

3. Rename a file with **mv** command –
   a. UNIX does not have a command specifically for renaming files. Instead, the **mv** command is used both to change the name of a file and to move a file into a different directory.
   b. Let us change the name of file **first.txt** to **myFirstFile.txt**, do the following:
      ➢
      ```
      [user@localhost: Lab2]$ mv first.txt myFirstFile.txt
      ```
      (Please give a space between mv, first.txt, and myFirstFile.txt)
   c. List the content of the directory **Lab2** as a result of executing the above command you will find that there is no longer a file called **first.txt,** but a new file called **myFirstFile.txt** contains what was previously in **first.txt**. You can see that by displaying the contents of the file using **cat** command.

4. Removing a File with **rm** command –
   a. Use the **rm** command to remove a file.
   b. Go to the directory mvTest and type in the following command:
      ➢
      ```
      [user@localhost:~/ Lab2]$ rm myFirstFile.txt
      ```
   c. The file will be deleted. If, rather you want a prompt for confirmation of whether you really want to remove the file, use **rm -i**
      ➢ rm: remove firstfile (y/n)? y
      ➢ Type **y** or **yes** to remove a file; type **n** or **no** to leave it.
   d. List the contents of the directory and you will not find **myFirstFile.txt** file.
   e. To remove all files with extension **.txt**:
      ```
      [user@localhost:~/ Lab2]$ rm *.txt
      ```
   f. A directory can be deleted using the command **rm –r**

5. Removing a directory with **rm / rmdir** command –
   a. Create a temperory directory called **temp**
   b. Goto **temp** directory
   c. Create two directories **temp1** and **temp2**
   d. To remove directory **temp2**
      ➢
      ```
      [user@localhost: temp]$ rmdir temp2
      ```
      ➢ Check the result by typing `ls` before and after the `rmdir` command.
   e. Come back to **temp** directory
   f. To remove directory **temp** and all its contents
      ➢
      ```
      [user@localhost: temp]$ rm -rvf temp
      ```
      ii. In the option set, r indicates recursive, v indicates verbose, and f indicates forced deletion.

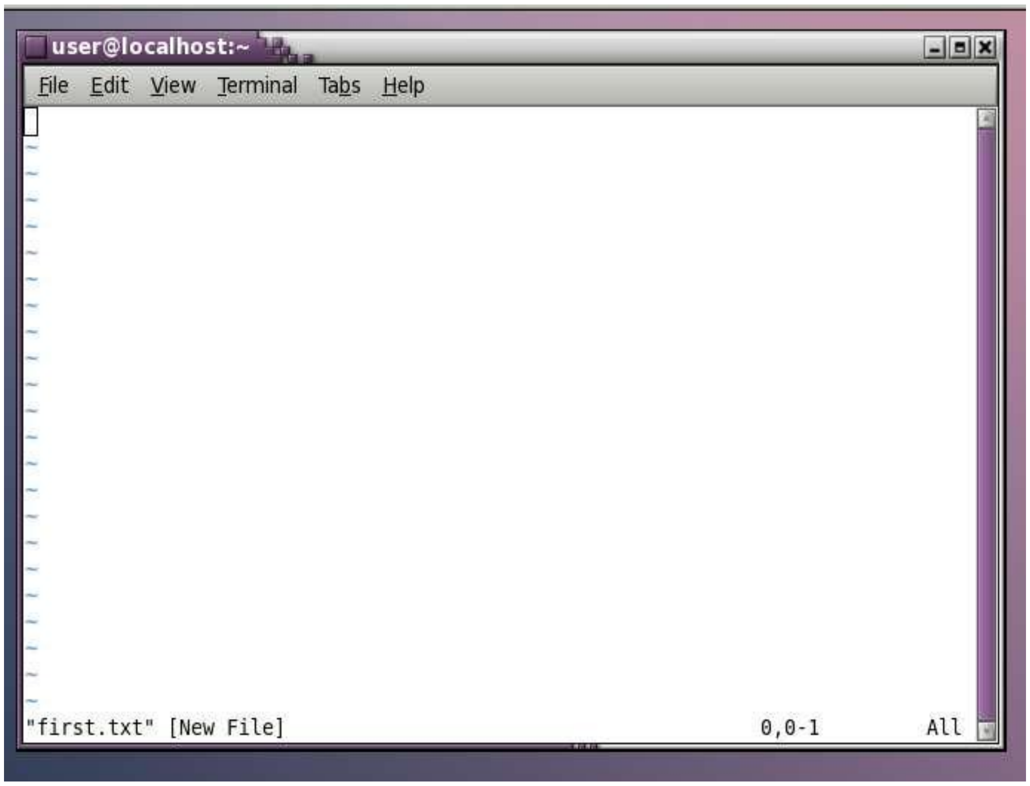
<u>CREATING AND WRITING INTO FILE USING VI EDITOR</u>

1. Create a directory with a name **Lab2** and go to that directory (Hint: use **mkdir** and **cd** command).
2. You should be in the following working directory
   ```
   [user@localhost: Lab2]$
   ```
3. You can verify that the current working directory is initially empty (Hint: use **ls** command)

4. Let us create a simple text file using **VI editor**. To create a file you should specify a file name that uniquely identifies that file in the directory. Two files cannot have same filename if they are in the same directory.
5. Let us assume **first.txt** to be the name of the file that you want to create.
6. To open the **VI editor** and create the file **first.txt** follow the following commands

```
[user@localhost: Lab2]$ vi first.txt
```

7. Refer to **Figure-1** that's what you will see when **VI editor** opens.
8. You are now ready to create/edit the file named **first.txt**.
9. Ok, now you just press the **ESC button** on your keyboard and then while holding the **SHIFT button** down press the colon key **(:)** release shift button and type **wq** on keyboard and press **ENTER** key. The **VI editor** will exit and you will again come to the command prompt.
10. Verify the contents of your directory again (use **ls** command)
11. You will see that a file with a name first.txt is created and is present in your working directory.
12. **Surprised!!!** What happened when you pressed **ESC** button and while holding **SHIFT** button typed colon key **(:)** and then **wq**? Doing this you instructed the **VI editor** to write and quit (**wq** stands for write and quit). Seeing this instruction the **VI editor** creates a file with the name that you specify and quits after saving that file in your directory.
13. Try to create one more file using vi editor let the name be **second.txt**, but this time while exiting the **VI editor** just press **ESC** and while holding the **SHIFT** button press the colon key **(:)** and then type **q**. Check the contents of your directory again, you will not find **second.txt**.
14. Can you tell why the **second.txt** file is not created??? Because you instructed the **VI editor** to quit (q stands for quit) without writing/saving the file **second.txt**.
15. **Congratulations!!!**, you know how to create a file using **VI editor**.



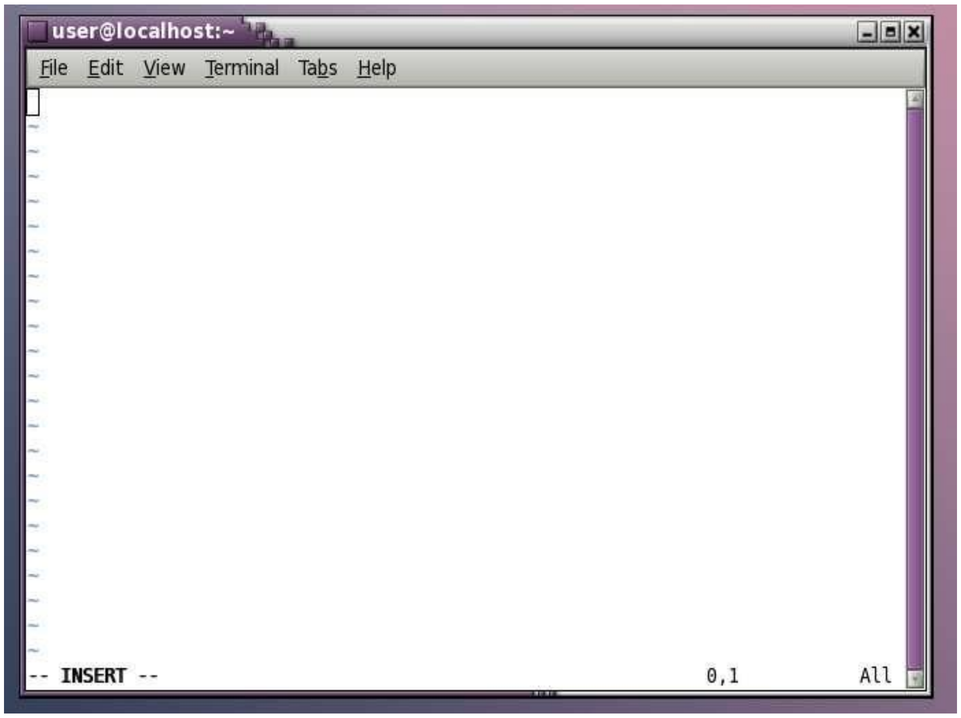
```
"first.txt" [New File]                                   0,0-1         All
```

Figure-1: VI editor opens given the file name as "first.txt"

16. Now Let us write something inside the file you have just created (**first.txt**).
17. Open the file using vi editor

```
[user@localhost: Lab2]$ vi first.txt
```

   a. Do you observe the cursor blinking on the upper left hand corner of the screen? The editor is waiting for the instruction from the user to do one of the many things that it can do (you will learn them gradually).
   b. Press **i** on your keyboard, and observe the bottom of the screen as shown in **Figure-2**. It reads **--INSERT --**
   c. Type the following:
      i. My journey begins from here.
      ii. It's going to be an enjoyment



```
-- INSERT --                                   0,1          All
```

Figure-2: Press "i" to insert text

18. Again press **ESC** and while holding the **SHIFT** button press the colon key **(:)** and then type **wq** to write the contents to the file **first.txt** and quit the **VI editor**.

DISPLAY THE CONTENTS OF A TEXT FILE

Now if you want to display the contents of the file use the **cat** command, see below:

```
[user@localhost: Lab2]$ cat first.txt
```

Basic Operating Modes in vi

We mentioned while in the process of creating file, that we are switching between two operating modes. The vi Editor at any instant can be found to be in one of the following 3 modes.

Command Mode
Command Line Mode
Insert mode

The navigation among the modes is best illustrated by the following figure.

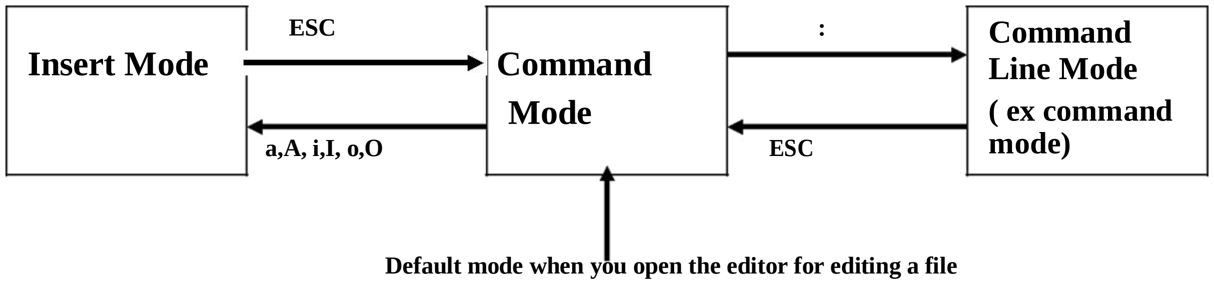

Insert Mode
ESC
a,A, i,I, o,O
Command Mode
:
ESC
Command Line Mode (ex command mode)
Default mode when you open the editor for editing a file

Fig. 3 Modes in vi

The technical details of each of the modes are as follows:

Command Mode:

Once you finished entering or changing the text, you can move to command mode by pressing ESC key. Unlike in Insertion mode, when a key is pressed in the command mode, it doesn't show up on the screen but it performs some function based on the key pressed.

Refer to the Table-1 for the basic cursor movement commands and Table-2 for commands for editing a text.

Table: 1 Basic Cursor Movement commands

h or left arrow key	Move cursor to the left one character.
l or right arrow key	Move cursor to the right one character.
j or down arrow key	Move cursor down one line.
k or up arrow key	Move cursor up one line.
^	Move cursor to the beginning of the line.
$	Move cursor to the end of the current line.
G	Move cursor to the last line of your file.
w	Move to first character of next word
b	Move to first character of previous word

Table: 2 Commands for Editing text

i	Inserts text to the left of the cursor.
I	Inserts text at the beginning of the line, no matter where the cursor is positioned on the current line.
a	Begins inserting after the character (append) on which the cursor is positioned.
A	Begins inserting at the end of the current line, no matter where the cursor is positioned on that line.
o	Begins inserting text on a new line below the current line, no matter where the cursor is positioned on that line.
O	Begins inserting text on a new line above the current line.
Ins key	Toggle between insert mode and replace mode. If not currently in Insert Mode, works as same as pressing **i**

Insert Mode:

vi editor opens in command mode. To perform the said activities on the file, we should change the current working mode to insert mode. Insert mode allows editing the contents of the current file, making a new file and other manipulations with the file content. This can be done by pressing the keys **i, a, o, O** at command mode (Table-2). Pressing **i** on the keyboard at command mode allows to insert character in the page on the left side of the cursor, press **a** to insert characters on the right side of the cursor.

To open a new line under the cursor, type **o** in command mode and pressing **O** (Uppercase letter) allows opening a new line above the cursor. Once we are changed to insert mode, any character of the keyboard can be typed and written on the screen.

Task: Practice to use **i, a, o, O** with the file you have created i.e. **will**. While working with it, try to keep track of the current mode you are in with the help of Fig. 2.

Command Line Mode:

The bottom line of the vi editor is called command line. This mode allows giving command at the command line. (This mode is also called as ex command mode, as the commands issued at the command line are compatible with ex editor.) All commands issued in command line mode are displayed in the command line.

Let us introduce a minimal set of commands issued at command line mode here.

Command	Purpose
:w	To save your file but not quit *vi*
:w <*filename*>	To save the contents in the screen (buffer) into the file named *filename*.
:q	To quit if you haven't made any edits
:q!	To quit with out saving the changes made in the current edit
:wq	To both quit and save your edits.

Exercise-1

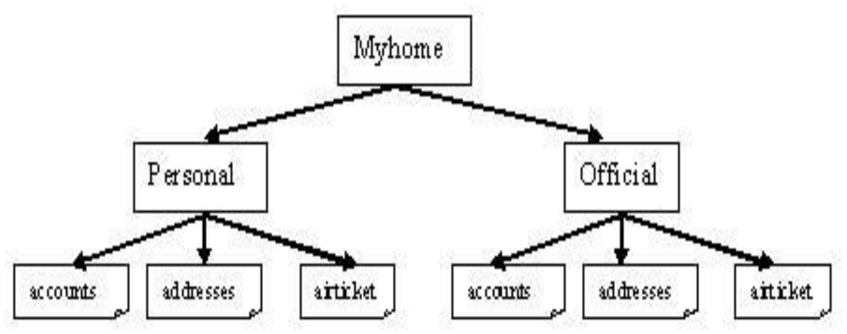


Fig. 4 Directory Structure

Refer the directory structure shown in Fig-4. Accomplish the following tasks, in sequence.

1. Create the directory structure as in Fig-4 in your home directory
2. Rename each file in **Personal** with the prefix P. That is, the file **accounts** in **Personal** directory should be renamed as **Paccounts**.
3. Allow ***others*** from modifying the file **Pairticket** in **Personal** and **Official** directories by setting appropriate file permissions.
4. The **address** file in the **Official** directory contains some confidential addresses. To protect it,
    a. Remove all permissions to this file by anyone other than the owner.
    b. Move the file named **address** to a newly created directory in **Myhome** (you may create your own directory under **Myhome**) from its current location.
5. You may also rename the above mentioned file as **medicalreport**,

6. Create a directory **Backup** under **Myhome** directory and move all the contents of **Personal** into **Backup**.
7. Remove all the contents of **Personal** directory. You should do it from the **Official** directory.
8. Now rename the **Backup** directory as **Personal**.

File editing

Exercise-2: Create a file using vi. Add few lines of text and tryout the following vi commands.

Copying and Pasting Text

The commands for copy/paste are issued in command mode. Make sure that you are in command line mode. If not, first switch to command mode.

Command	**Explanation**
yy	Copy (yank) the current line
6yy	Copy Six lines, beginning with the current line
p	Put the copied text in next line
P	Put the copied text above the current line

Once you issue the copy command (yy, 2yy, 4yy etc), the copied text goes into a temporary memory area (buffer) that is replaced each time when you copy (or delete) more text. Only the current contents of the temporary buffer can be put back into your file. As a result, when you use copy (yy), use the put (p) command immediately.

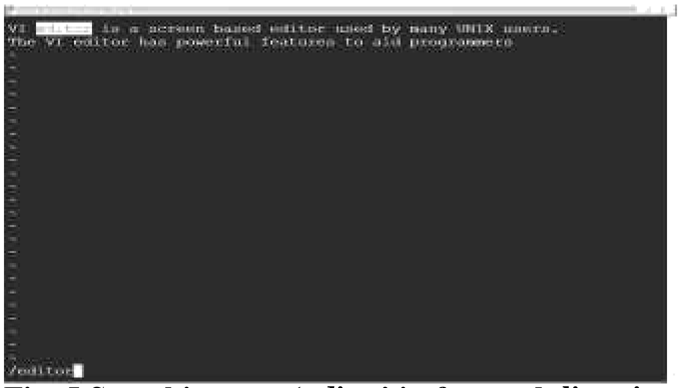
Showing Line number:

Command	**Explanation**
:set number	**Shows line numbers**
:se nu	**Shows line numbers // short cut for set number**
:set nonumber	**Hides line numbers**
:se nonum	**Hides line numbers // short cut**
:set number!	**Toggle between showing and hiding line numbers**
:se nu!	**Toggle between showing and hiding line numbers**

Searching for a Text

Search for a string or a character is possible with vi in command mode. For a string search, the / and **?** commands are used. When you start these commands, the command just typed will be shown on the bottom line as shown in Fig.1 where you type the particular string to look for. These two commands differ only in the direction where the search takes place. The / command searches forwards (downwards) in the file, while the **?** command searches backwards (upwards) in the file. List of commands are as follows:

Command	**Explanation**
/text	Search forward (down) for text
?text	Search backward (up) for text
fchar	Search forward for a character on current line (e.g. fb)
Fchar	Search backward for a character on current line



```
VI editor is a screen based editor used by many UNIX users.
The VI editor has powerful features to aid programmers
~
~
~
/editor
```

Fig. 5 Searching text '*editor*' in forward direction

Substituting the Text

If you want to do substitutions over a range of lines, or throughout the file, the **s** command is used as follows:

:n1,n2s/old/new/gc

n1	is the beginning line number
n2	is the ending line number
s	means to substitute text matching the pattern (**old**) with text specified by (**new**)
g	stands for global (optional). Indicates you want to substitute all occurrences on the indicated lines. If you don't use **g**, the editor substitutes only the first occurrence on the indicated lines.
c	stands for confirm (optional). It indicates you want to confirm each substitution before vi completes it.

Examples:

:%s/old/new/g	Substitutes old with new throughout the line
:.,$s/old/new/g	Substitutes old with new from the current cursor position to the end of the line

How to recover your work if something goes wrong!!!

The vi editor edits a temporary copy of your file, and after the editing is complete, or when you tell it to save, it puts the contents of the temporary copy into the original file. If something goes wrong while you are editing your file, the vi editor will attempt to save whatever work you had in progress, and store it for later recovery. If you were editing the file, and you accidentally got logged out, then the **-r** option of the vi editor helps.

Use the following command to open the file:

```
vi -r first
```

It will show you the temporary file options for recovery. The **-r** option stands for recovery.

After using the **-r** option once, you MUST save what you have recovered to the actual file.