

Unix Programming

UNIT-II

B.Tech(CSE)-V SEM

UNIT-II : THE FILE SYSTEM

- Types of Files, Directories and Files
- UNIX File System
- Absolute and relative pathnames
- File Attributes and Permissions
- File Command -knowing the File Type
- Chmod Command- Changing File Permissions
- Chown Command-Changing the Owner of a File
- Chgrp Command- Changing the Group of a File
- Vi editor-editing with vi, moving the cursor, editing, copying and moving text, pattern searching

File Types

- A file is a collection of related information that is resident in the file store and is identified by a unique filename. The UNIX operating system understands three different types of files:
 - **Ordinary files**
 - **Directory files**
 - **Special files or Device Files**

File Types (Contd...)

1. Ordinary File

- A regular file simply holds data.
- Used to store your information, such as some text you have written or an image you have drawn. This is the type of file that you usually work with.
 - Always located within/under a directory file
- Regular files can hold ASCII text, binary data, image data, databases, application-related data, and more.
- Do not contain other files

File Types (Contd...)

2. Directory File

- Directories are files that contain other files and sub-directories.
- Directories are used to organize the data by keeping closely related files in the same place.
- The directories are just like the folders in windows operating system.
- The kernel alone can write the directory file. When a file is added to or deleted from this directory, the kernel makes an entry.
- A directory file can be visualized as the branch of the UNIX tree.

File Types (Contd...)

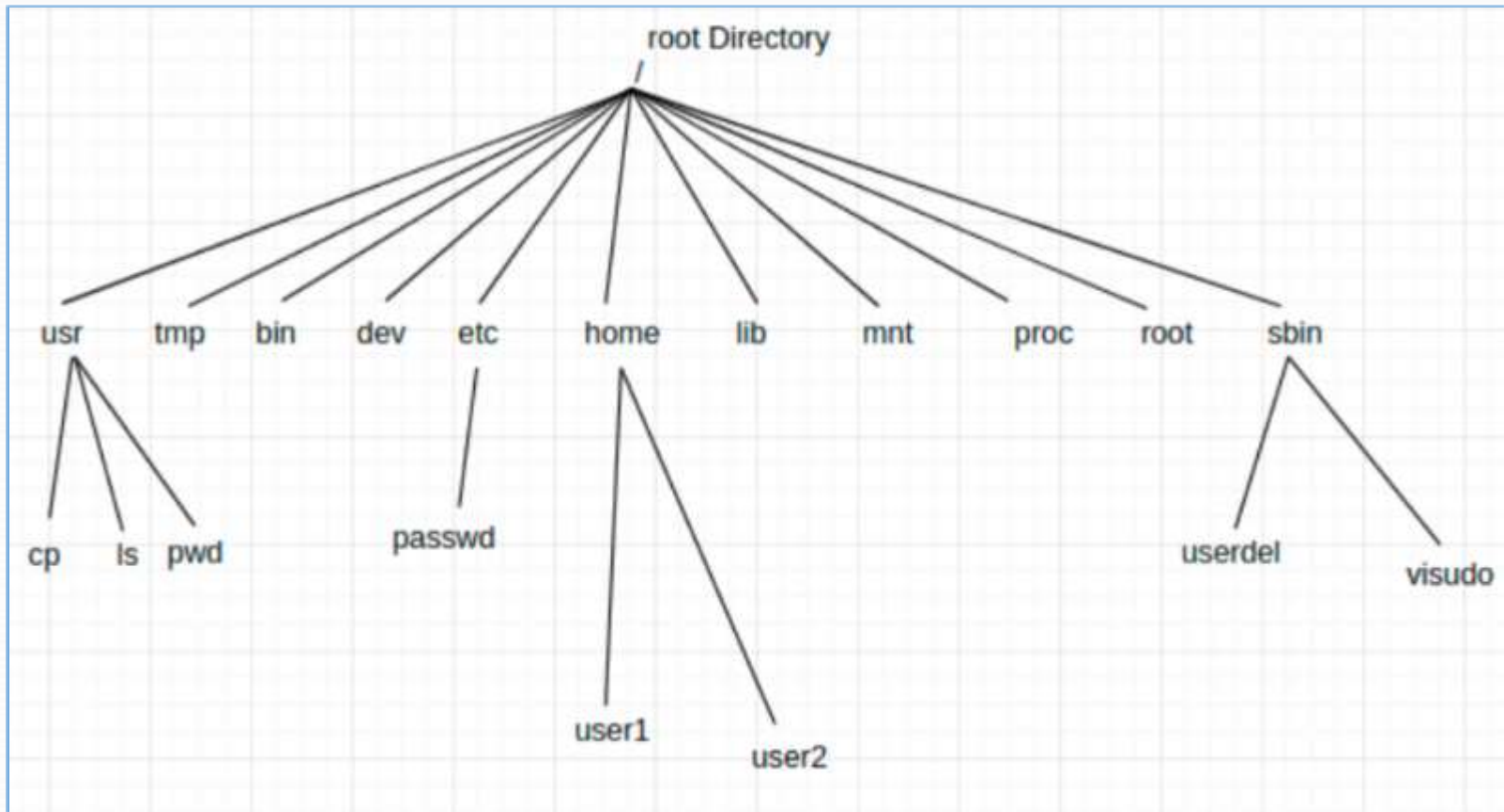
3. Special or Device Files

- Used to represent a real physical device such as a printer, tape drive or terminal, used for Input/Output (I/O) operations.
- On UNIX systems there are two flavors of special files for each device, character special files and block special files. Linux systems only provide one special file for each device.
 - When a **character special file** is used for device I/O, data is transferred one character at a time. This type of access is called raw device access.
 - When a **block special file** is used for device I/O, data is transferred in large fixed-size blocks. This type of access is called block device access.

The File System Hierarchy

- The File system is a hierarchical system of organizing files and directories.
- The top level in the hierarchy is called the '**root**' and holds all files and directories in the filesystem.
- All the data of Unix is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the **filesystem**.
- The name of the root directory is /

The File System Hierarchy (Contd...)



The File System Hierarchy (Contd...)

/	<ul style="list-style-type: none">• Root Directory• It is the top level directory• It is the parent directory for all the other directories• The only root user has the permission to write under this directory.• It can be Considered same as C:\ of Windows
/home	<ul style="list-style-type: none">• It is home directory for the normal users (non-root user)• It provides the working environment for other users (other than root)• It can be considered same as C:\Documents and Settings\Username of Windows
/etc	<ul style="list-style-type: none">• It contains all configurations file• Examples: /etc/passwd → User information is managed here , /etc/hosts → Local DNS Entries etc• It Can be considered same as C:\Windows\system32\drivers\
/boot	<ul style="list-style-type: none">• It Contain all the bootable files in the Linux• Example: vmlinuz (Kernel), Initrd (initial Ram disk) and GRUB (Grand unified bootloader)
/usr	<ul style="list-style-type: none">• By default, software is installed in this directory• Can be considered same as C:\program files of Windows
/bin	<ul style="list-style-type: none">• It Contains commands used by all the users in the system.• It contains the command which can also be executed even in the single user mode.• Example: cd, cp, less etc..

The File System Hierarchy (Contd...)

<i>/sbin</i>	<ul style="list-style-type: none">• It contains command used by the only superuser (root)• Normal user does not have permission to use command which is under /sbin• Example: fdisk, mount, iptables etc..
<i>/lib</i>	<ul style="list-style-type: none">• It contains library files used by operating systems• It can be considered same as DLL (Dynamic Link Libraries) of Windows• Library files in Linux have usually shared object files.
<i>/proc</i>	<ul style="list-style-type: none">• It contains process files• It is often called as virtual directory• Content of /proc directory is not static, they keep on changing• It also contains a lot of important information used by operating system.• Example: /proc/meminfo -> information of RAM, /proc/cpuinfo -> CPU information etc.
<i>/dev</i>	<ul style="list-style-type: none">• Contains device files like /dev/hda for hard disk, /dev/cdrom for cd ROM• Its similar to the device manager of Windows.
<i>/media</i>	<ul style="list-style-type: none">• It Contains all the removable media devices like CD-ROM, Pendrive etc.
<i>/var</i>	<ul style="list-style-type: none">• It has variables data like mails, logs etc.
<i>/mnt</i>	<ul style="list-style-type: none">• It is default mount point for mounting any partition.• By default, this directory is empty.
<i>/tmp</i>	<ul style="list-style-type: none">• Its a temporary directory used to hold temporary files

Absolute and Relative Pathnames in UNIX

- A path is a unique location to a file or a folder in a file system of an OS.
- A path to a file is a combination of / and alpha-numeric characters.
- **Pathnames can be categorized in two types:**
 - (a) Absolute path name
 - (b) Relative path name

Absolute and Relative Pathnames (Contd..)

(a) **Absolute path** : (start from the / directory)

An absolute path is defined as the specifying the location of a file or directory from the root directory(/). In other words we can say *absolute path* is a complete path from start of actual filesystem from / directory.

Example:

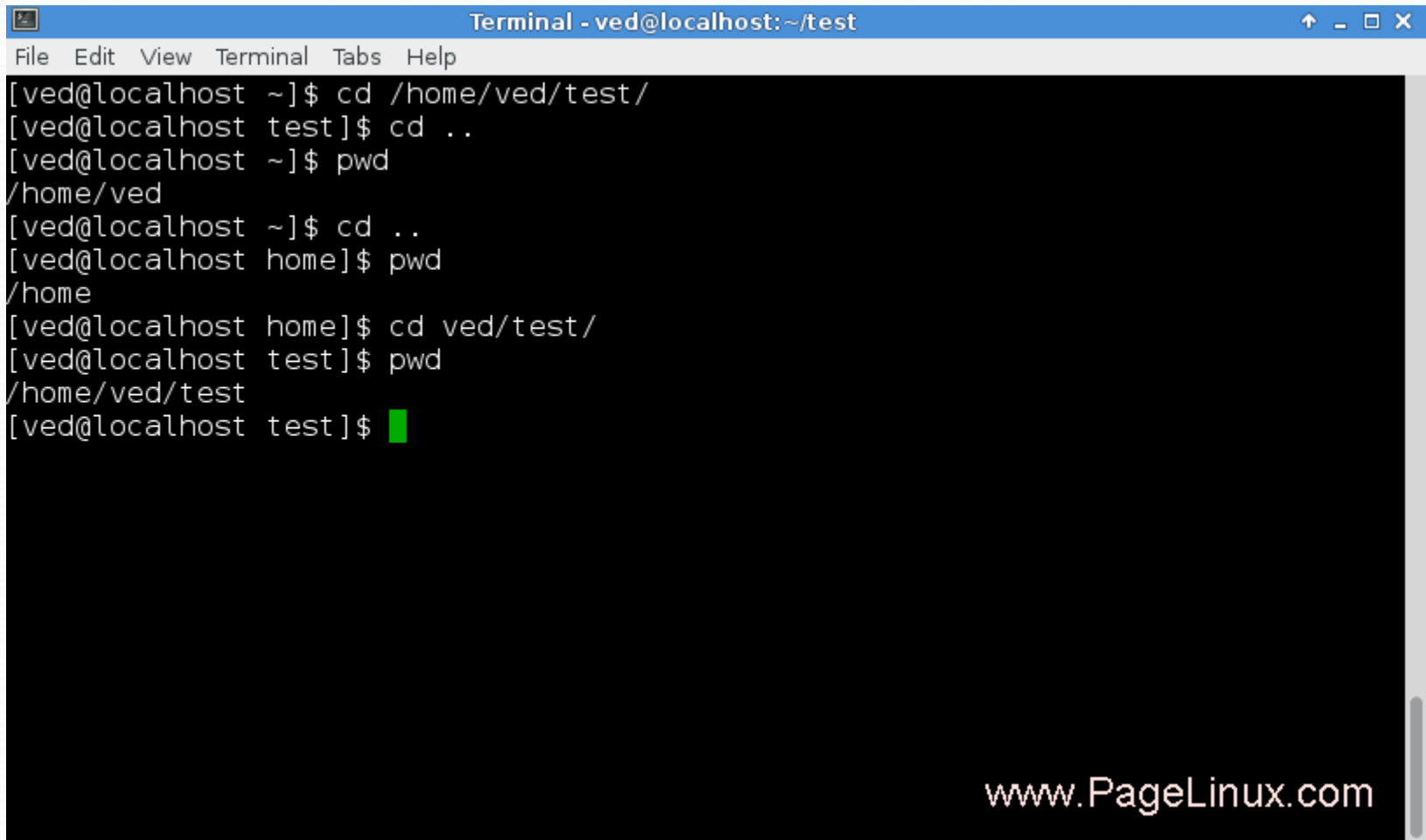
/home/user/Document/srv.txt

/root/data/dev.jpg

/var/log/messages

Absolute and Relative Pathnames (Contd..)

All are absolute Path



```
Terminal - ved@localhost:~/test
File Edit View Terminal Tabs Help
[ved@localhost ~]$ cd /home/ved/test/
[ved@localhost test]$ cd ..
[ved@localhost ~]$ pwd
/home/ved
[ved@localhost ~]$ cd ..
[ved@localhost home]$ pwd
/home
[ved@localhost home]$ cd ved/test/
[ved@localhost test]$ pwd
/home/ved/test
[ved@localhost test]$
```

www.PageLinux.com

Absolute and Relative Pathnames in (Contd..)

(b)Relative path : (start from the current directory)

It is defined as the path related to the present working directly(pwd). It starts at your current directory and **never starts with a /** .

Example: suppose I am located in /home/user1 and I want to change directory to /home/user1/Documents. I can use *relative path* concept to change directory to Documents.

Absolute and Relative Pathnames (Contd..)

Example: Here are two examples for changing directory, 1st by using relative path, 2nd by using absolute path.

```
$ pwd
```

```
/home/user1
```

```
$cd Documents/ (using relative path)
```

```
$pwd
```

```
/home/user1/Documents
```

(or)

```
$ pwd
```

```
/home/user1
```

```
$cd /home/user1/Documents/ (using absolute path)
```

```
$ pwd
```

```
/home/user1/Documents
```

File Attributes

Any type of file will have :

1. File type and File Access Permissions
2. Number of links
3. File Owner
4. Group Owner (Group Name)
5. File Size (in bytes)
6. Month
7. Date
8. Time of the last modification of the file
9. Name of the File

File Attributes (Contd..)

```
shum@sol1:~$ ls -l
total 20
drwx----- 2 shum staff 4096 Jan 16 22:04 Mail
drwx----- 3 shum staff 4096 Jan 16 14:15 csc128
drwxr-xr-x  2 shum staff 4096 Jan 13 16:42 public
drwxr-xr-x  2 shum staff 4096 Jan 16 14:07 public_html
-rw-r--r--  1 shum staff 628 Jan 15 20:04 verse
```

Diagram illustrating the components of the `ls -l` output:

- File type:** Indicated by the first character of the permissions (e.g., `d` for directory, `-` for regular file).
- Permissions:** Indicated by the next nine characters (e.g., `rw-r--r--`).
- Number of hard links:** Indicated by the number following the permissions (e.g., `1`).
- User (owner) name:** Indicated by the user name (e.g., `shum`).
- Group name:** Indicated by the group name (e.g., `staff`).
- Size:** Indicated by the file size in bytes (e.g., `628`).
- Date/Time last modified:** Indicated by the date and time (e.g., `Jan 15 20:04`).
- Filename:** Indicated by the file name (e.g., `verse`).

Legend for permissions:

- rwx:** Read, Write, Execute permissions.
- rwx:** Read, Write, Execute permissions (for owner, group, and others).
- rwx:** Read, Write, Execute permissions (for owner, group, and others).

file command knowing file type

- The file command determines the file type of a file. It reports the file type in;
 - Human readable format (Ex: ASCII text)
 - MIME type (Ex: 'text/plain; charset=us-ascii')

Syntax:

`$ file [options] filename`

Example 1: To view ASCII type of a file

- `$file sample` **# filename along with the file type will be printed to standard output**

Output: sample: ASCII text

- `$file -b sample` **# To show just file type pass the -b option**

Output: ASCII text

file command knowing file type (Contd...)

- **Example 2:** To view mime type of a file

To view MIME type of a file rather than the human readable format pass the **-i** option

```
$file -i sample
```

output: sample: text /palin charset=us-ascii

This can be combined with the **-b** option to just show the MIME type:

```
$file -i -b sample
```

output: text /palin; charset=us-ascii

file command knowing file type (Contd...)

- **Example 3:**

```
$file /dev/sda1
```

output: /dev/sda1 :block special

```
$file /bin/cat
```

- **Example 4:**

```
$file /bin
```

output: /bin: directory

- **Example 5:**

```
$file a.out
```

output: ELF 32-bit LSB executable

- **Example 6:** How to view compressed files without decompressing (use -Z option)

```
$file -z a.gz
```

File Permissions

- A set of 9 characters denote the permissions. There are 3 types of permissions to a file.
 - 1. read(r):** If authorized, the user can read the contents of the file.
 - 2. Write(w):** If authorized, the user can modify the file.
 - 3. Execute(x):** If authorized, the user can execute the file as a program.
- The permissions has also numeric representation like below:

Permission	Number
read (r)	4
Write (w)	2
Execute (x)	1

File Permissions (Contd...)

- These are 3 entities to which any combination of these permissions are assigned. Every file in Unix has the following attributes:
 1. **Owner / user permissions(u)** – A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.
 2. **Group permissions (g)** – The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
 3. **Other (world) permissions (o)** – The permissions for others indicate what action all other users can perform on the file

File Permissions (Contd...)

```
# ls -l file
-rw-r--r-- 1 root root 0 Nov 19 23:49 file
```

File type

Owner (rw-)

Group (r- -)

Other (r - -)

r = Readable
w = Writeable
x = Executable
- = Denied

chmod - Changing Permissions

- To change the file or the directory permissions, we can use the chmod (change mode) command. There are two ways to use chmod:

a) Symbolic mode

b) Absolute mode

a) symbolic mode: With symbolic mode we can we can add, delete, or specify the permissions with the following syntax:

Syntax:

\$chmod [who] [+/-/=] [permissions] filename

Here :

- **who refers** to whom the permissions are to be assigned. It may be the **user /owner(u), group(g) and others(o)**.

chmod - Changing Permissions (Contd...)

- **[+/-/=] :**

chmod operator	Description
+	Adds the designated permission(s) to a file or directory.
-	Removes the designated permission(s) from a file or directory
=	Sets the designated permission(s) and take away all other permissions present.

- **Permissions:**

r	read
w	Write
x	execute

chmod - Changing Permissions (Contd...)

Examples on Symbolic mode:

- `$chmod +w a` #It will give write permission to all
- `$chmod go -x a` # In order to take away execute permission from group and others
- `$chmod go +r, go -w a`
- `$chmod go=r u=rw a`

chmod - Changing Permissions (Contd...)

b) Absolute mode :

The second way to modify permissions with the chmod command is to use a number to specify each set of permissions for the file.

Examples:

- | | |
|------------------|----------------|
| 1. \$chmod 744 a | (rwx r —r--) |
| 2. \$chmod 654 a | (rw- r-x r- -) |
| 3. \$chmod 457 a | (r- - r-x rwx) |
| 4. \$chmod 400 a | (r-----) |
| 5. \$chmod 010 a | (--- --x ---) |

umask

- It decide the default permissions
- umask stands for “user file creation mask”
- The term mask implying which permissions has to mask (or) hide.
- To view the current mask value, simply type umask without any arguments:

- **\$umask**

0022

Here;

- First ‘o’ indicates what follows is an octal number.
- Next, ‘o’ indicates that no permission is denied to owner.
- Where as for group and others write(2) permission is denied.

umask (Contd...)

- **Whenever a file or directory is created** : The default creation permissions for **files** are **666** and for **directories** **777**. To calculate the permission bits of the new files subtract the umask value from the default value.

***For example**, to calculate how umask 022 will affect newly created files and directories, use:*

- **Files:** $666 - 022 = 644$. The owner can read and modify the files. Group and others can only read the files.
- **Directories:** $777 - 022 = 755$. The owner can cd into the directory and list read, modify, create or delete the files in the directory. Group and others can cd into the directory and list and read the files.

umask (Contd...)

- We can change the umask value. Like ,

```
$umask 0242
```

- The file permissions for this are $(666-242=424)$
- The directory permissions are $(777-242=535)$

Changing Owners and Groups

- While creating an account on Unix, it assigns a owner ID and a group ID to each user. All the permissions mentioned above are also assigned based on the Owner and the Groups.
- Two commands are available to change the owner and the group of files:
 - **chown** – The chown command stands for "change owner" and is used to change the owner of a file.
 - **chgrp** – The chgrp command stands for "change group" and is used to change the group of a file.

NOTE: The above commands can only be used or performed by the root user or the user who is having rights by given by the root user. So the standard user can not change the group (or) owner for that we need to contact administrator.

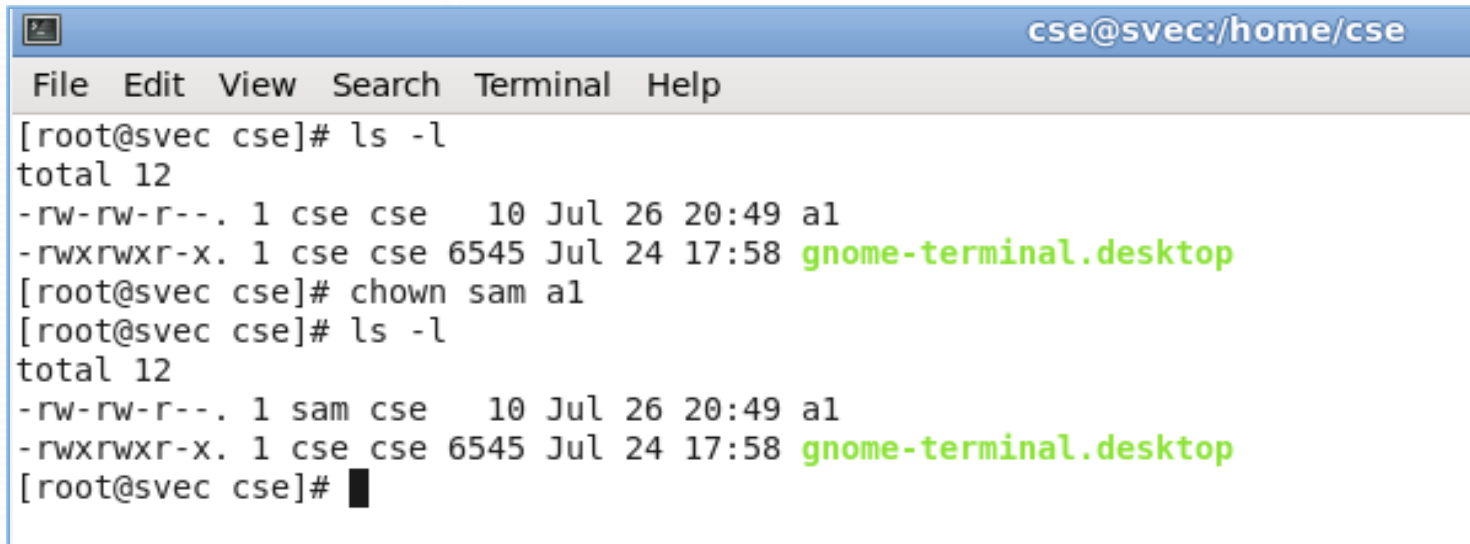
chown : Changing Ownership

- The chown command changes the ownership of a directory / file. The basic syntax is as follows :
- **SYNOPSIS:**

#chown [OPTIONS] [OWNERNAME] [DIR /FILE]

Example:

#chown sam a1



```
cse@svec:/home/cse
File Edit View Search Terminal Help
[root@svec cse]# ls -l
total 12
-rw-rw-r--. 1 cse cse  10 Jul 26 20:49 a1
-rwxrwxr-x. 1 cse cse 6545 Jul 24 17:58 gnome-terminal.desktop
[root@svec cse]# chown sam a1
[root@svec cse]# ls -l
total 12
-rw-rw-r--. 1 sam cse  10 Jul 26 20:49 a1
-rwxrwxr-x. 1 cse cse 6545 Jul 24 17:58 gnome-terminal.desktop
[root@svec cse]#
```


chgrp : Changing group

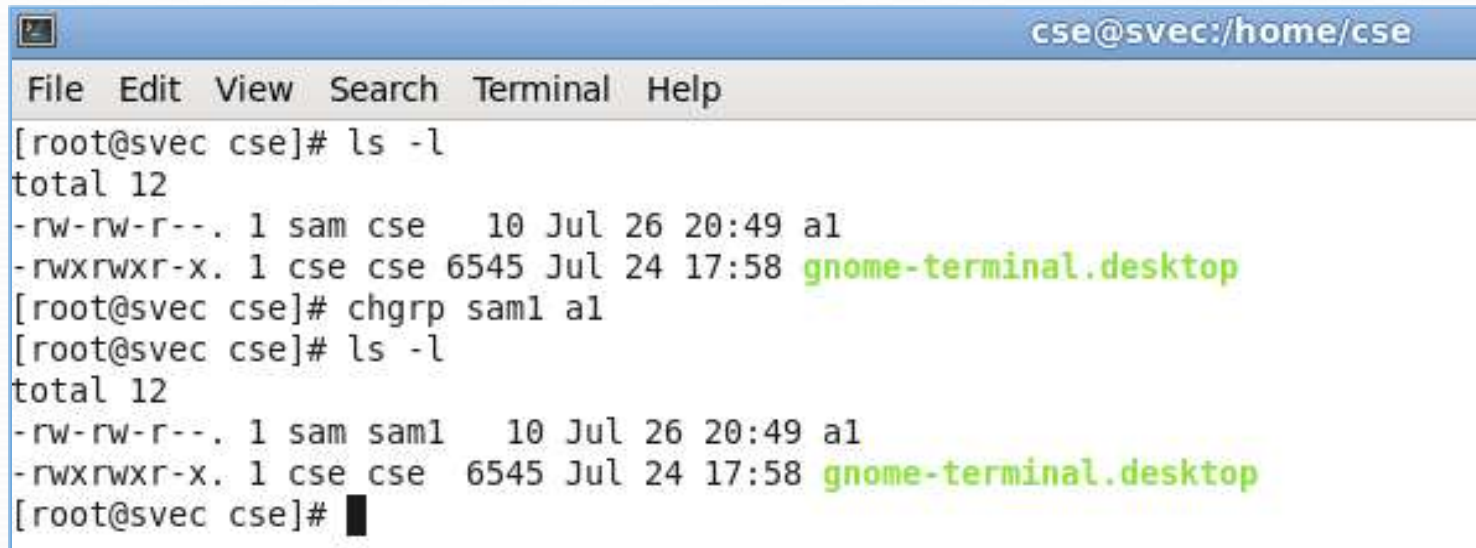
- The chgrp command changing the existing group of a file / directory. The basic syntax is as follows :

- **SYNOPSIS:**

#chgrp [OPTIONS] [GROUPNAME] [DIR /FILE]

- **Example:**

#chown sam1 a1



```
cse@svec:/home/cse
File Edit View Search Terminal Help
[root@svec cse]# ls -l
total 12
-rw-rw-r--. 1 sam cse 10 Jul 26 20:49 a1
-rwxrwxr-x. 1 cse cse 6545 Jul 24 17:58 gnome-terminal.desktop
[root@svec cse]# chgrp sam1 a1
[root@svec cse]# ls -l
total 12
-rw-rw-r--. 1 sam sam1 10 Jul 26 20:49 a1
-rwxrwxr-x. 1 cse cse 6545 Jul 24 17:58 gnome-terminal.desktop
[root@svec cse]#
```

changing Owners and Groups (Contd..)

- To change both user and group name in a single command:
- **Syntax:** # chown ownername.groupname filename
- **Example:** # chown svec.svec1 f

```
cse@svec:/home/cse
File Edit View Search Terminal Help
[root@svec cse]# useradd svec
[root@svec cse]# groupadd svec1
[root@svec cse]# ls -l
total 12
-rw-rw-r--. 1 svec svec1 10 Jul 26 20:49 a1
-rwxrwxr-x. 1 cse cse 6545 Jul 24 17:58 gnome-terminal.desktop
[root@svec cse]# ls /home/
cse sam svec
[root@svec cse]# touch f
[root@svec cse]# ls -l
total 12
-rw-rw-r--. 1 svec svec1 10 Jul 26 20:49 a1
-rw-r--r--. 1 root root 0 Jul 26 22:28 f
-rwxrwxr-x. 1 cse cse 6545 Jul 24 17:58 gnome-terminal.desktop
[root@svec cse]# chown svec.svec1 f
[root@svec cse]# ls -l
total 12
-rw-rw-r--. 1 svec svec1 10 Jul 26 20:49 a1
-rw-r--r--. 1 svec svec1 0 Jul 26 22:28 f
-rwxrwxr-x. 1 cse cse 6545 Jul 24 17:58 gnome-terminal.desktop
[root@svec cse]#
```

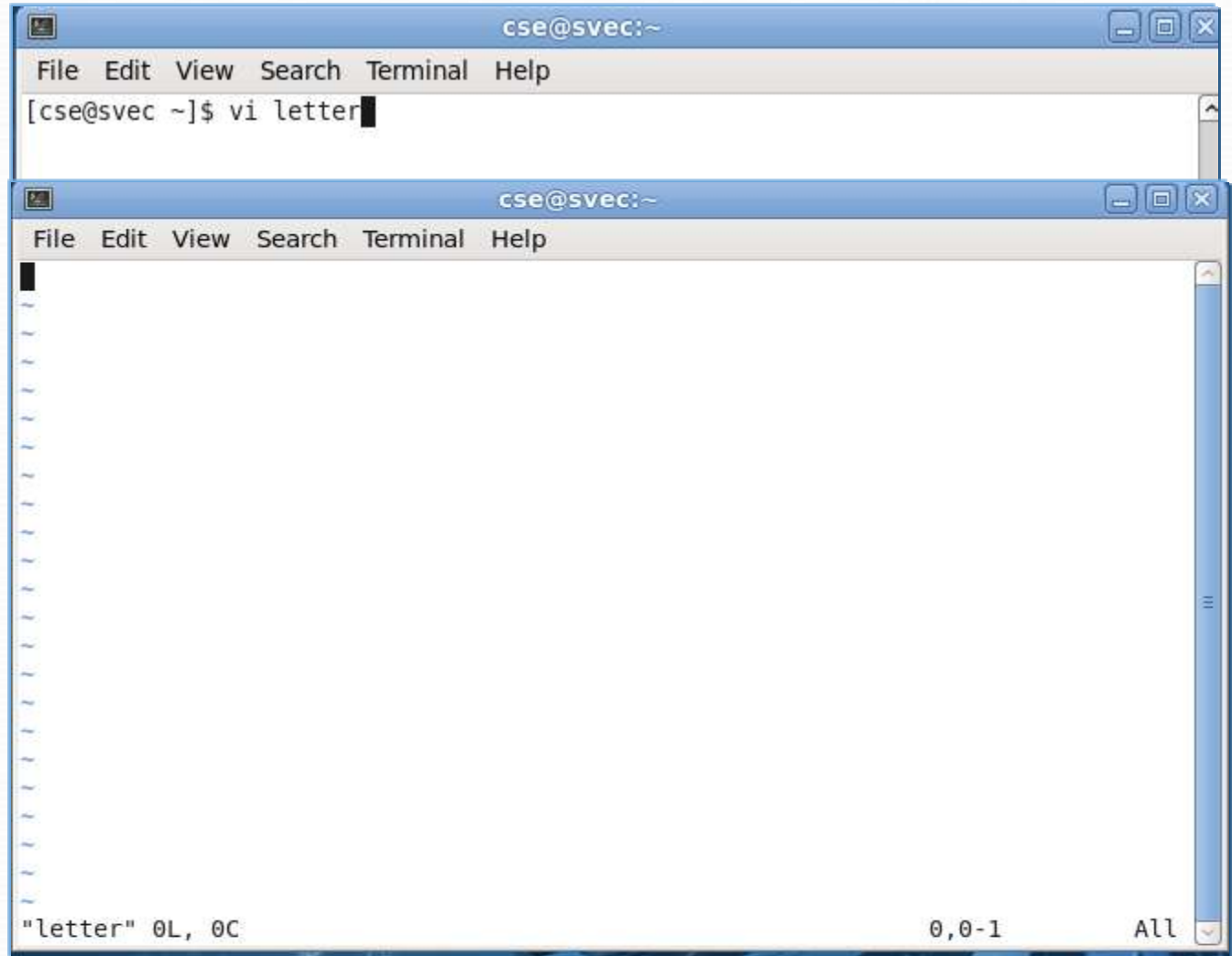
vi Editor

- The default editor that comes with the UNIX operating system is called vi (visual editor).
- A text editor has a wide range of **applications** such as:
 - Creating and managing documents
 - Writing programs and utilities
 - Writing mail messages
- **Functions of an vi editor:**
 - create a new file from scratch
 - Opening and Editing an existing file
 - Copying and pasting text
 - Searching for a text
 - Handling a large amount of data.

vi Editor (Contd...)

- **Syntax:** `$vi filename`

Example:



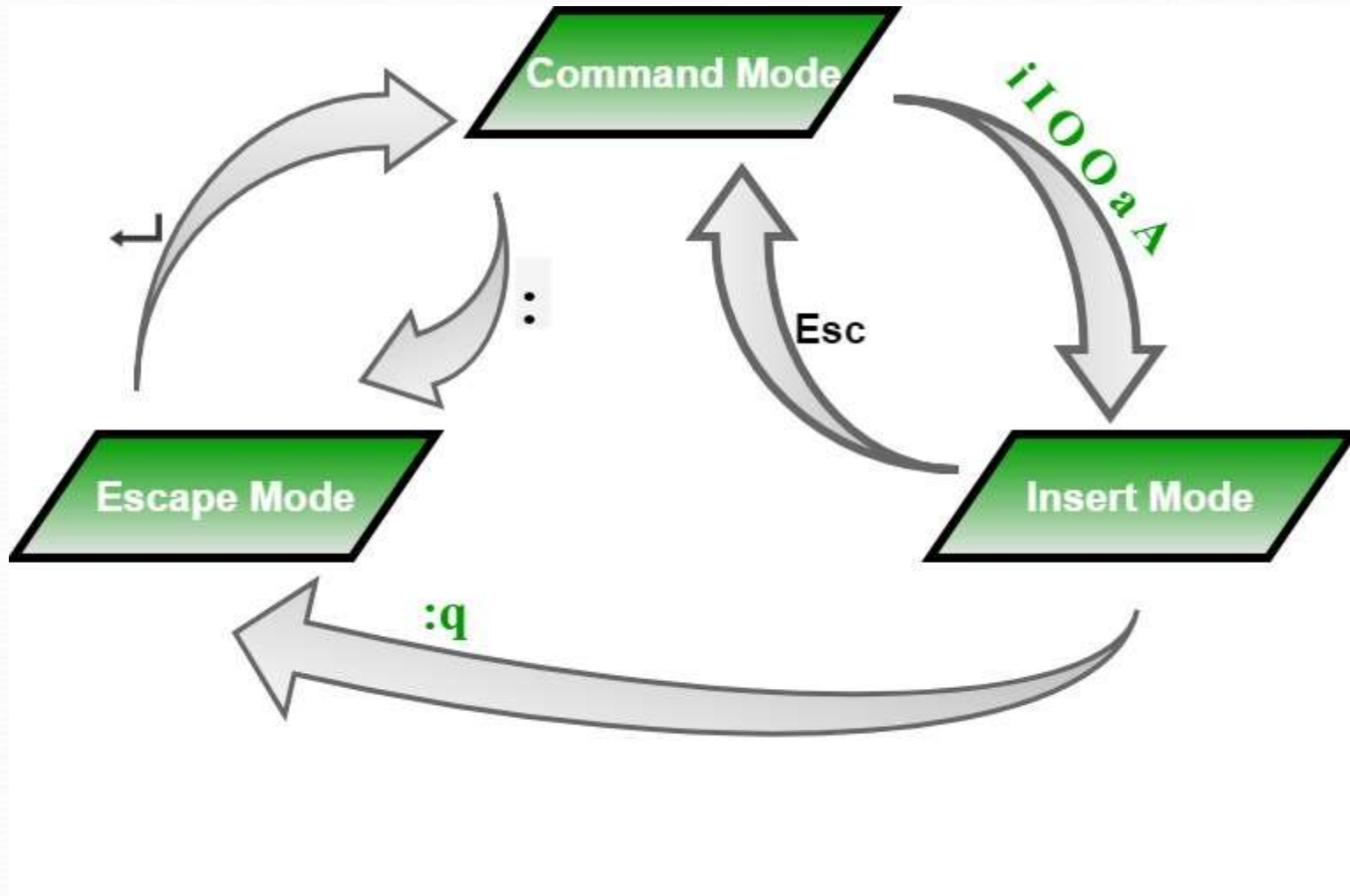
```
cse@svec:~  
File Edit View Search Terminal Help  
[cse@svec ~]$ vi letter
```



```
cse@svec:~  
File Edit View Search Terminal Help  
"letter" 0L, 0C 0,0-1 All
```

vi Editor (Contd...)

vi editor has 3 modes



vi Editor (Contd...)

1.command mode: In this mode, all the keys work as commands. These keys are used for inserting, appending, deleting, opening new lines, moving the cursor over the paragraphs and sentences, etc. In this mode, the keys are not displayed but each key performs an operation.

By default the vi editor is in command mode, hence we cannot type text in command mode. In order to write programs or text in vi editor, we need to switch to the insert mode which can be done by pressing the escape button.

U	To undo the previous changes
Ctrl+R	To redo the changes
yy	To copy a line
nyy	To copy n lines (5yy or 4yy)
p	To paste line below the cursor position
P	To paste line above the cursor position
dw	To delete the word letter by letter (like backspace)
x	To delete the word letter by letter (like DEL key)
dd	To delete entire line
ndd	To delete en number of lines from cursor position
/	To search a word in the file
%s/oldword/newword/g	Replace

vi Editor (Contd...)

2. Insert mode: In this mode, we can insert, append, edit or replace texts. We can switch from the command mode to Insert mode by pressing the escape button and then press I or A to enter into insert mode.

i	To begin insert mode at the cursor position
I	To insert at the beginning of line
a	To append to the next words letter
A	To append at the end of the line
o	To insert a new line below the cursor position
O	To insert a new line above the cursor position

vi Editor (Contd...)

3. Ex command mode: This mode is used for entering commands at the bottom line of the vi editor called as a command line.

To switch to Ex command mode press escape key then type: (colon).

In order to save the contents and quit from the vi editor press wq after the : (colon). i.e : wq.

- a) All the Block commands work in ex command mode
- b) Line numbers must be associated with the text before we issue any block commands

To set numbers: (Esc): set number /(Esc) : set nu

:nd	Delete nth line
:m,nd	Delete lines from m to n
:n mo p	move line n after line p
m,n mo p	moves lines m to n after line p
:m co p	copies lines m after line p

vi Editor (Contd...)

Vi editor saving and quitting commands:

- **:w** -Save the contents of the file.
- **:q** – Quit from vi editor.
- **:q!** -quit from vi editor by discarding any changes.
- **:wq** -Save the file and quit from the vi editor.
- **Moving within a file** (we need to be in the command mode to move within a file)
 - **k** - Move cursor up
 - **j** - Move cursor down
 - **h** - Move cursor left
 - **l** - Move cursor right



Thank You