***LINUX PROGRAMMING AND SCRIPTING***

***UNIT – 1***

***LINUX BASICS***

**What is Operating System?**

* An Operating System (OS) is an interface between computer user and computer hardware. An operating system is software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.
* Some popular Operating Systems include Linux, Windows, OS X, VMS, OS/400, AIX, z/OS, etc.
* Following are some of important functions of an operating System.

1. Memory Management
2. Processor Management
3. Device Management
4. File Management
5. Security
6. Control over system performance
7. Job accounting
8. Error detecting aids
9. Coordination between other software and users

**What is UNIX?**

* UNIX is an operating System developed in 1969 by a group of AT&T employees at Bell Labs, including Ken Thompson, Dennis Ritchie.
* The UNIX operating system is a set of programs that act as a link between the computer and the user.
* The computer program that allocates the system resources and coordinates all the details of the computer's internal’s is called the operating system or kernel.
* Users communicate with the kernel through a program known as the shell. The shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.
* Several people can use a UNIX computer at the same time; hence UNIX is called a multiuser system.
* A user can also run multiple programs at the same time; hence UNIX is called multitasking or Multiprogramming System.

**UNIX Architecture:**

* Here is a basic block diagram of a UNIX system:



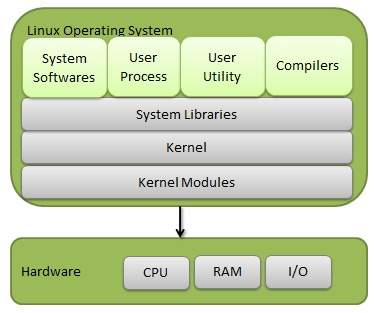
* The main concept that unites all versions of UNIX is the following four basics:
* **Kernel:**The kernel is the heart of the operating system. It interacts with hardware and most of the tasks like memory management, task scheduling and file management.
* **Shell:** The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. The shell uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are most famous shells which are available with most of the UNIX variants.
* **Commands and Utilities:** There are various command and utilities which you would use in your day to day activities. **cp, mv, cat** and **grep** etc. are few examples of commands and utilities. All the commands come along with various optional options.
* **Files and Directories:** All data in UNIX is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the file system.

**INTRODUCTION TO LINUX:**

* Linux is one of popular version of UNIX operating System (Linux is a UNIX-base operating system). It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.
* UNIX was initially developed for mainframes and minicomputers especially for universities and carried a universal compatibility. When microcomputers were being developed, many UNIX versions supported PC architecture but lacked compatibility and speed. As a result, the users started running MS-DOS or Windows on their home computer.
* In 1990, PCs were fully powered by UNIX but still free software was unavailable. In 1991, Linus Torvalds began developed an operating system called Linux which is member of large family UNIX-like OS.
* Linux was developed for IBM compatible personal computer. All features of UNIX were added in Linux within few years, which gave rise to a mature operating system called Linux. Today Linux is supported by all kinds of workstations, home user PC and both server and client.

**Components of Linux System**

* Linux Operating System has primarily three components.
* **Kernel** − Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
* **System Library** − System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not require kernel module's code access rights.
* **System Utility** − System Utility programs are responsible to do specialized, individual level tasks.

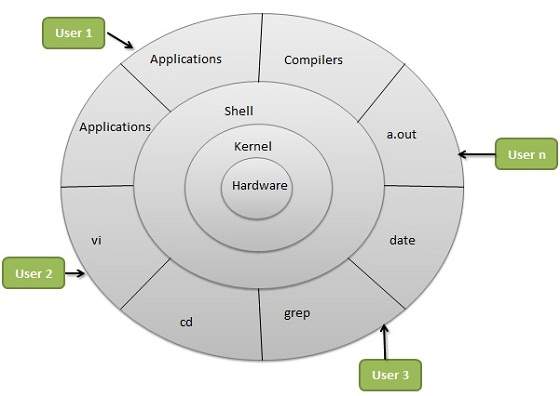


**Basic Features:**

* Following are some of the important features of Linux Operating System.
* **Portable** − Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.
* **Open Source** − Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
* **Multi-User** − Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
* **Multiprogramming** − Linux is a multiprogramming system means multiple applications can run at same time.
* **Hierarchical File System** − Linux provides a standard file structure in which system files/ user files are arranged.
* **Shell** − Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.
* **Security** − Linux provides user security using authentication features like password protection / controlled access to specific files / encryption of data.

**Architecture:**

* The following illustration shows the architecture of a Linux system:



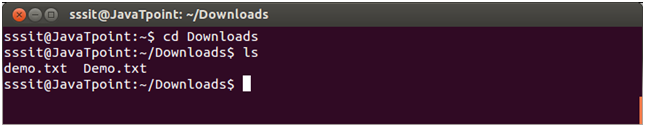
* The architecture of a Linux System consists of the following layers:
* **Hardware layer** − Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
* **Kernel** − It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
* **Shell** − An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.
* **Utilities** − Utility programs that provide the user most of the functionalities of an operating systems.

### Advantages of Linux:

* **Low cost:** There is no need to spend time and huge amount money to obtain licenses since Linux and much of its software come with the GNU General Public License. There is no need to worry about any software's that you use in Linux.
* **Stability:** Linux has high stability compared with other operating systems. There is no need to reboot the Linux system to maintain performance levels. Rarely it freeze up or slow down. It has continuous up-times of hundreds of days or more.
* **Performance:** Linux provides high performance on various networks. It has the ability to handle large numbers of users simultaneously.
* **Networking:** Linux provides a strong support for network functionality; client and server systems can be easily set up on any computer running Linux. It can perform tasks like network backup faster than other operating systems.
* **Flexibility:** Linux is very flexible. Linux can be used for high performance server applications, desktop applications, and embedded systems. You can install only the needed components for a particular use. You can also restrict the use of specific computers.
* **Compatibility:** It runs all common Unix software packages and can process all common file formats.
* **Wider Choice:** There is a large number of Linux distributions which gives you a wider choice. Each organization develops and support different distribution. You can pick the one you like best; the core functions are the same.
* **Fast and easy installation:** Linux distributions come with user-friendly installation.
* **Better use of hard disk:** Linux uses its resources well enough even when the hard disk is almost full.
* **Multitasking:** Linux is a multitasking operating system. It can handle many things at the same time.
* **Security:** Linux is one of the most secure operating systems. File ownership and permissions make Linux more secure.
* **Open source:** Linux is an Open source operating systems. You can easily get the source code for Linux and edit it to develop your personal operating system.

**FILE SYSTEM OF THE LINUX:**

* In Linux system, everything is a file and if it is not a file, it is a process. A file doesn't include only text files, images and compiled programs but also include partitions, hardware device drivers and directories. Linux consider everything as a file.
* File system organizes the data in systematic way.
* Files are always case sensitive. Let's understand it through an example.



* In above example, we have two files named as 'Demo.txt' and 'demo.txt'. Although, they both share the same name but still they are two different files.

**Types of Files:**

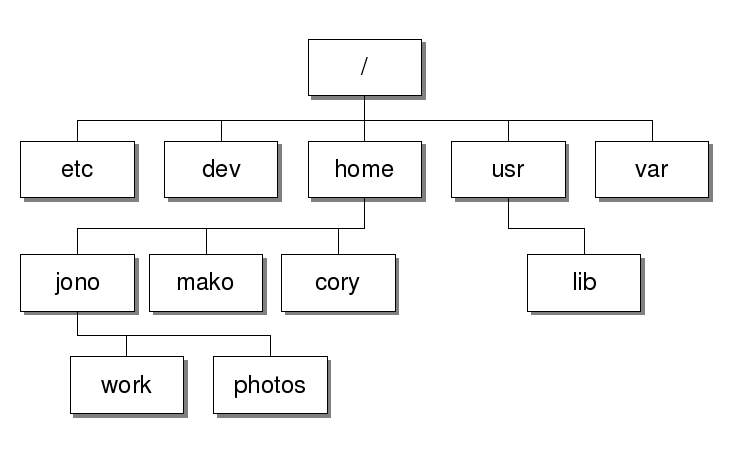
1. **Regular files (-):** It contains programs, executable files and text files.
2. **Directory files (d):** It is shown in blue color. It contain list of files.
3. **Special files**
   * **Block file (b)**
   * **Character device file (c)**
   * **Named pipe file (p)**
   * **Symbolic link file (l)**
   * **Socket file (s)**

## Linux File System (LFS):

* Linux accesses every object as file.
* Files are systematically organized in directories.
* Linux starts file system with root directory (**/**).
* All files and directories are created and managed under the root directory. Since root directory stands on the top in file system, it has no parent directory. Besides root directory, every directory in Linux has a parent directory.
* Linux allows us to create as many files and directories as we want. We can create files under the existing directories or may create new directories.

### System Directories:

* System directories contain files, software, applications and scripts which are required to run and maintain the Linux. System directories are automatically created during the installation.
* Following figure illustrates some common system directories with their location in LFS.



**Common System directories:**

|  |  |
| --- | --- |
| **Directory** | **Description** |
| / | First directory in Linux File System. It is also known as root directory or main directory. All files and directories are created and managed under this directory. |
| /home | Default directory for user data. Whenever we add a new user, Linux automatically creates a home directory matching with his username in this directory. Whenever user login, Linux starts his login session from home directory. |
| /root | This is the home directory for root user. Root user is the super user in Linux. For security reason Linux creates a separate home directory for root user. Root user account is also being created during the installation automatically. |
| /bin | This directory contains standard commands files. Commands stored in this directory are available for all users and usually do not require any special permission to run. |
| /sbin | This directory contains system administration commands files. Commands stored in this directory are available only for root user and usually requires special privilege to run. |
| /usr | This directory contains user application software files, third party software and scripts, document files and libraries for programming languages. |
| /var | This directory stores variable data files such as printing jobs, mail box etc. |
| /etc | This directory contains system configuration files. |
| /boot | This directory contains Linux boot loader file. |
| /mnt | This directory is used to mount remote file system and temporary devices such as CD, DVD and USB. |
| /dev | This directory contains device files. Usually files in this directory are dynamically generated and should be never edited. |
| /tmp | This directory provides temporary location for applications. |

## Linux File Naming Convention:

* Unlike Windows operating system Linux is not strict with naming convention.
* We can use any number or letter for file name. We can also use underscore, space, period and comma.
* Some special characters such as question mark, asterisks and slashes are not allowed for file name.
* These characters are reserved for shell functions.
* Just like Windows we can use dot to create a file extension. File extensions are not compulsory in Linux but we should use them wherever possible as they provide a good way to manage files.

###### **Key points:**

1. A Linux file name may have any characters or letters.
2. Maximum length for file name is 256 characters.
3. File name can use space, underscore, minus, period and comma.
4. File name cannot use question marks, asterisks and slashes.
5. File extension is not compulsory. We can create files with file extension or without file extension.
6. To create a hidden file, start its name with dot.
7. If file name has spaces, it needs to be quoted before we can access it on command prompt. On desktop we can use it without quotes.

**Linux File Commands:**

|  |  |
| --- | --- |
| **Command** | **Description** |
| [file](http://www.javatpoint.com/linux-file) | Determines file type. |
| [touch](http://www.javatpoint.com/linux-touch) | Used to create a file. |
| [rm](http://www.javatpoint.com/linux-rm) | To remove a file. |
| [cp](http://www.javatpoint.com/linux-cp) | To copy a file. |
| [mv](http://www.javatpoint.com/linux-mv) | To rename or to move a file. |
| [rename](http://www.javatpoint.com/linux-rename) | To rename file. |

**GENERAL USAGE OF LINUX KERNEL:**

### What is the Kernel?

* The **Linux kernel** is a monolithic [Unix-like](https://en.wikipedia.org/wiki/Unix-like) computer [operating system kernel](https://en.wikipedia.org/wiki/Operating_system_kernel).
* A kernel is the lowest level of easily replaceable software that interfaces with the hardware in your computer.
* It is responsible for interfacing all of your applications that are running in “user mode” down to the physical hardware, and allowing processes, known as servers, to get information from each other using inter-process communication (IPC).

### Different Types of Kernels:

* In general, most kernels fall into one of three types: **monolithic, microkernel, and hybrid.** Linux is a monolithic kernel while OS X (XNU) and Windows 7 use hybrid kernels.

**Micro kernel:**

* A microkernel usually provides only minimal services, such as defining memory address spaces, interprocess communication (IPC) and process management.
* All other functions, such as hardware management, are implemented as processes running independently of the kernel.
* Examples of microkernel operating systems are AIX, BeOS, Hurd, Mach, Mac OS X, [MINIX](http://www.linfo.org/minix.html) and QNX.

**Monolithic Kernel:**

* Monolithic kernels, which have traditionally been used by Unix-like operating systems, contain all the operating system core functions and the *device drivers* (small programs that allow the operating system to interact with hardware devices, such as disk drives, video cards and printers).
* Modern monolithic kernels, such as those of Linux and FreeBSD, both of which fall into the category of Unix-like operating systems, feature the ability to load *modules* at runtime, thereby allowing easy extension of the kernel's capabilities as required, while helping to minimize the amount of code running in kernel space.

**Hybrid Kernel:**

* Hybrid kernels are similar to microkernels, except that they include additional code in kernel space so that such code can run more swiftly than it would were it in user space.
* These kernels represent a compromise that was implemented by some developers before it was demonstrated that pure microkernels can provide high performance.
* Hybrid kernels should not be confused with monolithic kernels that can load modules after booting (such as Linux).

**Usage of Kernel:**

* The kernel is the first part of the operating system to load into memory during *booting* (i.e., system startup), and it remains there for the entire duration of the computer session because its services are required continuously.
* Thus it is important for it to be as small as possible while still providing all the essential services needed by the other parts of the operating system and by the various application programs.
* The kernel performs its tasks, such as executing [*processes*](http://www.linfo.org/process.html) and handling *interrupts*, in kernel space, whereas everything a user normally does, such as writing text in a text editor or running programs in a [GUI](http://www.linfo.org/gui.html) (graphical user interface), is done in [*user space*](http://www.linfo.org/user_space.html).
* When a computer *crashes*, it actually means the kernel has crashed.
* If only a single program has crashed but the rest of the system remains in operation, then the kernel itself has not crashed.
* A crash is the situation in which a program, either a user application or a part of the operating system, stops performing its expected function(s) and responding to other parts of the system. The program might appear to the user to *freeze*.
* If such program is a critical to the operation of the kernel, the entire computer could stall or shut down.
* The kernel provides basic services for all other parts of the operating system, typically including memory management, process management, file management and I/O (input/output) management (i.e., accessing the peripheral devices).
* These services are requested by other parts of the operating system or by application programs through a specified set of program interfaces referred to as *system calls*.
* A few kernels have been designed with the goal of being suitable for use with any operating system.

**BASIC LINUX COMMANDS:**

#### id COMMAND:

id command prints the effective(current) and real userid(UID)s and groupid(GID)s.

#### SYNTAX:

id [options]

#### EXAMPLE:

id -a

**Output:**

uid=501(username) gid=501(username) groups=48(apache), 501(username)

#### man COMMAND:

man command which is short for manual, provides in depth information about the requested command (or) allows users to search for commands related to a particular keyword.

#### SYNTAX :

man command name [options]

#### OPTIONS:

|  |  |
| --- | --- |
| **-a** | **Print a one-line help message and exit.** |
| -k | Searches for keywords in all of the manuals available.. |

#### EXAMPLE:

man mkdir

Display the information about mkdir command.

man man

Display the information about man command.

#### cal COMMAND:

cal command is used to display the calendar.

#### SYNTAX :

cal [options] [month] [year]

#### OPTIONS:

|  |  |
| --- | --- |
| **-1** | **Displays single month as output.** |
| -3 | Displays prev/current/next month output. |
| -s | Displays sunday as the first day of the week. |
| -m | Displays Monday as the first day of the week. |
| -j | Displays Julian dates (days one-based, numbered from January 1). |
| -y | Displays a calendar for the current year. |

#### EXAMPLE:

1. cal

**Output:**

September 2008  
Su Mo Tu We Th Fr Sa  
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

cal command displays the current month calendar.

1. cal -3 5 2008

**Output:**

April 2008  
May 2008  
June 2008  
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa  
1 2 3 4 5  
1 2 3  
1 2 3 4 5 6 7 6 7 8 9 10 11 12  
4 5 6 7 8 9 10  
8 9 10 11 12 13 14 13 14 15 16 17 18 19 11 12 13 14 15 16 17 15 16 17 18 19 20 21 20 21 22 23 24 25 26 18 19 20 21 22 23 24 22 23 24 25 26 27 28 27 28 29 30  
25 26 27 28 29 30 31 29 30

Here the cal command displays the calendar of April, May and June month of year 2008.

#### date COMMAND:

date command prints the date and time.

#### SYNTAX :

date [options][+format][date]

#### EXAMPLE:

date command

date

The above command will print Wed Jul 23 10:52:34 IST 2008

#### cat COMMAND:

cat linux command concatenates files and print it on the standard output.

#### SYNTAX :

cat [OPTIONS] [FILE]...

#### OPTIONS:

|  |  |
| --- | --- |
| **-A** | **Show all.** |
| -b | Omits line numbers for blank space in the output. |
| -e | A $ character will be printed at the end of each line prior to a new line. |
| -E | Displays a $ (dollar sign) at the end of each line. |
| -n | Line numbers for all the output lines. |
| -s | If the output has multiple empty lines it replaces it with one empty line. |
| -T | Displays the tab characters in the output. |
| -v | Non-printing characters (with the exception of tabs, new-lines and form-feeds) are printed visibly. |

#### EXAMPLE:

**1. To Create a new file:**

Cat > file1.txt

This command creates a new file file1.txt. After typing into the file press control+d (^d) simultaneously to end the file.

**2.To Append data into the file:**

Cat >> file1.txt

**3.To display a file:**

Cat file1.txt

This command displays the data in the file.

**4.To concatenate several files and display:**

Cat file1.txt file2.txt

The above cat command will concatenate the two files (file1.txt and file2.txt) and it will display the output in the screen. Sometimes the output may not fit the monitor screen. In such situation you can print those files in a new file or display the file using less command.

catfile1.txtfile2.txt | less

1. **To concatenate several files and to transfer the output to another file.**

Cat file1.txt file2.txt > file3.txt

In the above example the output is redirected to new file file3.txt. The cat command will create new file file3.txt and store the concatenated output into file3.txt.

#### cd COMMAND:

cd command is used to change the directory.

#### SYNTAX :

cd [directory | ~ | ./ | ../ | - ]

#### EXAMPLE:

**1. cd linux-command**  
This command will take you to the sub-directory(linux-command) from its parent directory.

**2.cd ..**This will change to the parent-directory from the current working directory/sub-directory.

**3.cd ~**  
This command will move to the user's home directory which is "/home/username".

#### whoami COMMAND:

whoami command is used to print current user's login name along with effective userid.

#### SYNTAX :

whoami [options]

#### OPTIONS:

|  |  |
| --- | --- |
| **--help** | **Display Help.** |
| --version | Display program version info |

#### EXAMPLE:

Let us consider an example to print effective userid.

1. **whoami**

**Output:**

root

In the above example, the cmd prints the current user's userid as 'root'.

#### clear COMMAND:

This command clears the terminal screen.

#### SYNTAX :

clear

#### OPTIONS:

There is no options for clear screen command.

#### EXAMPLE:

**clear**

clear command clearscreen like cls command.

#### echo COMMAND:

echo command prints the given input string to standard output.

#### SYNTAX :

echo [options..] [string]

#### EXAMPLE:

1. **echo command**

echo "Welcome to India"

The above command will print as Welcome to India

#### mkdir COMMAND:

mkdir command is used to create one or more directories.

#### SYNTAX :

mkdir [options] directories

#### OPTIONS:

|  |  |
| --- | --- |
| **-m** | **Set the access mode for the new directories.** |
| -p | Create intervening parent directories if they don't exist. |
| -v | Print help message for each directory created. |

#### EXAMPLE:

**1. Create directory:**

mkdir test

The above command is used to create the directory 'test'.

**2.Create directory and set permissions:**

mkdir -m 666 test

The above command is used to create the directory 'test' and set the read and write permission.

#### rmdir COMMAND:

rmdir command is used to delete/remove a directory and its subdirectories.

#### SYNTAX :

rmdir [options..] Directory

#### OPTIONS:

|  |  |
| --- | --- |
| **-p** | **Allow users to remove the directory dirname and its parent directories which become empty.** |

#### EXAMPLE:

1. **To delete/remove a directory**

rmdir tmp

rmdir command will remove/delete the directory tmp if the directory is empty.

1. **To delete a directory tree:**

rm –ir tmp

This command recursively removes the contents of all subdirectories of the tmp directory, prompting you regarding the removal of each file, and then removes the tmp directory itself.

#### pwd COMMAND:

pwd - Print Working Directory. pwd command prints the full filename of the current working directory.

#### SYNTAX :

pwd [options]

#### EXAMPLE:

1. **Displays the current working directory.**

pwd

If you are working in home directory then, pwd command displays the current working directory as /home.

#### bc COMMAND:

bc command is used for command line calculator. It is similar to basic calculator. By using which we can do basic mathematical calculations.

#### SYNTAX :

bc [options]

#### EXAMPLE:

1. **bc**

**Output:**

bc 1.06 Copyright 1991-1994,1997,1998,2000 Free Software Foundation,Inc. This is free software with ABSOLUTELY NO WARRANTY. For details type `warranty'. 9\*2 18

The above command used is for mathematical calculations.

1. **bc -l**

**Output:**

bc 1.06 Copyright 1991-1994,1997,1998,2000 Free Software Foundation,Inc. This is free software with ABSOLUTELY NO WARRANTY. For details type `warranty'. 1+2 3

The above command displays the sum of '1+2'.

#### head COMMAND:

head command is used to display the first ten lines of a file, and also specifies how many lines to display.

#### SYNTAX :

head [options] filename

#### OPTIONS:

|  |  |
| --- | --- |
| **-n** | **To specify how many lines you want to display.** |
| -n number | The number option-argument must be a decimal integer whose sign affects the location in the file, measured in lines. |
| -c number | The number option-argument must be a decimal integer whose sign affects the location in the file, measured in bytes. |

#### EXAMPLE:

1. head index.php

This command prints the first 10 lines of 'index.php'.

1. head -5 index.php

The head command displays the first 5 lines of 'index.php'.

1. head -c 5 index.php

The above command displays the first 5 characters of 'index.php'.

#### tail COMMAND:

tail command is used to display the last or bottom part of the file. By default it displays last 10 lines of a file.

#### SYNTAX :

tail [options] filename

#### OPTIONS:

|  |  |
| --- | --- |
| **-l** | **To specify the units of lines.** |
| -b | To specify the units of blocks. |
| -n | To specify how many lines you want to display. |
| -c number | The number option-argument must be a decimal integer whose sign affects the location in the file, measured in bytes. |
| -n number | The number option-argument must be a decimal integer whose sign affects the location in the file, measured in lines. |

#### EXAMPLE:

1. tail index.php

It displays the last 10 lines of 'index.php'.

1. tail -2 index.php

It displays the last 2 lines of 'index.php'.

1. tail -n 5 index.php

It displays the last 5 lines of 'index.php'.

1. tail -c 5 index.php

It displays the last 5 characters of 'index.php'.

#### passwd COMMAND:

passwd command is used to change your password.

#### SYNTAX :

passwd [options]

#### OPTIONS:

|  |  |
| --- | --- |
| **-a** | **Show password attributes for all entries.** |
| -l | Locks password entry for name. |
| -d | Deletes password for name. The login name will not be prompted for password. |
| -f | Force the user to change password at the next login by expiring the password for name. |

#### EXAMPLE:

**passwd**

Entering just passwd would allow you to change the password. After entering passwd you will receive the following three prompts:

**Current Password:  
New Password:  
Confirm New Password:**

Each of these prompts must be entered correctly for the password to be successfully changed.

#### find COMMAND:

find command finds one or more files assuming that you know their approximate filenames.

#### SYNTAX :

find path [options]

#### EXAMPLE:

1. find -name 'cal.txt'

The system would search for any file named 'cal.txt' in the current directory and any subdirectory.

1. find / -name 'cal.txt'

The system would search for any file named 'cal.txt' on the root and all subdirectories from the root.

1. find -name '\*' -size +1000k

The system would search for any file in the list that is larger than 1000k.

#### locate COMMAND:

locate command is used to find the location of files and directories in the system. It is used to find the location of list of files in the particular path, which is specified in the command.

#### SYNTAX :

locate [options] pattern

#### EXAMPLE:

**locate -b 'test'**

**Output:**

/home/uname/Documents/qforquiz/test /home/uname/opt/dev/lampp/htdocs/xampp/test /home/uname/opt/dev/lampp/lib/php/test /home/uname/opt/dev/lampp/lib/php/Math/test

The above command locates and returns the path of files where exactly file name test is stored.

#### chgrp COMMAND:

chgrp command is used to change the group of the file or directory. This is an admin command. Root user only can change the group of the file or directory.

#### SYNTAX :

chgrp [options] newgroup filename/directoryname

#### OPTIONS:

|  |  |
| --- | --- |
| -R | Change the permission on files that are in the subdirectories of the directory that you are currently in. |
| -c | Change the permission for each file. |
| -f | Force. Do not report errors. |

#### EXAMPLE:

1. chgrp hiox test.txt

The group of 'test.txt' file is root, Change to newgroup hiox.

1. chgrp -R hiox test

The group of 'test' directory is root. With -R, the files and its subdirectories also changes to newgroup hiox.

1. chgrp -c hiox calc.txt

They above command is used to change the group for the specific file('calc.txt') only.

#### chmod COMMAND:

chmod command allows you to alter / Change access rights to files and directories.  
**File Permission is given for users,group and others as,**

#### SYNTAX : chmod [options] [MODE] FileName

#### File Permission

|  |  |
| --- | --- |
| **#** | **File Permission** |
| 0 | none |
| 1 | execute only |
| 2 | write only |
| 3 | write and execute |
| 4 | read only |
| 5 | read and execute |
| 6 | read and write |
| 7 | set all permissions |

#### OPTIONS:

|  |  |
| --- | --- |
| **-c** | **Displays names of only those files whose permissions are being changed** |
| -f | Suppress most error messages |
| -R | Change files and directories recursively |
| -v | Output version information and exit. |

#### EXAMPLE:

1. **To view your files with what permission they are:**

ls -alt

This command is used to view your files with what permission they are.

1. **To make a file readable and writable by the group and others.**

chmod 066 file1.txt

1. **To allow everyone to read, write, and execute the file**

chmod 777 file1.txt

#### chown COMMAND:

chown command is used to change the owner / user of the file or directory. This is an admin command, root user only can change the owner of a file or directory.

#### SYNTAX :

chown [options] newowner filename/directoryname

#### OPTIONS:

|  |  |
| --- | --- |
| **-R** | **Change the permission on files that are in the subdirectories of the directory that you are currently in.** |
| -c | Change the permission for each file. |
| -f | Prevents chown from displaying error messages when it is unable to change the ownership of a file. |

#### EXAMPLE:

1. chown hiox test.txt

The owner of the 'test.txt' file is root, Change to new user hiox.

1. chown -R hiox test

The owner of the 'test' directory is root, With -R option the files and subdirectories user also gets changed.

1. chown -c hiox calc.txt

Here change the owner for the specific 'calc.txt' file only.

#### cmp COMMAND:

cmp linux command compares two files and tells you which line numbers are different.

#### SYNTAX :

cmp [options..] file1file2

#### OPTIONS:

|  |  |
| --- | --- |
| **- c** | **Output differing bytes as characters.** |
| - l | Print the byte number (decimal) and the differing byte values (octal) for each difference. |
| - s | Prints nothing for differing files, return exit status only. |

#### EXAMPLE:

1. **Compare two files:**

cmpfile1file2

The above cmp command compares file1.php with file2.php and results as follows.

file1.php file2.php differ: byte 35, line 3

1. **Compare two files output differing bytes as characters:**

cmp –c file1.php file2.php

The above cmp command compares file1.php with file2.php and results as follows.

file1.php file2.php differ: byte 35, line 3 is 151 i 15

#### cp COMMAND:

cp command copy files from one location to another. If the destination is an existing file, then the file is overwritten; if the destination is an existing directory, the file is copied into the directory (the directory is not overwritten).

#### SYNTAX :

cp [OPTIONS]... SOURCE DEST  
cp [OPTIONS]... SOURCE... DIRECTORY  
cp [OPTIONS]... --target-directory=DIRECTORY SOURCE...

#### EXAMPLE:

1. **Copy two files:**

cp file1 file2

The above cp command copies the content of file1.php to file2.php.

1. **To backup the copied file:**

cp –b file1.php file2.php

Backup of file1.php will be created with '~' symbol as file2.php~.

1. **Copy folder and subfolders:**

Cp –r scripts scripts1

The above cp command copy the folder and subfolders from scripts to scripts1.

#### df COMMAND:

df command is used to report how much free disk space is available for each mount you have. The first column show the name of the disk partition as it appears in the /dev directory. Subsequent columns show total space, blocks allocated and blocks available.

#### SYNTAX :

df [options]

#### EXAMPLE:

1. df

**Output:**

Filesystem  
1K-blocks  
Used Available Use% Mounted on /dev/mapper/VolGroup00-LogVol00  
150263916 14440324 128067408 11% / /dev/sda1  
101086  
10896  
84971 12% /boot tmpfs  
253336  
0  
253336  
0% /dev/shm **In the above output: /dev/mapper/VolGroup00-LogVol00 -> Specifies FileSystem. /dev/sda1  
-> Specifies FileSystem. tmpfs  
-> Specifies FileSystem.**

Prints default format.

1. df -h

**Output:**

Filesystem  
Size Used Avail Use% Mounted on /dev/mapper/VolGroup00-LogVol00  
144G  
14G 123G 11% / /dev/sda1  
99M  
11M  
83M 12% /boot tmpfs  
248M  
0 248M  
0% /dev/shm

Print size in human readable format.

1. df -H

**Output:**

Filesystem  
Size  
Used Avail Use% Mounted on /dev/mapper/VolGroup00-LogVol00  
154G  
15G  
132G 11% / /dev/sda1  
104M  
12M  
88M 12% /boot tmpfs  
260M  
0  
260M  
0% /dev/shm

Print size in human readable format but use powers of 1000 not to 1024.

#### du COMMAND:

du command is used to report how much disk space a file or directory occupies.

#### SYNTAX :

du [options] directories

#### EXAMPLE:

1. du -a images

**Output:**

12  
images/daisy.jpg 20  
images/flo.gif 76  
images/CHILD.gif 12  
images/indigo.gif 152  
images/flower.gif 12  
images/sunflower.jpg 12  
images/tulip-flower-clipart5.gif 12  
images/flower.jpg 8  
images/thumbnail.aspx 8  
images/baby.jpg 12  
images/woodpecker.gif 168  
images/baby.gif 8  
images/thumbnail.jpg 1012  
images/house.bmp 12  
images/peacock.gif 1544  
images

Displays the size of each file in the specified directory.

1. du -s images

**Output:**

1544  
images

Displays the total disk space used by the specified directory.

1. du -h

**Output:**

84K

Displays the current folder capacity.

1. du -h file1.php

**Output:**

8.0K file1.php

Displays the storage capacity of file1.php.

#### diff COMMAND:

diff command is used to find differences between two files.

#### SYNTAX :

diff [options..] from-file to-file

#### EXAMPLE:

Lets create two files file1.txt and file2.txt and let it have the following data.

|  |  |
| --- | --- |
| **Data in file1.txt** | **Data in file2.txt** |
| HIOX TEST hscripts.com with friend ship hiox india | HIOX TEST HSCRIPTS.com withfriend ship |

1. **Compare files ignoring white space:**

diff –w file1.txt file2.txt

This command will compare the file file1.txt with file2.txt ignoring white/blank space and it will produce the following output.

#### exit COMMAND:

exit is a command used to terminate a program, shell or log you out of a network normally.

#### SYNTAX :

exit

#### EXAMPLE:

1. **exit**

**Output:**

exit

It terminates from a program, shell or log you out of a network.

#### ln COMMAND:

ln command is used to create link to a file (or) directory. It helps to provide soft link for desired files. Inode will be different for source and destination.

#### SYNTAX :

ln [options] existingfile(or directory)name newfile(or directory)name

#### OPTIONS:

|  |  |
| --- | --- |
| **-f** | **Link files without questioning the user, even if the mode of target forbids writing. This is the default if the standard input is not a terminal.** |
| -n | Does not overwrite existing files. |
| -s | Used to create soft links. |

#### EXAMPLE:

1. ln -s file1.txt file2.txt

Creates a symbolic link to 'file1.txt' with the name of 'file2.txt'. Here inode for 'file1.txt' and 'file2.txt' will be different.

1. ln -s nimi nimi1

Creates a symbolic link to 'nimi' with the name of 'nimi1'.

#### ls COMMAND:

ls command lists the files and directories under current working directory.

#### SYNTAX :

ls [OPTIONS]... [FILE]

|  |  |
| --- | --- |
| **-l** | **Lists all the files, directories and their mode, Number of links, owner of the file, file size, Modified date and time and filename.** |
| **-a** | **Lists all entries including hidden files.** |

1. **Display root directory contents:**

ls /

lists the contents of root directory.

1. **Display hidden files and directories:**

ls -a

lists all entries including hidden files and directories.

1. **Display inode information:**

ls -i

7373073 book.gif  
7373074 clock.gif 7373082 globe.gif 7373078 pencil.gif 7373080 child.gif 7373081 email.gif 7373076 indigo.gif

The above command displays filename with inode value.

1. **WC COMMAND:**

The **wc** (**word count**) command in Unix/Linux operating systems is used to find out number of **newline count**, **word count**, **byte and characters** count in a files specified by the file arguments. The syntax of **wc** command as shown below.

**Syntax:**

# wc [options] filenames

**Example:**

**wc -l** : Prints the number of lines in a file.

**wc -w** : prints the number of words in a file.

**wc -c** : Displays the count of bytes in a file.

**wc -m** : prints the count of characters from a file.

**wc -L** : prints only the length of the longest line in a file.

**Example:**

[root@tecmint ~]# cat tecmint.txt

Red Hat

CentOS

Fedora

Debian

Scientific Linux

OpenSuse

Ubuntu

Xubuntu

Linux Mint

Pearl Linux

Slackware

Mandriva

The ‘**wc**‘ command without passing any parameter will display a basic result of ”**tecmint.txt**‘ file. The three numbers shown below are **12** (**number of lines**), **16** (**number of words**) and **112** (**number of bytes**) of the file.

root@tecmint ~]# wc tecmint.txt

**12** **16** **112** tecmint.txt

To count number of newlines in a file use the option ‘**-l**‘, which prints the number of lines from a given file

[root@tecmint ~]# wc -l tecmint.txt

**12** tecmint.txt

Using ‘**-w**‘ argument with ‘**wc**‘ command prints the number of words in a file.

[root@tecmint ~]# wc -w tecmint.txt

**16** tecmint.txt

When using options ‘**-c**‘ and ‘**-m**‘ with ‘**wc**‘ command will print the total **number of bytes** and **characters** respectively in a file.

[root@tecmint ~]# wc -c tecmint.txt

**112** tecmint.txt

[root@tecmint ~]# wc -m tecmint.txt

**112** tecmint.txt

**LINUX USERS AND GROUP:**

* Linux is a multi-user OS that is based on the Unix concepts of file ownership and permissions to provide security, at the file system level.

**About Users:**

* In Linux, there are two types of users: **system users and regular users.**
* Traditionally, system users are used to run non-interactive or background processes on a system, while regular users used for logging in and running processes interactively.
* When you first log in to a Linux system, you may notice that it starts out with many system users that run the services that the OS depends on--this is completely normal.
* An easy way to view all of the users on a system is to look at the contents of the /etc/passwd file. Each line in this file contains information about a single user, starting with its user name (the name before the first :). Print the passwd file with this command:

**cat /etc/passwd**

### Superuser:

* In addition to the two user types, there is the **superuser, or root user**, that has the ability to override any file ownership and permission restrictions.
* This means that the superuser has the rights to access anything on its own server. This user is used to make system-wide changes, and must be kept secure.
* It is also possible to configure other user accounts with the ability to assume "superuser rights".

### Creating and Deleting User Accounts:

* To create a new standard user, use the useradd command. The syntax is as follows:

**Useradd <name>**

* You will need to set a password for the new user by using the passwd command. Note, you will need root privileges to change a user password. The syntax is as follows:

**passwd <username>**

* The user will be able to change their password at any time using the passwd command with the syntax. Below is an example:

$ passwd

Changing password for Martin.

(current) UNIX password:

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

* To remove a user account, enter the following command:

**userdel <name>**

## To remove the user, their home folder, and their files, use this command:

## Userdel –r <name>

## About Groups:

* Groups are collections of zero or more users.
* A user belongs to a default group, and can also be a member of any of the other groups on a server.
* An easy way to view all the groups and their members is to look in the /etc/group file on a server. We won't cover group management in this article, but you can run this command if you are curious about your groups:

**cat /etc/group**

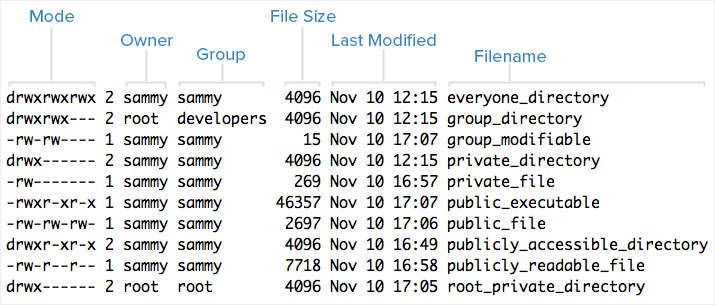
* All Linux users have a user ID and a group ID and a unique numerical identification number called a userid (UID) and a groupid (GID) respectively. Groups can be assigned to logically tie users together for a common security, privilege and access purpose.
* It is the foundation of Linux security and access.
* Files and devices may be granted access based on a users ID or group ID.
* File, directory and device (special file) permissions are granted based on “user”, “group” or “other” identification status.
* Permission is granted (or denied) for read, write and execute access.

**PERMISSION FOR FILE, DIRECTORY AND USERS:**

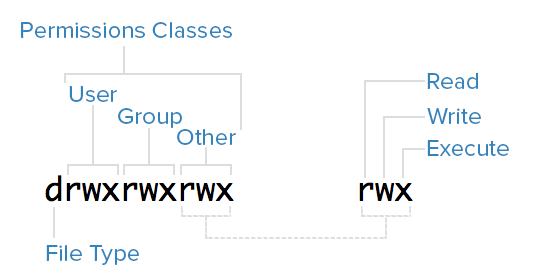
**Viewing Ownership and Permissions:**

* In Linux, each and every file is owned by a single user and a single group, and has its own access permissions.
* The most common way to view the permissions of a file is to use **ls** with the long listing option, **e.g. ls -l myfile.**
* If you want to view the permissions of all of the files in your current directory, run the command without an argument, like this:

**ls -l**



**Understanding Mode:**



### File Type:

* In Linux, there are two basic types of files: normal and special. The file type is indicated by the first character of the mode of a file -- , we refer to this as the file type field.
* Normal files can be identified by files with a hyphen (-) in their file type fields. Normal files are just plain files that can contain data.
* They are called normal, or regular, files to distinguish them from special files.
* Special files can be identified by files that have a non-hyphen character, such as a letter, in their file type fields, and are handled by the OS differently than normal files.
* The character that appears in the file type field indicates the kind of special file a particular file is. For example, a directory, which is the most common kind of special file, is identified by the d character that appears in its file type field.

**Permissions Classes:**

* From the diagram, we know that Mode column indicates the file type, followed by three classes, of permissions: user (owner), group, and other.
* The order of the classes is consistent across all Linux distributions.
* Let's look at which users belong to each permissions class:
* **User**: The owner of a file belongs to this class
* **Group**: The members of the file's group belong to this class
* **Other**: Any users that are not part of the user or group classes belong to this class.

### Reading Symbolic Permissions:

* The next thing to pay attention to are the sets of three characters, as they denote the permissions, in symbolic form, that each class has for a given file.
* In each triad read, write, and execute permissions are represented in the following way:
* **Read**: Indicated by an r in the first position.
* **Write**: Indicated by a w in the second position.
* **Execute**: Indicated by an x in the third position. In some special cases, there may be a different character here.
* A hyphen (-) in the place of one of these characters indicates that the respective permission is not available for the respective class.
* For example, if the group triad for a file is r--, the file is "read-only" to the group that is associated with the file.

## Understanding Read, Write, Execute:

### Read:

* For a normal file, read permission allows a user to view the contents of the file.
* For a directory, read permission allows a user to view the names of the file in the directory.

### Write:

* For a normal file, write permission allows a user to modify and delete the file.
* For a directory, write permission allows a user to delete the directory, modify its contents (create, delete, and rename files in it), and modify the contents of files that the user can read.

### Execute:

* For a normal file, execute permission allows a user to execute a file (the user must also have read permission). As such, execute permissions must be set for executable programs and shell scripts before a user can run them.
* For a directory, execute permission allows a user to access, or traverse, into (i.e. cd) and access metadata about files in the directory (the information that is listed in an ls -l).

## Examples of Modes (and Permissions):

* Now that know how to read the mode of a file, and understand the meaning of each permission, we will present a few examples of common modes, with brief explanations, to bring the concepts together.
* -rw-------: A file that is only accessible by its owner
* -rwxr-xr-x: A file that is executable by every user on the system. A "world-executable" file
* -rw-rw-rw-: A file that is open to modification by every user on the system. A "world-writable" file
* drwxr-xr-x: A directory that every user on the system can read and access
* drwxrwx---: A directory that is modifiable (including its contents) by its owner and group
* drwxr-x---: A directory that is accessible by its group

**Set file, directory and device permissions:**

**Chmod Command:**

* File, directory and device permissions can be set to allow or deny access to members of their own group or all others.
* Modification of file, directory and device access is achieved with the chmod command.
* The permissions can be assigned in octal notation or in the more easily recognized character format where the command form is:

**chmod [ugoa][+-=][rwxXst] *file or directoryName***

|  |  |
| --- | --- |
| u | User access |
| g | Group access |
| o | Other system user's access |
| a | Equivilent to "ugo" |

|  |  |
| --- | --- |
| + | Add access |
| - | Remove access |
| = | Access explicitly assigned |

|  |  |
| --- | --- |
| r | Permission to read a file Permission to read a directory (also requires "x") |
| w | Permission to delete or modify a file Permission to delete or modify files in a directory |
| x | Permission to execute a file/script Permission to read a directory (also requires "r") |
| s | Set user or group ID on execution. |
| u | Permissions granted to the user who owns the file |
| t | Set "sticky bit. Execute file/script as user root for regular user. |

**Examples:**

* Grant read access (r) to a file to all members of your group (g):   
  [**chmod**](http://man.yolinux.com/cgi-bin/man2html?cgi_command=chmod)**g+r *file-name***
* Grant read access to a directory to all members your group:   
  **chmod g+rx *directory-name***Note that "execute" permission is required in order to read a directory.
* Grant read permissions to everyone on the system to a file which you own so that everyone may read it: **(u)ser, (g)roup and (o)ther.   
  chmod ugo+r *file-name***
* Grant read permissions on a directory to everyone on the system:   
  **chmod ugo+rx *directory-name***
* Grant modify or delete permissions to a file which you own for everyone in the group:   
  **chmod ugo+rw *file-name***   
  Note: In order for modify and delete permissions to be useful, one must be able to modify the directory in which the file is located: **chmod ugo+rwx *./***
* Deny read access to a file by everyone except yourself:   
  **chmod go-r *file-name***
* Allow everyone in your group to be able to modify the file:   
  **chmod 660 *file-name***

### Chmod Octal Format:

* To use the octal format, you have to calculate the permissions for each portion of the file or directory.
* The execute permission is equal to the number one (1), the write permission is equal to the number two (2), and the read permission is equal to the number four (4).
* Therefore, when you use the octal format, you will need to calculate a number between 0 and 7 for each portion of the permission. A table has been provided below for clarification.

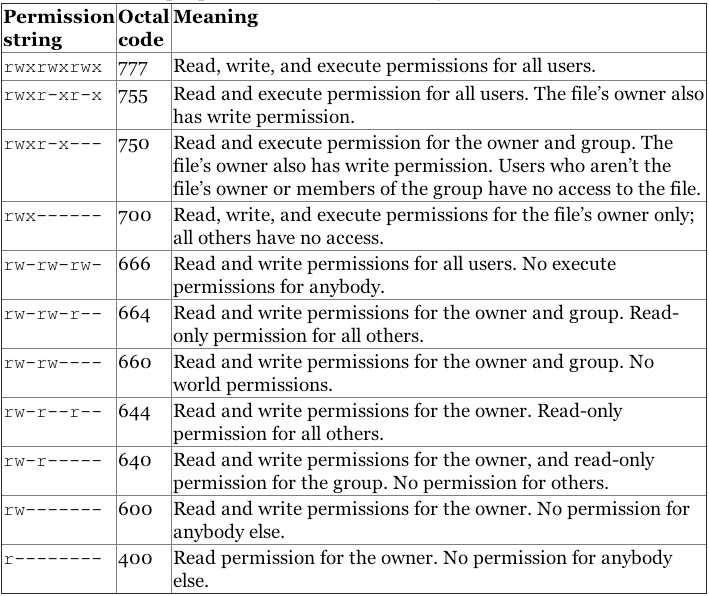


**Octal format:**  chmod 444 work

* The output of ls -al after the chmod command above would look as follows:

**dr - - r - - r - - 2 user user 4096 Dec 17 14:38 work**

* An octal table showing the numeric equivalent for permissions is provided below.



### Changing File Ownership:

* By default, all files are “owned” by the user who creates them and by that user’s default group.
* To change the ownership of a file, use the chown command in the  chown user:group /path/to/file format.
* In the following example, the ownership of the “list.html” file will be changed to the “ramesh” user in the “marketing” group:

**chown ramesh:marketing list.html**

* To change the ownership of a directory and all the files contained inside, use the recursive option with the –R flag.
* In the following example, change the ownership of /srv/smb/leadership/ to the “ramesh” user in the “marketing” group:

**chown –R ramesh:marketing /srv/smb/leadership/**

**SEARCHING A FILE AND DIRECTORY:**

* There may be times when you know a file or directory exists but you do not know where to find it. There are several commands you can use to search for it, including find, locate, and which.

## Find:

* The format of the find command is:

**find path pattern**

* If you do not specify a path, find starts in the current working directory and looks through all subdirectories for the specified pattern.

## Finding by Name:

* The most obvious way of searching for files is by name.
* To find a file by name, type:

**find -name "query"**

* This will be case sensitive, meaning a search for "file" is different than a search for "File".
* To find a file by name, but ignore the case of the query, type:

**find -iname "query"**

## Finding by Type:

* You can specify the type of files you want to find with the "-type" parameter. It works like this:

**find -type type\_descriptor query**

* Some of the most common descriptors that you can use to specify the type of file are here:
* **f**: regular file
* **d**: directory
* **l**: symbolic link
* **c**: character devices
* **b**: block devices
* For instance, if we wanted to find all of the character devices on our system, we could issue this command:

find / -type c

/dev/parport0

/dev/snd/seq

/dev/snd/timer

/dev/autofs

/dev/cpu/microcode

/dev/vcsa7

/dev/vcs7

/dev/vcsa6

/dev/vcs6

/dev/vcsa5

/dev/vcs5

/dev/vcsa4

. . .

* We can search for all files that end in ".conf" like this:

find / -type f -name "\*.conf"

/var/lib/ucf/cache/:etc:rsyslog.d:50-default.conf

/usr/share/base-files/nsswitch.conf

/usr/share/initramfs-tools/event-driven/upstart-jobs/mountall.conf

/usr/share/rsyslog/50-default.conf

/usr/share/adduser/adduser.conf

/usr/share/davfs2/davfs2.conf

/usr/share/debconf/debconf.conf

/usr/share/doc/apt-utils/examples/apt-ftparchive.conf

. . .

## Filtering by Time and Size:

* Find gives you a variety of ways to filter results by size and time.

### Size:

* You can filter by size with the use of the "-size" parameter.
* We add a suffix on the end of our value that specifies how we are counting. These are some popular options:
* **c**: bytes
* **k**: Kilobytes
* **M**: Megabytes
* **G**: Gigabytes
* **b**: 512-byte blocks
* To find all files that are exactly 50 bytes, type:

**find / -size 50c**

* To find all files less than 50 bytes, we can use this form instead:

**find / -size -50c**

* To Find all files more than 700 Megabytes, we can use this command:

**find / -size +700M**

### Time:

* Linux stores time data about access times, modification times, and change times.
* **Access Time**: Last time a file was read or written to.
* **Modification Time**: Last time the contents of the file were modified.
* **Change Time**: Last time the file's inode meta-data was changed.
* We can use these with the "-atime", "-mtime", and "-ctime" parameters. These can use the plus and minus symbols to specify greater than or less than, like we did with size.

## Finding by Owner and Permissions:

* You can also search for files by the file owner or group owner.

**find** **/** -user pat -iname "filename"

**find** **/** -group **users** -iname "filename"

**find** **/** -perm 777 -iname "filename"

* You do this by using the "-user" and "-group" parameters respectively. Find a file that is owned by the "syslog" user by entering:

**find / -user syslog**

* Similarly, we can specify files owned by the "shadow" group by typing:

**find / -group shadow**

* We can also search for files with specific permissions.
* If we want to match an exact set of permissions, we use this form:

**find / -perm 644**

* This will match files with exactly the permissions specified.
* If we want to specify anything with at least those permissions, you can use this form:

**find / -perm -644**

* This will match any files that have additional permissions. A file with permissions of "744" would be matched in this instance.

## Locate:

* The format of the locate command is:

**locate pattern**

* With locate, you can see every file or directory whose name contains the search criterion. For example, to search for all files with the word finger in the name, type:

**locate finger**

* The locate command uses a database to locate files and directories that have the word finger in the file or directory name.
* The search results could include a file called finger.txt, a file called pointerfinger.txt, a directory named /fingerthumbnails/, and so on.
* To find files with locate, simply use this syntax:

**locate query**

* You can filter the output in some ways.
* For instance, to only return files containing the query itself, instead of returning every file that has the query in the directories leading to it, you can use the "-b" for only searching the "basename":

**locate -b query**

* To have locate only return results that still exist in the filesystem, use the "-e" flag:

**locate -e query**

* To see statistics about the information that locate has cataloged, use the "-S" option:

**locate -S**

Database /var/lib/mlocate/mlocate.db:

3,315 directories

37,228 files

1,504,439 bytes in file names

594,851 bytes used to store database

**ZIPPING AND UNZIPPING CONCEPTS:**

* Zip utility is used to combine as well as compress files in Linux.
* Zip utility is to compress or you can say zip a file or files or a directory but what matters is it's compression ratio.
* It can be used by different compression level and each level has its own benefit and requirement which depends up on client’s requirement.

## How to zip a file using zip utility in Linux?

* So now if you want to zip a file in Linux, you must have some files to zip.Here i have created three files satish.txt, sarath.txt and slashroot.txt just for testing purpose.
* Let's have a look on the size of these files.

**[root@localhost test]# ls -lh**

**total 196K**

**-rw-r--r-- 1 root root  12K Apr 19 05:26 sarath.txt**

**-rw-r--r-- 1 root root 113K Apr 19 05:26 satish.txt**

**-rw-r--r-- 1 root root  57K Apr 19 05:26 slashroot.txt**

* We can see the file sizes of sarath.txt, satish.txt and slashroot.txt  are 12K,113K and 57K respectively.
* Now zip satish.txt file here.

**[root@localhost test]# zip satish.zip satish.txt**

**adding: satish.txt (deflated 98%)**

**[root@localhost test]# ls -lh**

**total 200K**

**-rw-r--r-- 1 root root  12K Apr 19 05:26 sarath.txt**

**-rw-r--r-- 1 root root 113K Apr 19 05:26 satish.txt**

**-rw-r--r-- 1 root root 2.1K Apr 19 05:27 satish.zip**

**-rw-r--r-- 1 root root  57K Apr 19 05:26 slashroot.txt**

* Now you can see the size of zip file is less than the size of original file. i.e satish.txt file.
* The size of satish.txt was 113 KB while after zipping it's become 2.1KB only. It means 98% deflated. And hence you can see that by using zip command you have saved 98% of your Disk space here.
* In this way you can keep approximately 2TB Data in a Hard Disk of size 1 TB.

**Now how to unzip zip file back to original?**

* First i am going to remove the satish.txt file and then i will get it back by unziping satish.zip file using unzip command.

**[root@localhost test]# rm -rf satish.txt**

**[root@localhost test]# unzip satish.zip**

**Archive:  satish.zip**

**inflating: satish.txt**

**[root@localhost test]#**

## How to zip multiple files and keep them inside a single file?

* I have three files satish.txt, sarath.txt and slashroot.txt and i am going to zip them all at a time and archieve them inside a single file named allfiles.zip.

**[root@localhost test]# zip allfiles.zip satish.txt sarath.txt slashroot.txt**

**adding: satish.txt (deflated 98%)**

**adding: sarath.txt (deflated 97%)**

**adding: slashroot.txt (deflated 98%)**

**[root@localhost test]#**

* Now to see whether files are zipped or not, or if zipped how much deflated type the below command.

**[root@localhost test]# ls -lh**

**total 200K**

**-rw-r--r-- 1 root root 3.7K Apr 19 06:01 allfiles.zip**

**-rw-r--r-- 1 root root  12K Apr 19 05:26 sarath.txt**

**-rw-r--r-- 1 root root 113K Apr 19 05:26 satish.txt**

**-rw-r--r-- 1 root root  57K Apr 19 05:26 slashroot.txt**

* Here you can compare the size easily and hence calculate how much compression is done by using simple mathematical calculation.

**How to unzip it you can simply use the unzip command:**

* Let remove the files first so that it does not ask for overwrite option.

**[root@localhost test]# rm -rf sarath.txt satish.txt slashroot.txt**

* Now you can see we have only one zip file. so i am going to unzip it using unzip command.

**[root@localhost test]# unzip allfiles.zip**

**Archive:  allfiles.zip**

**inflating: satish.txt**

**inflating: sarath.txt**

**inflating: slashroot.txt**

### How to Encrypt a zip file in Linux?

* For testing encryption decryption technology for zip files i have created a plain text file linux.txt.
* Now i will first zip it and then i will encrypt the zip file.

**[root@localhost test]# zip linux.zip linux.txt**

**adding: linux.txt (deflated 9%)**

### Encrypting linux.zip file through gpg command:

**[root@localhost test]# gpg -c linux.zip**

**gpg: directory `/root/.gnupg' created**

**gpg: new configuration file `/root/.gnupg/gpg.conf' created**

**gpg: WARNING: options in `/root/.gnupg/gpg.conf' are not yet active during this run**

**gpg: keyring `/root/.gnupg/pubring.gpg' created**

**passphrase:**

* gpg utility has been used here to encrypt the zip file.
* -c option is used to create gpg based encrypted file.

**How to give password to a ZIP file without encryption?**

* We have protected the file with the help of encryption done by GPG command. Now if you dont want to encrypt your file but want to just give a minimal amount of security, then you can do that by locking your zip file with a password.
* This can be done by -P option in the zip command. Let’s see an example.

**[root@localhost test]# zip -P redhat123 linux.zip linux.txt**

**adding: linux.txt (deflated 9%)**

* In above command redhat123 is the password.
* Linux.zip is password protected file now.
* Linux.txt is the file on which we have implemented this security.
* Now if you want to unzip linux.zip you must know the password.

**[root@localhost test]# unzip linux.zip**

**Archive:  linux.zip**

**[linux.zip] linux.txt password:**