Table of Content

1. Permissions
2. Process Management
3. User & Group management
4. Package Management
5. Input Output Redirection in Linux
6. Linux networking Introduction
7. Unix / Linux - What is Shells?

Permissions

Execute “ls” with the “long listing” option, you would type ls -l

When you do so, each file will be listed on a separate line in long format. There is an example in the window below.

The first character will almost always be either a ‘-‘, which means it’s a file, or a ‘d’, which means it’s a directory.

The next nine characters (rw-r–r–) show the security; we’ll talk about them later.

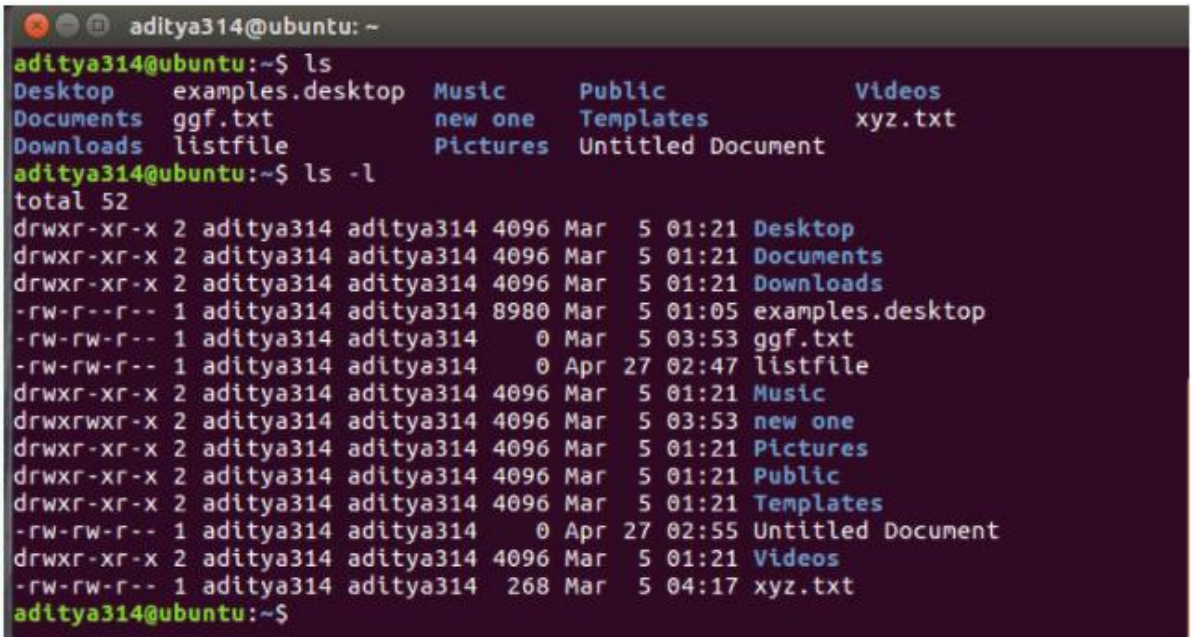
The next column shows the owner of the file. In this case it is me, my userID is “aditya314”.

The next column shows the group owner of the file. In my case I want to give the “aditya314” group of people special access to these files.

The next column shows the size of the file in bytes.

The next column shows the date and time the file was last modified.

And, of course, the final column gives the filename.



Important commands:

chmod - modify file access rights

chown - change file ownership

chgrp - change a file's group ownership

Understanding the security permissions

First, you must think of those nine characters as three sets of three characters (see the box at the bottom). Each of the three “rwx” characters refers to a different operation you can perform on the file.

--- --- ---

rwx rwx rwx

user group other

Read, write, execute and –

The ‘r’ means you can “read” the file’s contents.

The ‘w’ means you can “write”, or modify, the file’s contents.

The ‘x’ means you can “execute” the file. This permission is given only if the file is a program.

If any of the “rwx” characters is replaced by a ‘-‘, then that permission has been revoked.

User, group and others

user – The user permissions apply only the owner of the file or directory, they will not impact the actions of other users.

group – The group permissions apply only to the group that has been assigned to the file or directory, they will not effect the actions of other users.

others – The others permissions apply to all other users on the system, this is the permission group that you want to watch the most.

File Permissions

On a Linux system, each file and directory is assigned access rights for the owner of the file, the members of a group of related users, and everybody else. Rights can be assigned to read a file, to write a file, and to execute a file (i.e., run the file as a program).

To see the permission settings for a file, we can use the ls command. As an example, we will look at the bash program which is located in the /bin directory:

[me@linuxbox me]$ ls -l /bin/bash

-rwxr-xr-x 1 root root 1113504 Jun 6 2019 /bin/bash

Here we can see:

The file "/bin/bash" is owned by user "root"

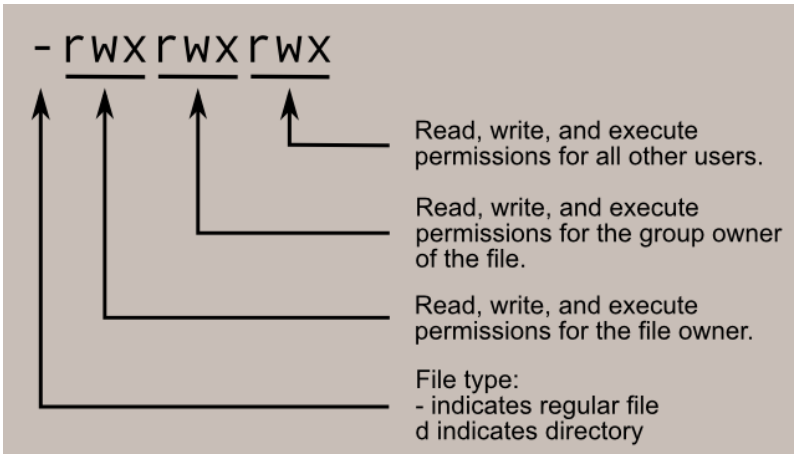
The superuser has the right to read, write, and execute this file

The file is owned by the group "root"

Members of the group "root" can also read and execute this file

Everybody else can read and execute this file

In the diagram below, we see how the first portion of the listing is interpreted. It consists of a character indicating the file type, followed by three sets of three characters that convey the reading, writing and execution permission for the owner, group, and everybody else.



chmod

The chmod command is used to change the permissions of a file or directory. To use it, we specify the desired permission settings and the file or files that we wish to modify. There are two ways to specify the permissions. In this lesson we will focus on one of these, called the octal notation method.

It is easy to think of the permission settings as a series of bits (which is how the computer thinks about them). Here's how it works:

rwx rwx rwx = 111 111 111

rw- rw- rw- = 110 110 110

rwx --- --- = 111 000 000

and so on...

rwx = 111 in binary = 7

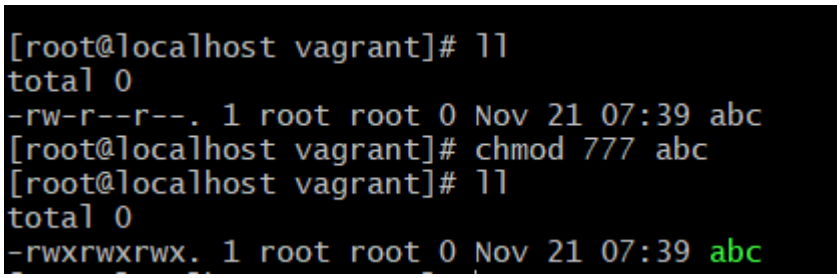
rw- = 110 in binary = 6

r-x = 101 in binary = 5

r-- = 100 in binary = 4

Now, if we represent each of the three sets of permissions (owner, group, and other) as a single digit, we have a pretty convenient way of expressing the possible permissions settings. For example, if we wanted to set some_file to have read ,write and execute permission for the owner, but wanted to keep the file private from others, we would:

Example: We are taking abc and giving full permissions (R+W+X)



Here is a table of numbers that covers all the common settings. The ones beginning with "7" are used with programs (since they enable execution) and the rest are for other kinds of files.

Value	Meaning
777	**(rwxrwxrwx)** No restrictions on permissions. Anybody may do anything. Generally not a desirable setting.
755	**(rwxr-xr-x)** The file's owner may read, write, and execute the file. All others may read and execute the file. This setting is common for programs that are used by all users.
700	**(rwx------)** The file's owner may read, write, and execute the file. Nobody else has any rights. This setting is useful for programs that only the owner may use and must be kept private from others.
666	**(rw-rw-rw-)** All users may read and write the file.
644	**(rw-r--r--)** The owner may read and write a file, while all others may only read the file. A common setting for data files that everybody may read, but only the owner may change.
600	**(rw-------)** The owner may read and write a file. All others have no rights. A common setting for data files that the owner wants to keep private.

Directory Permissions

The chmod command can also be used to control the access permissions for directories. Again, we can use the octal notation to set permissions, but the meaning of the r, w, and x attributes is different:

r - Allows the contents of the directory to be listed if the x attribute is also set.

w - Allows files within the directory to be created, deleted, or renamed if the x attribute is also set.

x - Allows a directory to be entered (i.e. cd dir).

Here are some useful settings for directories:

Value	Meaning
777	**(rwxrwxrwx)** No restrictions on permissions. Anybody may list files, create new files in the directory and delete files in the directory. Generally not a good setting.
755	**(rwxr-xr-x)** The directory owner has full access. All others may list the directory, but cannot create files nor delete them. This setting is common for directories that you wish to share with other users.
700	**(rwx------)** The directory owner has full access. Nobody else has any rights. This setting is useful for directories that only the owner may use and must be kept private from others.

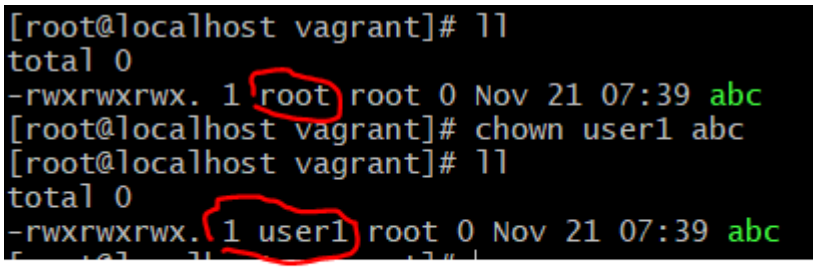
Changing File Ownership :

We can change the owner of a file by using the chown command. Here's an example: Suppose we wanted to change the owner of some_file from "me" to "you". We could:

[me@linuxbox me]$ sudo chown you some_file

Notice that in order to change the owner of a file, we must have superuser privileges. To do this, our example employed the sudo command to execute chown.

Example:



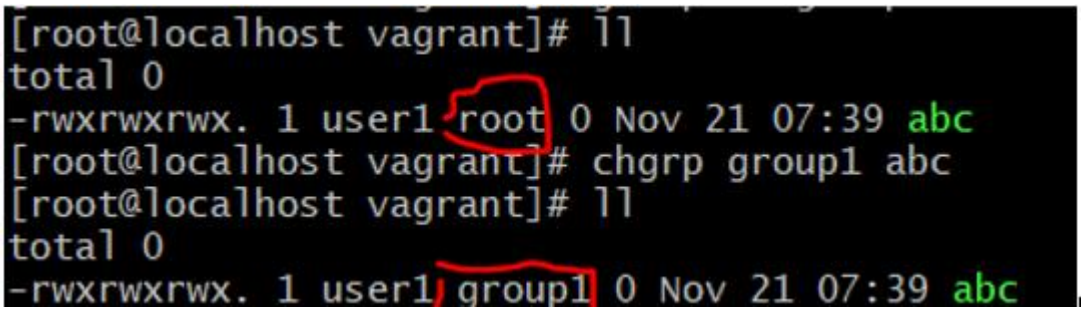
chown works the same way on directories as it does on files.

Changing Group Ownership

The group ownership of a file or directory may be changed with chgrp. This command is used like this:

[me@linuxbox me]$ chgrp new_group some_file

Example: Here we are changing group from **root** to **group1** of abc file



Process Management

Whenever you issue a command in Unix, it creates, or starts, a new process. When you tried out the **ls** command to list the directory contents, you started a process.

A process, in simple terms, is an instance of a running program.

The operating system tracks processes through a five-digit ID number known as the **pid** or the **process ID**. Each process in the system has a unique **pid**.

Pids eventually repeat because all the possible numbers are used up and the next pid rolls or starts over. At any point of time, no two processes with the same pid exist in the system because it is the pid that Unix uses to track each process.

- **Foreground Processes**: They run on the screen and need input from the user. For example Office Programs
- **Background Processes**: They run in the background and usually do not need user input. For example, Antivirus.

Process States in Linux

A process in Linux can go through different states after it's created and before it's terminated. These states are:

Running: A process in **running** state means that it is running or it's ready to run.

Sleeping

The process is in a **sleeping** state when it is waiting for a resource to be available.

- **Interruptible sleep**
- **Uninterruptible sleep**

A process in **Interruptible sleep** will wake up to handle signals, whereas a process in **Uninterruptible sleep** will not.

Stopped: A process enters a **stopped** state when it receives a stop signal.

Zombie: **Zombie** state is when a process is dead but the entry for the process is still present in the table.

Listing Running Processes

It is easy to see your own processes by running the ps (process status) command as follows −

```
$ps
PID   TTY      TIME     CMD
18358  ttyp3   00:00:00  sh
18361  ttyp3   00:01:31  abiword
18789  ttyp3   00:00:00  ps
```

One of the most commonly used flags for ps is the -f (f for full) option, which provides more information as shown in the following example −

```
$ps -f
UID    PID  PPID C STIME   TTY   TIME CMD
amrood  6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood  6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood  3662 3657 0 08:10:53 pts/6 0:00 -ksh
amrood  6892 3662 4 10:51:50 pts/6 0:00 ps -f
```

The terminology is as follows :

PID process ID

TTY terminal type

TIME total time the process has been running

CMD name of the command that launches the process

C cpu utilization of process

STIME Process start time

PPID Parent Process ID

$ ps -u

```
root@localhost:~# ps -u
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         645  0.0  0.1   8200  2236 ttyS0    Ss+  May19   0:00 /sbin/agetty
root         649  0.0  0.0   8428  1752 tty1     Ss+  May19   0:00 /sbin/agetty
root       98066  0.8  0.4  15892  8764 pts/0    Ss   17:25   0:00 -bash
root       98175  0.0  0.1  13200  3340 pts/0    R+   17:26   0:00 ps -u
```

Here: STAT represents process state.

%CPU represents the amount of computing power the process is taking.

%MEM represents the amount of memory the process is taking up.

Stopping Processes

Ending a process can be done in several different ways. Often, from a console-based command, sending a CTRL + C keystroke (the default interrupt character) will exit the command. This works when the process is running in the foreground mode.

If a process is running in the background, you should get its Job ID using the ps command. After that, you can use the kill command to kill the process as follows −

$ps -f

UID PID PPID C STIME TTY TIME CMD

amrood 6738 3662 0 10:23:03 pts/6 0:00 first_one

amrood 6739 3662 0 10:22:54 pts/6 0:00 second_one

amrood 3662 3657 0 08:10:53 pts/6 0:00 -ksh

amrood 6892 3662 4 10:51:50 pts/6 0:00 ps –f

$kill 6738

Terminated

Here, the kill command terminates the first_one process. If a process ignores a regular kill command, you can use kill -9 followed by the process ID as follows −

$kill -9 6738

Terminated

Parent and Child Processes

Each unix process has two ID numbers assigned to it: The Process ID (pid) and the Parent process ID (ppid). Each user process in the system has a parent process.

Most of the commands that you run have the shell as their parent. Check the **ps -f** example where this command listed both the process ID and the parent process ID.

Zombie and Orphan Processes

Normally, when a child process is killed, the parent process is updated via a **SIGCHLD** signal. Then the parent can do some other task or restart a new child as needed. However, sometimes the parent process is killed before its child is killed. In this case, the "parent of all processes," the **init** process, becomes the new PPID (parent process ID). In some cases, these processes are called orphan processes.

When a process is killed, a **ps** listing may still show the process with a **Z** state. This is a zombie or defunct process. The process is dead and not being used. These processes are different from the orphan processes. They have completed execution but still find an entry in the process table.

Daemon Processes

Daemons are **system-related background processe**s that often run with the permissions of root and services requests from other processes.

A daemon has no controlling terminal. It cannot open **/dev/tty**. If you do a **"ps -ef"** and look at the **tty** field, all daemons will have a **?** for the **tty**.

To be precise, a daemon is a process that runs in the background, usually waiting for something to happen that it is capable of working with. For example, a printer daemon waiting for print commands.

If you have a program that calls for lengthy processing, then it’s worth to make it a daemon and run it in the background.

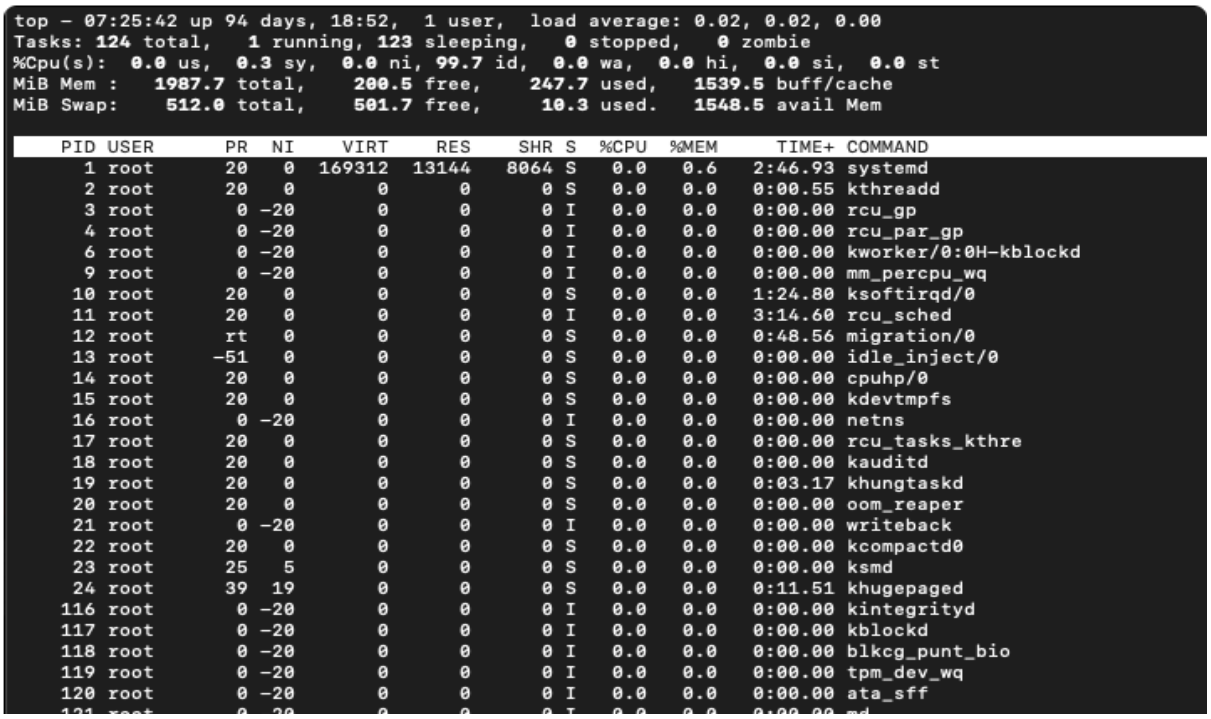
The top Command

The top command is a very useful tool for quickly showing processes sorted by various criteria.

It is an interactive diagnostic tool that updates frequently and shows information about physical and virtual memory, CPU usage, load averages, and your busy processes.

Here is the simple syntax to run top command and to see the statistics of CPU utilization by different processes −

$top



Sleep

The sleep command pauses for a specified time. It's generally used in a script, but works on the command line as well. In the example below, sleep pauses a minute between the two date commands.

$ date; sleep 60; date

Wed Sep 8 12:10:40 PM EDT 2021

Wed Sep 8 12:11:40 PM EDT 2021

wc

The wc command in Linux expands to 'word count'. It is used to display the number of lines, words, characters, and bytes corresponding to any file mentioned in the arguments.

Syntax

$wc [Option].. [File]..

Here are a few examples to help you understand this concept better. Let us assume we have a sample text file with the following contents

cd ..: This command is used to navigate to the parent directory of the current directory (here, Desktop is the parent directory of the Linux folder)

$ cd ..

echo

The echo command in Linux simply displays a line of text/string which is passed in as an argument. It is commonly used for debugging shell programs inside the terminal.

Syntax

$echo [Option] [String]

Here are a few simple examples to help you understand this command better.

echo "String": Used to display the string passed in as an argument inside quotes.

$ echo "String"

echo -e "Learn \nBy \nDoing": The '-e' tag enables the echo command to recognize the backslash escape sequences inside the argument.

$ echo -e "Learn \nBy \nDoing"

history

The history command in Linux is used to view a history of all the commands previously executed inside the bash terminal. The total number of executed commands will vary from one system to another.

Syntax

$ history

Here's an example to help you understand this command.

$ history

Cal:

Print calender

Date:

Print date

Time:

Print time

clear Clear a command line screen/window for a fresh start.

who [options] Display who is logged on.

User & Group management

Since Linux is a multi-user operating system, several people may be logged in and actively working on a given machine at the same time. Security-wise, it is never a good idea to allow users to share the credentials of the same account. In fact, best practices dictate the use of as many user accounts as people needing access to the machine.

At the same time, it is to be expected that two or more users may need to share access to certain system resources, such as directories and files. User and group management in Linux allows us to accomplish both objectives.

Linux systems have two types of users:

1) general/normal user
2) root/super user .

While general users have limited access to the Linux system, root users have access to anything & everything on the Linux system.

When a user is created a group with the same user name is also created. Every user has its own home directory, for user root its /root & for general users its located in /home/.

These operations are performed using the following commands:

Useradd : Adds accounts to the system

Usermod : Modifies account attributes

Userdel : Deletes accounts from the system

USER Management

Below mentioned are the commands that are used for user management,

Purpose	*Command*
• Adding a user	useradd dan
• Assigning password to user	passwd dan
• Changing home directory for user	useradd dan –d /home/new
• Setting expiry for user	useradd dan –e 2017-11-25
• Addding inactive period before expiry	useradd dan –f 2
• Changing default shell	useradd dan –s /bin/sh
• Removing user	userdel dan
• Removing user with home directory	userdel –r dan

We can also modify default settings of a user after it has bee added with ***usermod command***

• Setting expiry for user	usermod –e 2017-11-25 dan
• Changing home directory	usermod –d /home/new dan
• Changing default shell	usermod –s /bin/sh dan
• Locking an account	usermod –L dan
• Unlocking a locked account	usermod –u dan

Group Management

There are 2 categories of groups in the Linux operating system i.e. **Primary** and **Secondary** groups.

The *Primary Group* is a group that is automatically generated while creating a user with a unique user ID simultaneously a group with ID same as the user ID is created and the user gets added to the group and becomes the first and only member of the group. This group is called the primary group.

The *secondary group* is a group that can be created separately with the help of commands and we can then add users to it by changing the group ID of users.

These operations are performed using the following commands:

Groupadd : Adds groups to the system

Groupmod : Modifies group attributes

Groupdel : Removes groups from the system

Add a user to Group : usermod -aG group user

Following are the commands for managing groups

- **Adding a group** groupadd linuxgroup
- **Adding user to group** usermod –aG linuxgroup dan
- **Changing owner & group of a file** chown dan:linuxgroup newfile.txt
- **Changing only owner of a file** chown dan: newfile.txt
- **Changing only group of a file** chown :linuxgroup newfile.txt
- **Deleting a group** groupdel linuxgroup

Command to Set the Password for the Group: Below command is used to set the password of the group. After executing the command we have to enter the new password which we want to assign to the group. The password has to be given twice for confirmation purposes.

gpasswd group_name

Example:

gpasswd Group1

```
[root@localhost ~]#
[root@localhost ~]# gpasswd Group1
Changing the password for group Group1
New Password:
Re-enter new password:
[root@localhost ~]#
```

```
[root@localhost ~]#
[root@localhost ~]# cat /etc/gshadow
root:::
bin:::
daemon:::
sys:::
adm:::
```

Command to Add a User to an Existing Group:

Below command is used to add a user to an existing group. The users which may be present in any primary or secondary group will exit the other groups and will become the part of this group.

usermod -G group_name username

usermod -G group1 John_Doe

```
[root@localhost ~]# usermod -G Group1 John_Wick
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# tail -3 /etc/group
Group1:x:1000:John_Doe,John_Wick
John_Doe:x:1001:
John_Wick:x:1002:
```

Command to Add User to Group Without Removing From Existing Groups:

This command is used to add a user to a new group while preventing him from getting removed from his existing groups.

usermod -aG *group_name *username

Example: usermod -aG group1 John_Doe

```
[root@localhost ~]#
[root@localhost ~]# groupadd Group2
[root@localhost ~]#
[root@localhost ~]# usermod -aG Group2 John_Wick
[root@localhost ~]# tail -3 /etc/group
John_Doe:x:1001:
John_Wick:x:1002:
Group2:x:1003:John_Wick
[root@localhost ~]# tail -5 /etc/group
screen:x:84:
Group1:x:1000:John_Doe,John_Wick
John_Doe:x:1001:
John_Wick:x:1002:
Group2:x:1003:John_Wick
```

Command to Add Multiple Users to a Group at once:

gpasswd -M *username1, *username2, *username3 ...., *usernamen *group_name

Example:

gpasswd -M Person1, Person2, Person3 Group1

```
[root@localhost ~]# gpasswd -M user1,user2,user3 Group2
[root@localhost ~]# tail -5 /etc/group
John_Wick:x:1002:
Group2:x:1003:user1,user2,user3
user1:x:1004:
user2:x:1005:
user3:x:1006:
```

Command to Delete a User From a Group: Below command is used to delete a user from a group. The user is then removed from the group though it is still a valid user in the system but it is no more a part of the group. The user remains part of the groups which it was in and if it was part of no other group then it will be part of its primary group.

gpasswd -d *username1 *group_name

Example:

gpasswd -d Person1 Group1

Managing Users and Groups

There are four main user administration files −

/etc/passwd − Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system.

/etc/shadow − Holds the encrypted password of the corresponding account. Not all the systems support this file.

/etc/group − This file contains the group information for each account.

/etc/gshadow − This file contains secure group account information.

Check all the above files using the cat command.

/etc/passwd

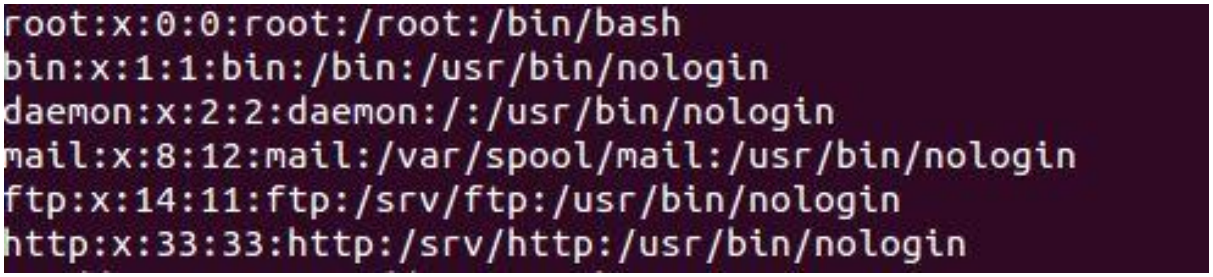
This file contains list of all users with every line of the file containing information regarding single user. Format for each line is

Username:x:UID:GID:Comment:Home Directory: Default shell

Here, **x** is password for the user in encrypted form (stored in /etc/shadow file)

UID, is the user id

& **GID** is the group id for the user.



/etc/group

Just like /etc/passwd, it contains information for groups with each line having information for single group. Format for entries in this file is

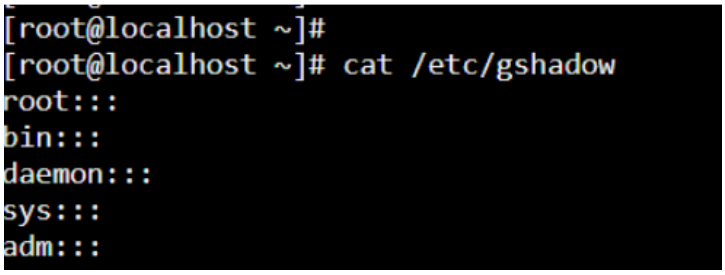
Where, **x** again means password in encrypted format.

/etc/gshadow

Use this Command to Display the Group Password File.

The below command gives us the password file as output. The file is present in a form such that no information about the file is open for the viewers. Instead of this try: “cat /etc/group” to get more information about the groups.

cat /etc/gshadow



sudo: run one or more commands as another user (typically with superuser permissions).

Relevant files: /etc/passwd (user information), /etc/shadow (encrypted passwords), /etc/group (group information) and /etc/sudoers (configuration for sudo).

Superuser permissions can be gained either by changing to the root user with the su command or using sudo. The latter approach is used by default in Ubuntu and derivatives and is preferred over the former in other distributions as well.

The /etc/sudoers File

Now that we have a regular user account created, we will explain how to utilize it to perform user management tasks.

To grant pluralsight superuser permissions: we will need to add an entry for it in /etc/sudoers. This file is used to indicate which users can run what commands with elevated permissions (most likely as root).

Open /etc/sudoers with visudo

Add an Entry in /etc/sudoers for the New User Account

The easiest method to grant superuser permissions for pluralsight is by adding the following line at the bottom of /etc/sudoers:

pluralsight ALL=(ALL) ALL

bash

Let's explain the syntax of this line:

First off, we indicate which user this rule refers to (pluralsight).

The first ALL means the rule applies to all hosts using the same /etc/sudoers file. Nowadays, this means the current host since the same file is not shared across other machines.

Next, (ALL) ALL tells us that pluralsight will be allowed to run all commands as any user. Functionally speaking, this is equivalent to (root) ALL.

Create Command Aliases

An alternative to using the wide permissions outlined above, we can restrict the list of commands that can be executed by a given user by grouping them into sets known as aliases.

For example, we may want to allow user pluralsight to only use adduser and usermod, but not other commands. To do so, we can either list the commands one by one (using the corresponding absolute paths) at the end of the same entry:

pluralsight ALL=(root) /usr/sbin/adduser, /usr/sbin/usermod

or define an alias (which we can name as we wish as long as it's all upper case, for example USERMANAGEMENT):

Cmnd_Alias USERMANAGEMENT = /usr/sbin/adduser, /usr/sbin/usermod

pluralsight ALL=(root) USERMANAGEMENT

While the latter requires two lines, it is often preferred instead of the former because it contributes to keep /etc/sudoers cleaner. In any event, pluralsight will not be able to execute any other commands as root other than those specified above.

For more information on the available options in /etc/sudoers, refer to man sudoers.

While saving, visudo will alert you if a syntax error is found in the file, and indicate the line where the error is found so that you can identify it more easily.

Switching Users

If no errors are found while saving the recent changes in /etc/sudoers, we'll be ready to start using pluralsight to perform user management tasks. To do so, use the su command to change to that account. Note that from this point, there is no need to use the root account if you're in CentOS or similar.

Becoming the Superuser for a Short While

If you want become the superuser to perform important system administration tasks. We have program called su (short for substitute user) and can be used in those cases when you need to be the superuser for a small number of tasks. To become the superuser, simply type the su command. You will be prompted for the superuser's password:

[me@linuxbox me]$ su

Password:

[root@linuxbox me]#

After executing the su command, we have a new shell session as the superuser. To exit the superuser session, type exit and we will return to your previous session.

In most modern distributions, an alternate method is used. Rather than using su, these systems employ the sudo command instead. With sudo, one or more users are granted superuser privileges on an as needed basis. To execute a command as the superuser, the desired command is simply preceded with the sudo command. After the command is entered, the user is prompted for the their own password rather than the superuser's:

[me@linuxbox me]$ sudo some_command

Password for me:

[me@linuxbox me]$

In fact, modern distributions don't even set the root account password thus making it impossible to log in as the root user. A root shell is still possible with sudo by using the "-i" option:

[me@linuxbox me]$ sudo -i

Password for me:

root@linuxbox:~#

Package Management

package

A package file is a collection of files that comprise the software package.

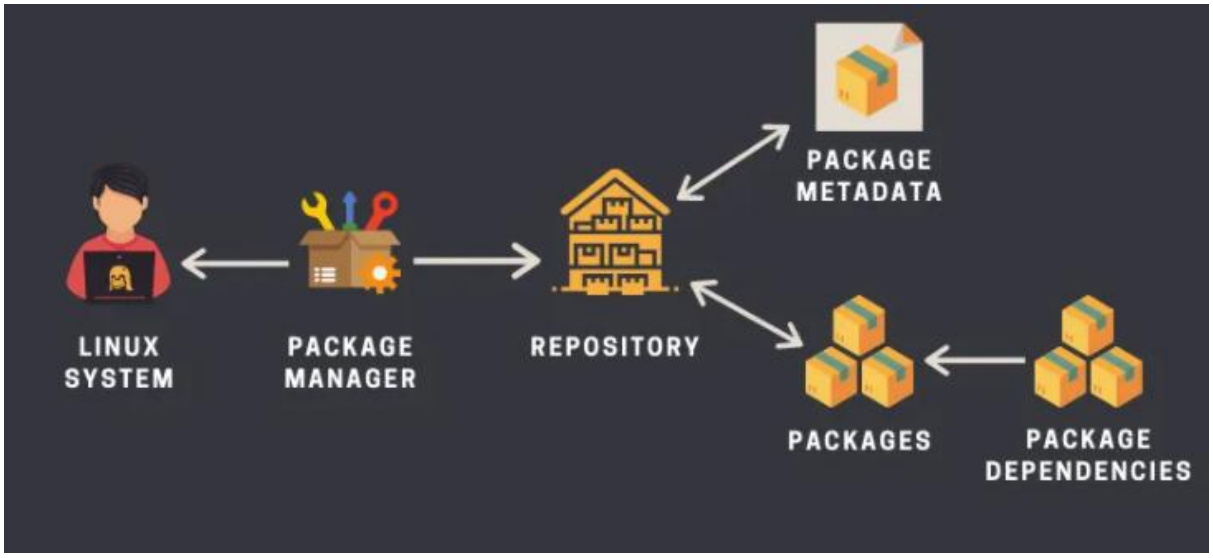
A package may consist of numerous package files and pre- and post-installation scripts that perform configuration tasks before and after the package installation.

High and low-level package tools

In order to perform the task of package management effectively, you need to be aware that you will have two types of available utilities: **low-level** tools (which handle in the backend the actual installation, upgrade, and removal of package files), and **high-level** tools (which are in charge of ensuring that the tasks of dependency resolution and metadata searching -"data about the data"- are performed).

DISTRIBUTION	LOW-LEVEL TOOL	HIGH-LEVEL TOOL
Debian and derivatives	dpkg	apt-get / aptitude
CentOS	rpm	yum
openSUSE	rpm	zypper

How a package manager works.



Linux repository

A **Linux repository** is a storage location from which your system retrieves and installs OS updates and applications. Each repository is a collection of software hosted on a remote server and intended to be used for installing and updating software packages on Linux systems.

Red Hat Enterprise Linux and Fedora-based Distribution Patching

The default RPM package manager for Red Hat and Fedora-based systems is yum (Yellowdog Updater, Modified).

To update the entire operating system, the following command can be used:

yum update

To run an update on a specific package, use this command:

yum update <package-name>

This one is used to entirely remove a package in case it's damaged, and the patch was not released yet:

yum remove <package-name>

Debian-Based Distribution Patching

Debian-based Linux distributions, such as Ubuntu, use APT as the default package manager. APT, or the Advanced Package Tool, uses the apt-get command to install, remove, and update packages.

To run an update on the entire system, use these commands:

apt-get update - Syncs packages with sources. Verifies the right files will be installed

apt-get upgrade - Downloads and installs updates across all software packages.

To update a current package to the latest version, or install one that is not released yet, the following command can be used:

apt-get install <package-name>

To remove an installed package, use the following command:

apt-get remove <package-name>

Again, undesirable packages should be removed with this command until they are confirmed.

Yellow Dog Updater, Modified (YUM)

YUM is the primary package management tool for installing, updating, removing, and managing software packages in Red Hat Enterprise Linux. YUM performs dependency resolution when installing, updating, and removing software packages. YUM can manage packages from installed repositories in the system or from .rpm packages.

The main configuration file for YUM is at **/etc/yum.conf**, and all the repos are at **/etc/yum.repos.d.**

yum check-update

Checks for packages that can update candidates. From this, we will assume this a production system that will be facing the Internet with no production applications that needs to be tested by DevOps before upgrading the packages. Let us now install the updated candidates onto the system.

[root@localhost rdc]# yum check-update

Loaded plugins: fastestmirror, langpacks

Loading mirror speeds from cached hostfile

* base: mirror.scalabledns.com

* extras: mirror.scalabledns.com

* updates: mirror.clarkson.edu

NetworkManager.x86_64 1:1.4.0-19.el7_3 updates

NetworkManager-adsl.x86_64 1:1.4.0-19.el7_3 updates

NetworkManager-glib.x86_64 1:1.4.0-19.el7_3 updates

NetworkManager-libnm.x86_64 1:1.4.0-19.el7_3 updates

[root@localhost rdc]#

yum update

This will install all updated candidates making your CentOS installation current. With a new installation, this can take a little time depending on your installation and your internet connection speed.

[root@localhost rdc]# yum update

vim-minimal x86_64 2:7.4.160-1.el7_3.1 updates 436 k

wpa_supplicant x86_64 1:2.0-21.el7_3 updates 788 k

xfsprogs x86_64 4.5.0-9.el7_3 updates 895 k

Transaction Summary

==========

Install 2 Packages

Upgrade 156 Packages

Total download size: 371 M

Is this ok [y/d/N]:

Most Common YUM Commands

Following are the commonly used YUM commands.

Command	Action

list installed	Lists packages installed via YUM
list all	Lists all currently available packages
group list	Lists grouped packages
info	Provides detailed information about a package
search	Searches package descriptions and names
install	Installs a package
localinstall	Installs a local rpm package
remove	Removes and installs package
clean all	Cleans /var/cache/yum to free disk-space
man yum	Like all linux commands, the help file

[root@localhost rdc]# yum search web browser

Loaded plugins: fastestmirror, langpacks

Loading mirror speeds from cached hostfile

* base: mirror.scalabledns.com

* extras: mirror.scalabledns.com

* updates: mirror.clarkson.edu

==========

N/S matched: web, browser

==========

icedtea-web.x86_64 : Additional Java components for OpenJDK - Java browser

plug-in and Web Start implementation

firefox.i686 : Mozilla Firefox Web browser

firefox.x86_64 : Mozilla Firefox Web browser

lynx.x86_64 : A text-based Web browser

Full name and summary matches only, use "search all" for everything.

[root@localhost rdc]#

We see, CentOS does offer the Lynx web browser in the repository. Let's see some more information about the package.

[root@localhost rdc]# lynx.x86_64

bash: lynx.x86_64: command not found...

[root@localhost rdc]# yum info lynx.x86_64

Loaded plugins: fastestmirror, langpacks

Loading mirror speeds from cached hostfile

 * base: mirror.scalabledns.com

 * extras: mirror.scalabledns.com

 * updates: mirror.clarkson.edu

Available Packages

Name : lynx

Arch : x86_64

Version : 2.8.8a

Release : 0.3.dev15.el7

Size : 1.4 M

Repo : base/7/x86_64

Summary : A text-based Web browser

URL : http://lynx.isc.org/

License : GPLv2

Description : Lynx is a text-based Web browser. Lynx does not display any images,

: but it does support frames, tables, and most other HTML tags. One

: advantage Lynx has over graphical browsers is speed; Lynx starts and

: exits quickly and swiftly displays web pages.

[root@localhost rdc]#

CentOS and Fedora

rpm

Task	Command
Install a .rpm file	rpm -i package-file-name.rpm
Remove a .npm file	rpm --erase package-name(s)
Listing Installed Files	rpm -qa
Check if a given package is installed	rpm --query package-name(s)

yum

Task	Command
Installs the package(s) with dependency	yum install package-name(s)
Remove the package(s) but not dependency	yum erase package-name(s)
Update Package List	yum update
search Packages	yum search search-pattern
get info about a package	yum info package-name(s)

Ubuntu's package management system is derived from the same system used by the Debian GNU/Linux distribution. The package files contain all of the necessary files, meta-data, and instructions to implement a particular functionality or software application on your Ubuntu computer.

Debian package files typically have the extension .deb, and usually exist in repositories which are collections of packages found online or on physical media, such as CD-ROM discs. Packages are normally in a pre-compiled binary format; thus installation is quick and requires no compiling of software.

Many packages use dependencies. Dependencies are additional packages required by the principal package in order to function properly. For example, the speech synthesis package festival depends upon the package alsa-utils, which is a package supplying the ALSA sound library tools needed for audio playback. In order for festival to function, it and all of its dependencies must be installed. The software management tools in Ubuntu will do this automatically.

Apt

The apt command is a powerful command-line tool, which works with Ubuntu's *Advanced Packaging Tool* (APT) performing such functions as installation of new software packages, upgrade of existing software packages, updating of the package list index, and even upgrading the entire Ubuntu system.

Install a Package: Installation of packages using the apt tool is quite simple. For example, to install the nmap network scanner, type the following:

sudo apt install nmap

Remove a Package: Removal of a package (or packages) is also straightforward. To remove the package installed in the previous example, type the following:

sudo apt remove nmap

Update the Package Index: The APT package index is essentially a database of available packages from the repositories defined in the /etc/apt/sources.list file and in the /etc/apt/sources.list.d directory. To update the local package index with the latest changes made in the repositories, type the following:

sudo apt update

Upgrade Packages: Over time, updated versions of packages currently installed on your computer may become available from the package repositories (for example security updates). To upgrade your system, first, update your package index as outlined above, and then type:

sudo apt upgrade

Debian and Ubuntu

dkg

Task	Command
Install a .deb file	dpkg -i package-file-name.deb
Remove a .deb file	dpkg -r package-file-name.deb
List installed Packages	dpkg --list search-pattern
Check if a package is installed	dpkg -s package-name
location of installed file	dpkg -L package-name

apt

Task	Command
Installs the package(s) with dependency	apt-get install package-name(s)
Remove the package(s) but not dependency	apt-get remove package-name(s)
Remove unused dependency	apt-get autoremove
Update Package List	apt-get update
search Packages	apt search search_string
get info about a package	apt show package

Input Output Redirection in Linux

Introduction

- The shell associates three files with the terminal.
- Two for the display and one for the keyboard.
- We see command output and error messages on the display and we provide command input through the keyboard.
- The command perform all terminal related activity with three files that the shell makes available to every command-

1. Standard Input
2. Standard Output
3. Standard Error

Redirection

Redirection is a feature in Linux such that when executing a command, you can change the standard input/output devices. The basic workflow of any Linux command is that it takes an input and give an output.

- The standard input (stdin) device is the keyboard.
- The standard output (stdout) device is the screen.
- With redirection, the above standard input/output can be changed.

- `>` Redirects standard output to a file. Overwrites (truncating) existing contents.
- `>>` Redirects standard output to a file. Appends to any existing contents.
- `<` Redirects input from a file to a command.

Redirecting output

- Using the "greater-than" sign with a file name like this:

```
ls > file2
```

Causes the shell to place the output from the command ls in a file called "file2" instead of on the screen. If the file "file2" already exists, the old version will be overwritten.

Redirecting input

- Using the "less-than" sign with a file name like this:

  ```
  < file1
  ```

  in a shell command instructs the shell to read input from a file called "file1" instead of from the keyboard.

```
user@ubuntu:~/Desktop$ ls
abcd  abcde  chapters  commandSub  escape  file  file~  file2  wxyz
user@ubuntu:~/Desktop$ cat file
foo
user@ubuntu:~/Desktop$ tr 'o' 'b' file
tr: extra operand `file'
Try `tr --help' for more information.
user@ubuntu:~/Desktop$ tr 'o' 'b' < file
fbb
user@ubuntu:~/Desktop$
```

Error Redirection

File	File Descriptor
Standard Input STDIN	0
Standard Output STDOUT	1
Standard Error STDERR	2

Why Error Redirection?

While executing shell scripts, you often do NOT want error messages cluttering up the normal program output.

The solution is to re-direct the error messages to a file.

And here comes error redirection!

```
user@ubuntu:~/Desktop$ rcho Hi
No command 'rcho' found, did you mean:
 Command 'echo' from package 'coreutils' (main)
 Command 'rwho' from package 'rwho' (universe)
rcho: command not found
user@ubuntu:~/Desktop$ rcho Hi 2>errorfile
user@ubuntu:~/Desktop$
```

Linux networking Introduction

Networking is a process of connecting two or more electronic devices for the purpose of exchange of information's and devices

How does a computer network work:

Specialized devices such as switches, routers, and access points form the foundation of computer networks.

Switches connect and help to internally secure computers, printers, servers, and other devices to networks in homes or organizations. Access points are switches that connect devices to networks without the use of cables.

Routers connect networks to other networks and act as dispatchers. They analyze data to be sent across a network, choose the best routes for it, and send it on its way. Routers connect your home and business to the world and help protect information from outside security threats.

While switches and routers differ in several ways, one key difference is how they identify end devices. A Layer 2 switch uniquely identifies a device by its "burned-in" MAC address. A Layer 3 router uniquely identifies a device's network connection with a network-assigned IP address.

Today, most switches include some level of routing functionality.

MAC and IP addresses uniquely define devices and network connections, respectively, in a network. A MAC address is a number assigned to a network interface card (NIC) by a device's manufacturer. An IP address is a number assigned to a network connection.

A routing table is a set of rules, often viewed in table format, that is used to determine where data packets traveling over an Internet Protocol (IP) network will be directed. All IP-enabled devices, including routers and switches, use routing tables.

A routing table contains the information necessary to forward a packet along the best path toward its destination. Each packet contains information about its origin and destination.

```
IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0       10.0.0.1        10.0.0.75     35
         10.0.0.0    255.255.255.0       On-link         10.0.0.75    291
        10.0.0.75  255.255.255.255       On-link         10.0.0.75    291
       10.0.0.255  255.255.255.255       On-link         10.0.0.75    291
        127.0.0.0        255.0.0.0       On-link          127.0.0.1    331
        127.0.0.1  255.255.255.255       On-link          127.0.0.1    331
  127.255.255.255  255.255.255.255       On-link          127.0.0.1    331
    192.168.56.0    255.255.255.0       On-link      192.168.56.1    281
    192.168.56.1  255.255.255.255       On-link      192.168.56.1    281
  192.168.56.255  255.255.255.255       On-link      192.168.56.1    281
        224.0.0.0        240.0.0.0       On-link          127.0.0.1    331
        224.0.0.0        240.0.0.0       On-link      192.168.56.1    281
        224.0.0.0        240.0.0.0       On-link         10.0.0.75    291
  255.255.255.255  255.255.255.255       On-link          127.0.0.1    331
```

Key Words / Terminologies

Host name:

Each device in the network is associated with a unique device name known as Hostname.

Type “hostname” in the command prompt(Administrator Mode) and press ‘Enter’, this displays the hostname of your machine.

IP Address (Internet Protocol address):

Also known as the Logical Address, the IP Address is the network address of the system across the network.

To identify each device in the world-wide-web, the Internet Assigned Numbers Authority (IANA) assigns an IPV4 (Version 4) address as a unique identifier to each device on the Internet.

The length of an IPv4 address is 32-bits, hence, we have 232 IP addresses available. The length of an IPv6 address is 128-bits.

Type “ipconfig” in the command prompt and press ‘Enter’, this gives us the IP address of the device.

MAC Address (Media Access Control address):

Also known as physical address, the MAC Address is the unique identifier of each host and is associated with its NIC (Network Interface Card).

A MAC address is assigned to the NIC at the time of manufacturing.

The length of the MAC address is : 12-nibble/ 6 bytes/ 48 bits

Type “ipconfig/all” in the command prompt and press ‘Enter’, this gives us the MAC address.

DNS Server:

DNS stands for Domain Name system.

DNS is basically a server which translates web addresses or URLs (ex: www.google.com) into their corresponding IP addresses. We don’t have to remember all the IP addresses of each and every website.

The command ‘nslookup’ gives you the IP address of the domain you are looking for. This also provides the information of our DNS Server.

Port:

A port can be referred to as a logical channel through which data can be sent/received to an application. Any host may have multiple applications running, and each of these applications is identified using the port number on which they are running.

A port number is a 16-bit integer, hence, we have 216 ports available which are categorized as shown below:

Port Types	Range
Well known Ports	0 – 1023
Registered Ports	1024 – 49151
Ephemeral Ports	49152 – 65535

Socket:

The unique combination of IP address and Port number together are termed as Socket.

Few important Port Numbers

20 – FTP Data (For transferring FTP data)

21 – FTP Control (For starting FTP connection)

22 – SSH (For secure remote administration which uses SSL to encrypt the transmission)

23 – Telnet (For insecure remote administration)

25 – SMTP (Mail Transfer Agent for e-mail server such as SEND mail)

53 – DNS (Special service which uses both TCP and UDP)

68 – DHCP

80 – HTTP/WWW(Apache)

123 – NTP (Network time protocol used for time syncing uses UDP protocol)

161 – SNMP (For network monitoring)

443 – HTTPS (HTTP+SSL for secure web access)

3306 – MySql

3389 - Windows RDP

3690 – SVN

8080 - Jenkins and Tomcat

Linux networking commands

1. ifconfig

Linux ifconfig stands for interface configurator. It is one of the most basic commands used in network inspection.

ifconfig is used to initialize an interface, configure it with an IP address, and enable or disable it. It is also used to display the route and the network interface.

Basic information displayed upon using ifconfig are:

IP address

MAC address

MTU(Maximum Transmission Unit)

To get all the details using ifconfig

Syntax:

Ifconfig

Output:

shows the IP address of networks, Ethernet, local network, and WLAN.

To get details of specific interface

Using this command, you can get details of a specific interface. This is shown below.

Commands:

ifconfig eth0

2. traceroute

Linux traceroute is one of the most useful commands in networking. It is used to troubleshoot the network. It detects the delay and determines the pathway to your target. It basically helps in the following ways:

Command:

$ traceroute google.com

The output provides the following information:

The specified hostname

Size of the packets

The maximum number of hops required.

The IP address.

To avoid the reverse DNS lookup, add -n in the command syntax.

Command:

$ traceroute -n google.com

The output indicates the network delays. The asterisks shown in the output indicates a potential problem in reaching that host. They indicate the packet loss during communication to the network.

3.ping

Linux ping is one of the most used network troubleshooting commands. It basically checks for the network connectivity between two nodes.

ping stands for Packet INternet Groper.

The ping command sends the ICMP echo request to check the network connectivity.

It keeps executing until it is interrupted.

Use Ctrl+C Key to interrupt the execution.

Syntax:

ping <destination>

Example:

Command:

$ ping google.com

The ping shows a successful connection to google.com

You can also use the IP address to ping directly.

You can limit the number of packets by including "-c" in the ping command.

Syntax:

ping -c <number> <destination>

You can specify the c count and limit the response packets to that.

4.netstat

Linux netstat command refers to the network statistics.

It provides statistical figures about different interfaces which include open sockets, routing tables, and connection information.

Syntax:

netstat

Output:

Observe the output displaying all the open sockets.

5.curl & wget

Linux curl and wget commands are used in downloading files from the internet through CLI. The curl command has to be used with the option "O" to fetch the file, while the wget command is used directly.

a) Curl

Syntax:

```
curl [options] [URL...]
```

Options:

- **-o :** saves the downloaded file on the local machine with the name provided in the parameters.
  Syntax:
  ```
  curl -o [file_name] [URL...]
  ```
  Example:
  ```
  curl -o hello.zip ftp://speedtest.tele2.net/1MB.zip
  ```

b) wget

Syntax:

wget <fileLink>

Example:

wget google.com/doodles/new-years-day-2012

6.dig

Linux dig command stands for Domain Information Groper. This command is used in DNS lookup to query the DNS name server. It is also used to troubleshoot DNS related issues.

It is mainly used to verify DNS mappings, MX Records, host addresses, and all other DNS records for a better understanding of the DNS topography.

This command is an improvised version of nslookup command.

Syntax:

dig <domainName>

Example:

$ dig google.com

Output:

dig command outputs the A records by default. If you want to specifically search for MX or NS type, use the syntax below.

Command: $ dig google.com MX

To get all types of records at once, use the keyword ANY ass below:

Command:

$ dig google.com ANY

The dig command does the query on the servers listed in /etc/resolv.conf.

7.nslookup

Linux nslookup is also a command used for DNS related queries. It is the older version of dig.

Syntax:

nslookup <domainName>

Example:

nslookup mindmajix.com

Output:

As we see in the output above, it displays the record information relating to

mindmajix.com

8.host

Linux host command displays the domain name for a given IP address and IP address for a given hostname. It is also used to fetch DNS lookup for DNS related query.

Example:

host mindmajix.com

host 149.77.21.18

You can combine the host command with -t, and get DNS resource records like SOA, NS, A, PTR, CNAME, MX, SRV.

Syntax:

host -t <resourceName>

9.route

Linux route command displays and manipulates the routing table existing for your system.

A router is basically used to find the best way to send the packets across to a destination.

Syntax:

route

Output:

The above output displays all the existing routing table entries for the system. It says that if the destination address is within the network range of 10.0.0.0 to 10.0.0.255, then the gateway is *, which 0.0.0.0. This is a special address that indicates a non-existent destination.

The packets which lie outside this network range will be forwarded to the default gateway, which is further routed.

Displaying numerical IP address

You can use -n in the option in the syntax to display the output incomplete numerical form.

Syntax:

route -n

To add a gateway

The packets that are not within the range are forwarded to the specific gateway. You can specify the gateway address using the following command.

10.hostname

Linux hostname is the simple command used to view and set the hostname of a system.

Syntax:

hostname

Output:

To set the hostname

Use the syntax below to set the hostname.

Syntax:

sudo hostname <newName>

The hostname set through this command is not permanent. It will be reset to the name in the hostname file back when the system reboots.

In order to permanently set a hostname, you have to re-write the hostname in the hostname file, present on the server. Once set, you have to reboot the box.

In Ubuntu, /etc/hostname file is used.

In RHEL, /etc/sysconfig/network is used.

Unix / Linux - What is Shells?

A Shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

The original Unix shell was written in the mid-1970s by Stephen R. Bourne while he was at the AT&T Bell Labs in New Jersey.

Shell Prompt

The prompt, $, which is called the command prompt, is issued by the shell. While the prompt is displayed, you can type a command.

Shell reads your input after you press Enter. It determines the command you want executed by looking at the first word of your input. A word is an unbroken set of characters. Spaces and tabs separate words.

Following is a simple example of the date command, which displays the current date and time −

$date

Thu Jun 25 08:30:19 MST 2009

Shell Types

In Unix, there are two major types of shells −

Bourne shell −

If you are using a Bourne-type shell, the $ character is the default prompt.

Bourne shell was the first shell to appear on Unix systems, thus it is referred to as "the shell".

Bourne shell is usually installed as /bin/sh on most versions of Unix. For this reason, it is the shell of choice for writing scripts that can be used on different versions of Unix.

The Bourne Shell has the following subcategories −

Bourne shell (sh)

Korn shell (ksh)

Bourne Again shell (bash)

POSIX shell (sh)

C shell –

If you are using a C-type shell, the % character is the default prompt.

The different C-type shells follow −

C shell (csh)

TENEX/TOPS C shell (tcsh)

Shell Scripts

The basic concept of a shell script is a list of commands, which are listed in the order of execution.

It would be a simple text file in which we would put all our commands and several other required constructs that tell the shell environment what to do and when to do it.

Shell scripts and functions are both interpreted. This means they are not compiled.

Example Script

#!/bin/sh

This tells the system that the commands that follow are to be executed by the Bourne shell. It's called a shebang because the # symbol is called a hash, and the ! symbol is called a bang.

To create a script containing these commands, you put the shebang line first and then add the commands −

#!/bin/bash

pwd

ls

Save the above content and make the script executable −

$chmod 755 test.sh