Experiment No. 3

Aim

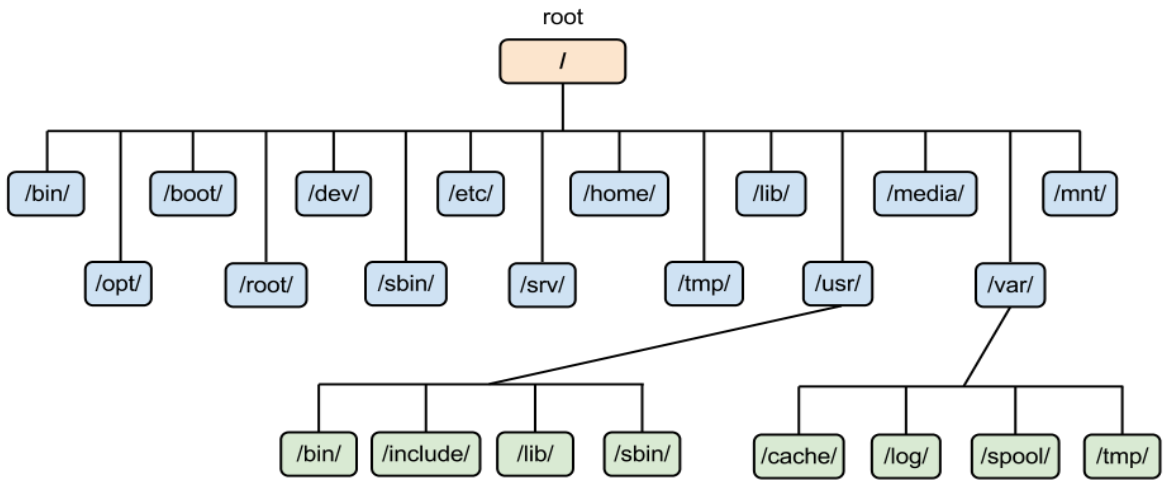
File system hierarchy in a common Linux distribution, file and device permissions, study of system configuration files in /etc, familiarizing log files for system events, user activity, network events.

Result

LINUX File Hierarchy Structure

The Linux File Hierarchy Structure or the Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Unix-like operating systems. It is maintained by the Linux Foundation.

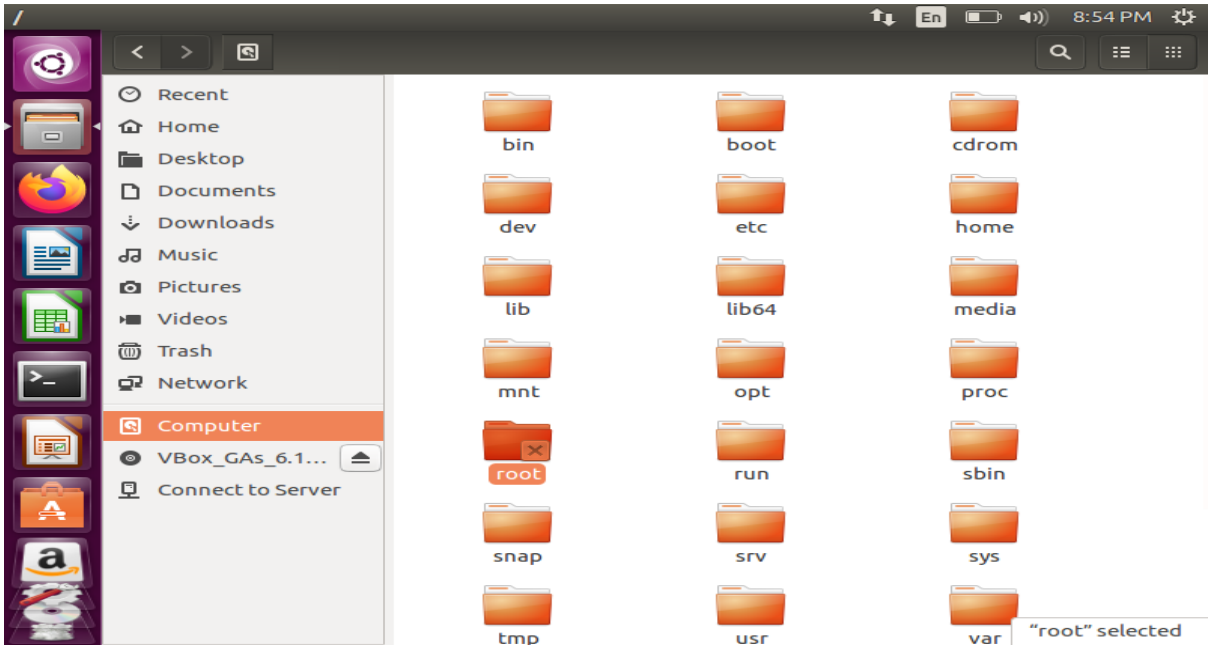
- In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.
- Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.
- Most of these directories exist in all UNIX operating systems and are generally used in much the same way; however, the descriptions here are those used specifically for the FHS and are not considered authoritative for platforms other than Linux.



1. / (Root): Primary hierarchy root and root directory of the entire file system hierarchy.

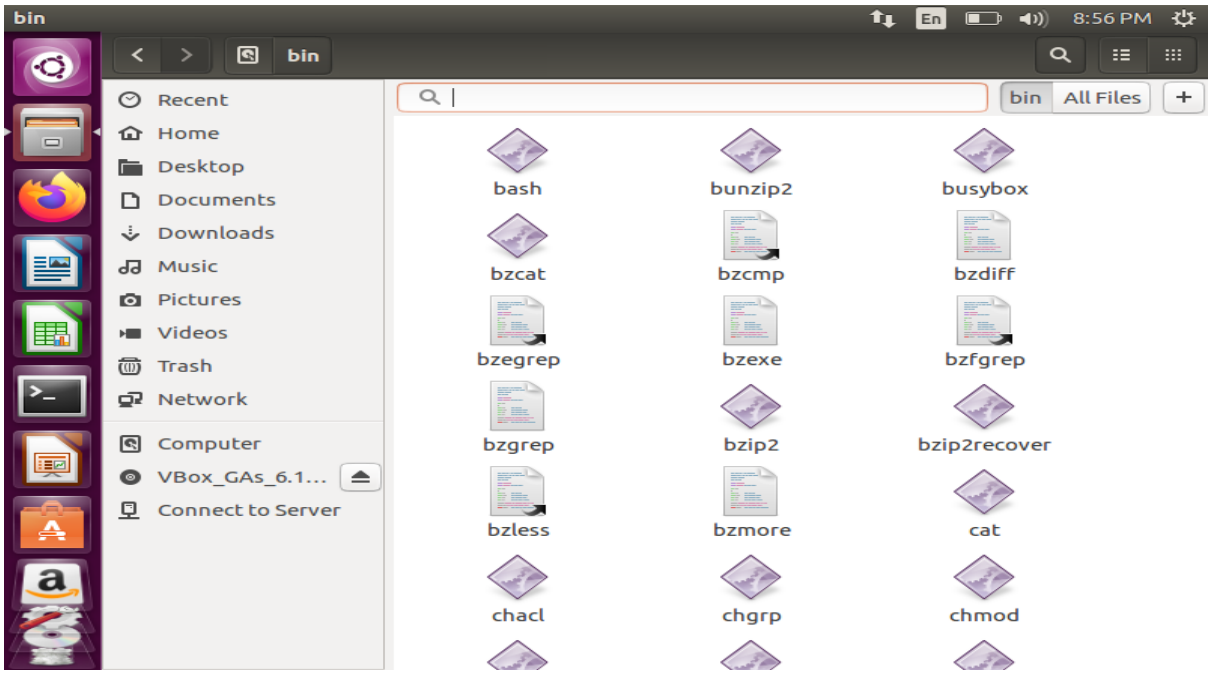
- Every single file and directory starts from the root directory

- The only root user has the right to write under this directory
- /root is the root user's home directory, which is not the same as /



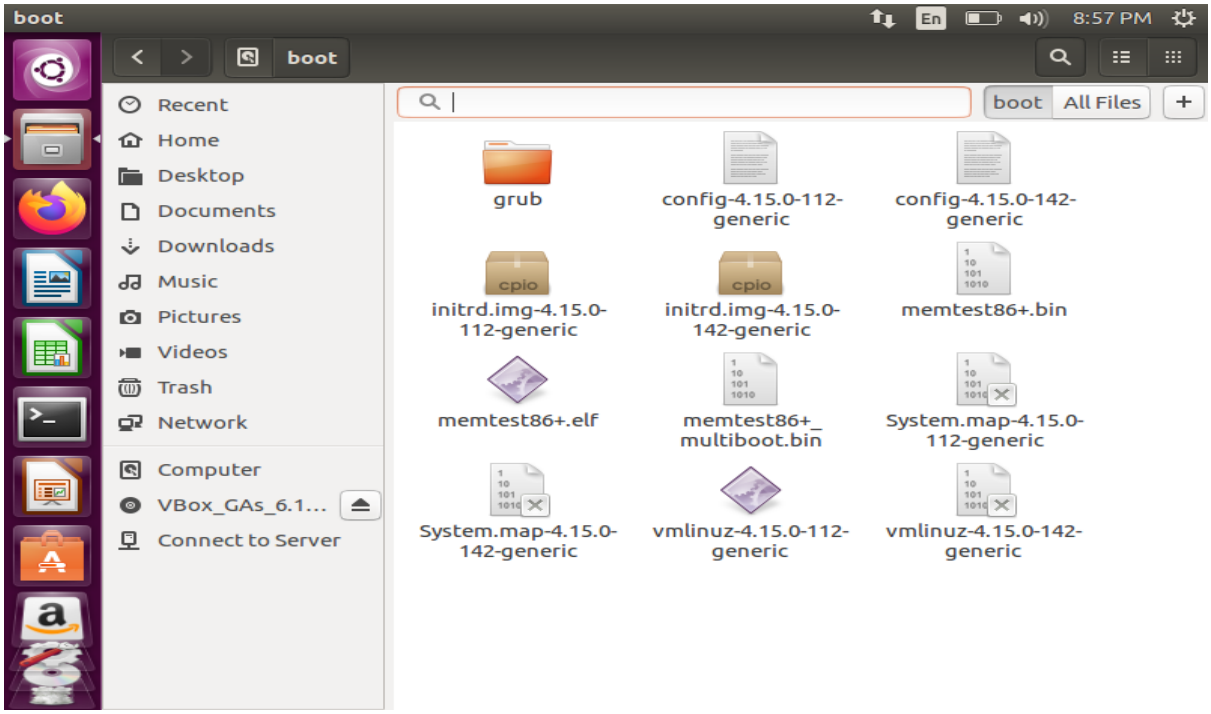
2. /bin : Essential command binaries that need to be available in single-user mode; for all users, e.g., cat, ls, cp.

- Contains binary executables
- Common linux commands you need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here e.g. ps, ls, ping, grep, cp



3. /boot : Boot loader files, e.g., kernels, initrd.

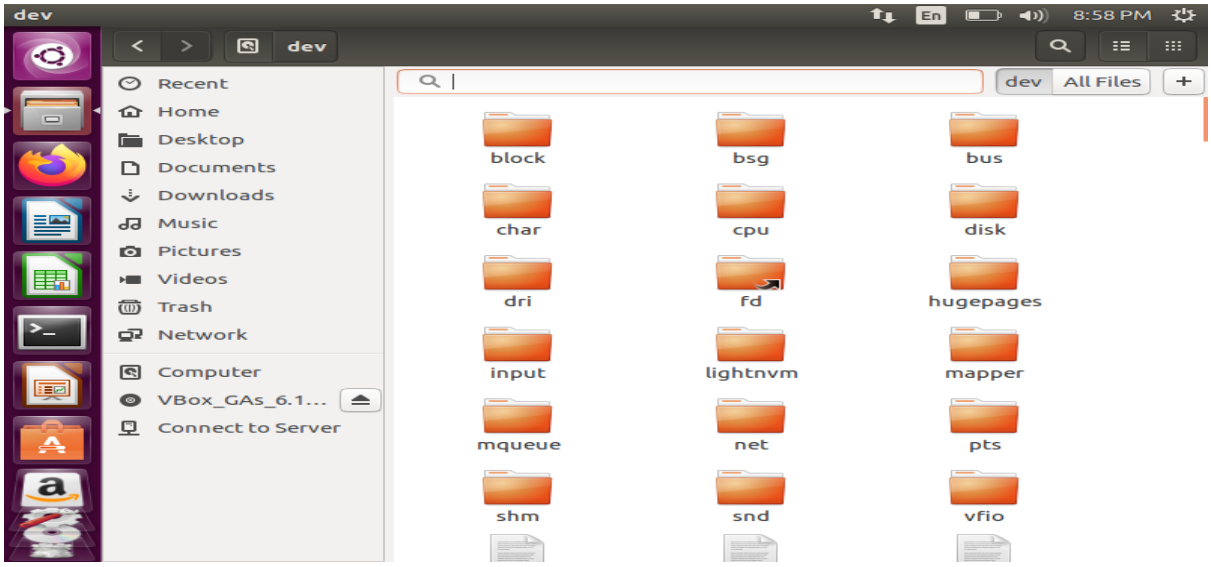
- Kernel initrd, vmlinux, grub files are located under /boot
- Example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic



4. /dev : Essential device files, e.g., /dev/null.

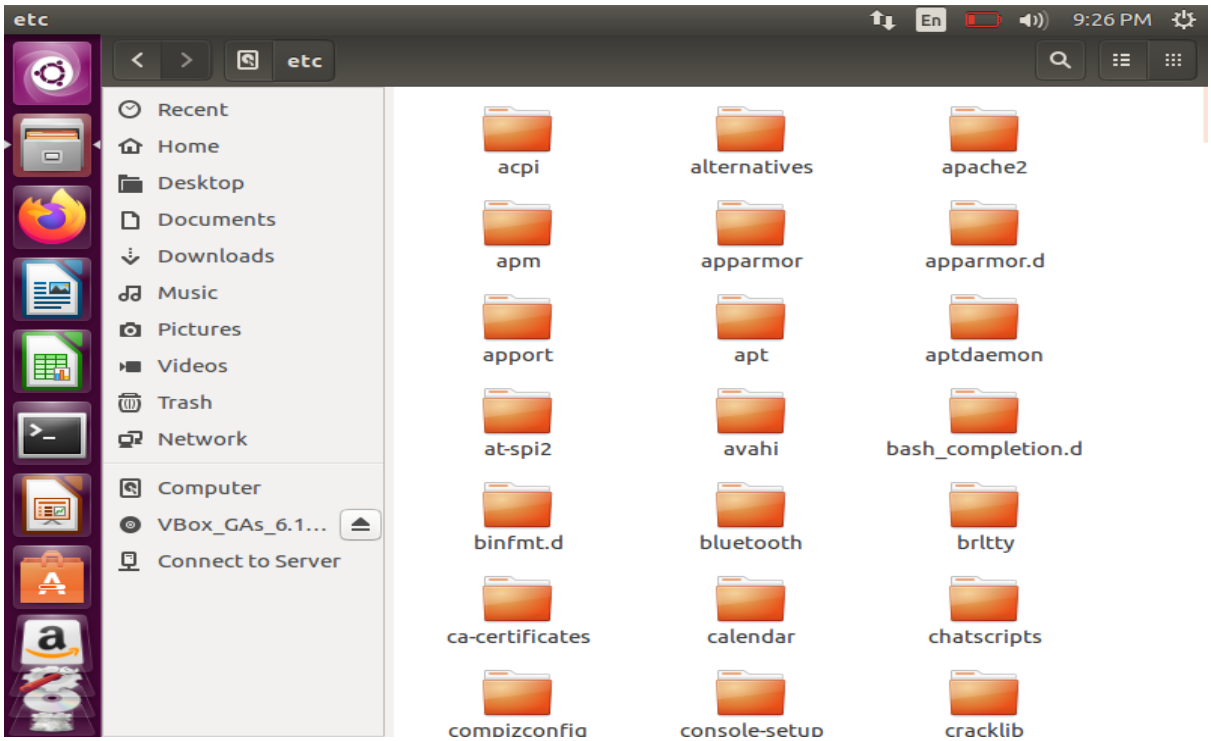
- These include terminal devices, usb, or any device attached to the system.

- Example: /dev/tty1, /dev/usbmon0



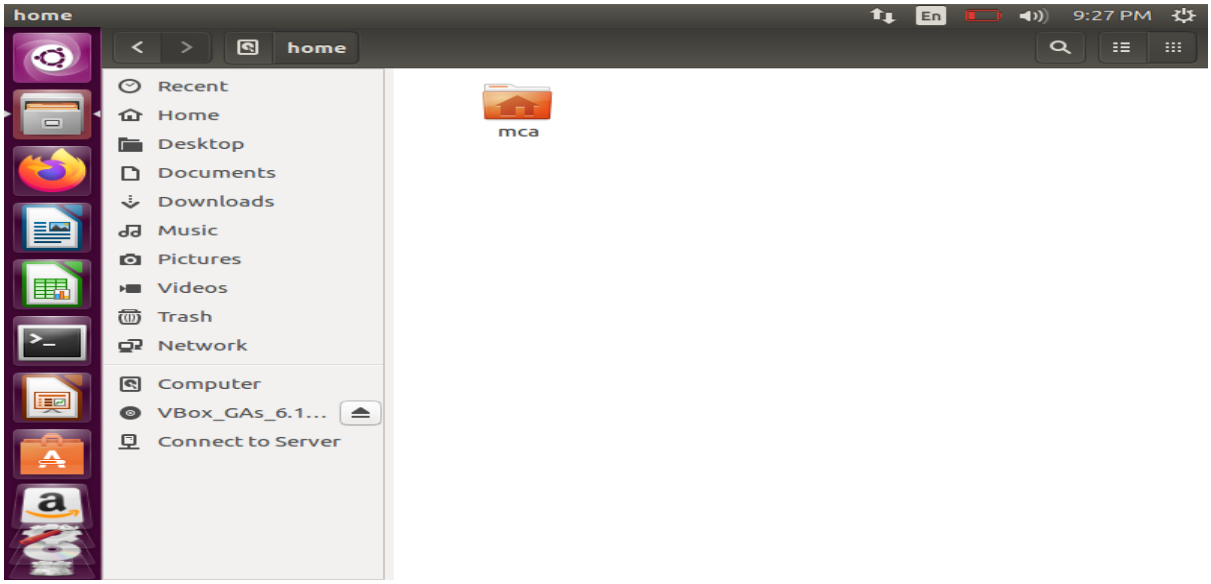
5. /etc : Host-specific system-wide configuration files.

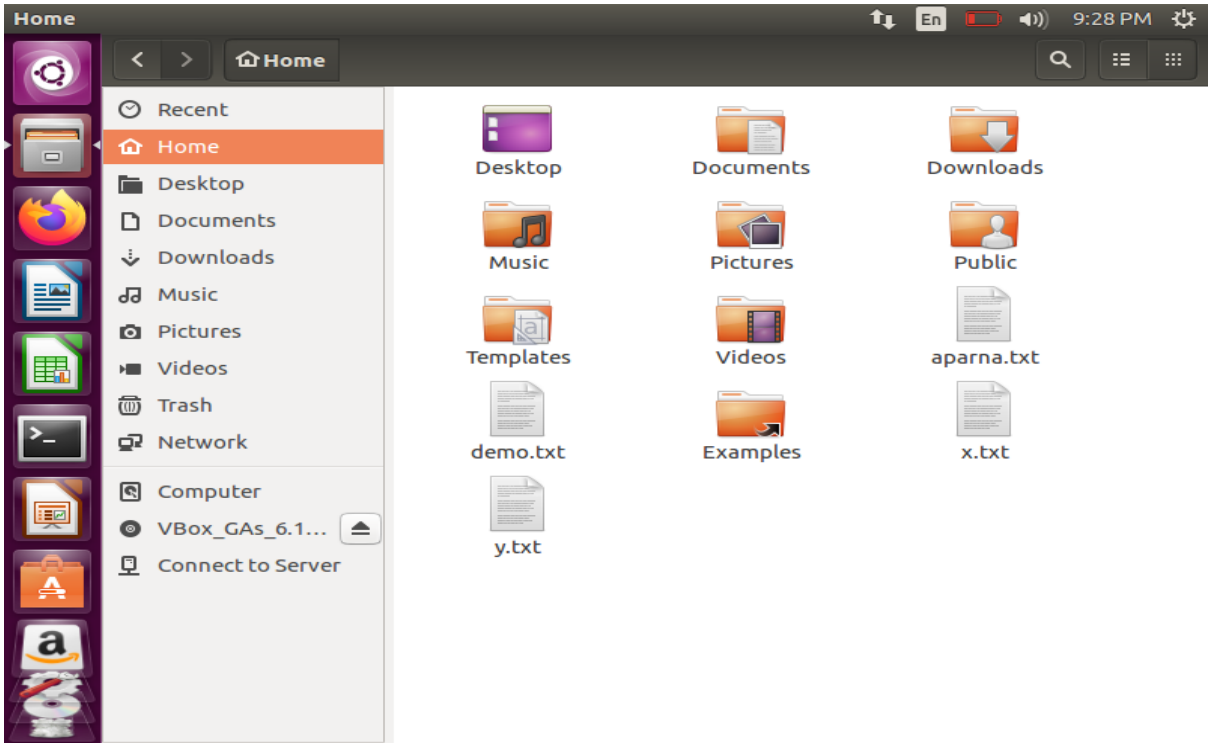
- Contains configuration files required by all programs.
- This also contains startup and shutdown shell scripts used to start/stop individual programs.
- Example: /etc/resolv.conf, /etc/logrotate.conf.



6. /home : Users' home directories, containing saved files, personal settings, etc.

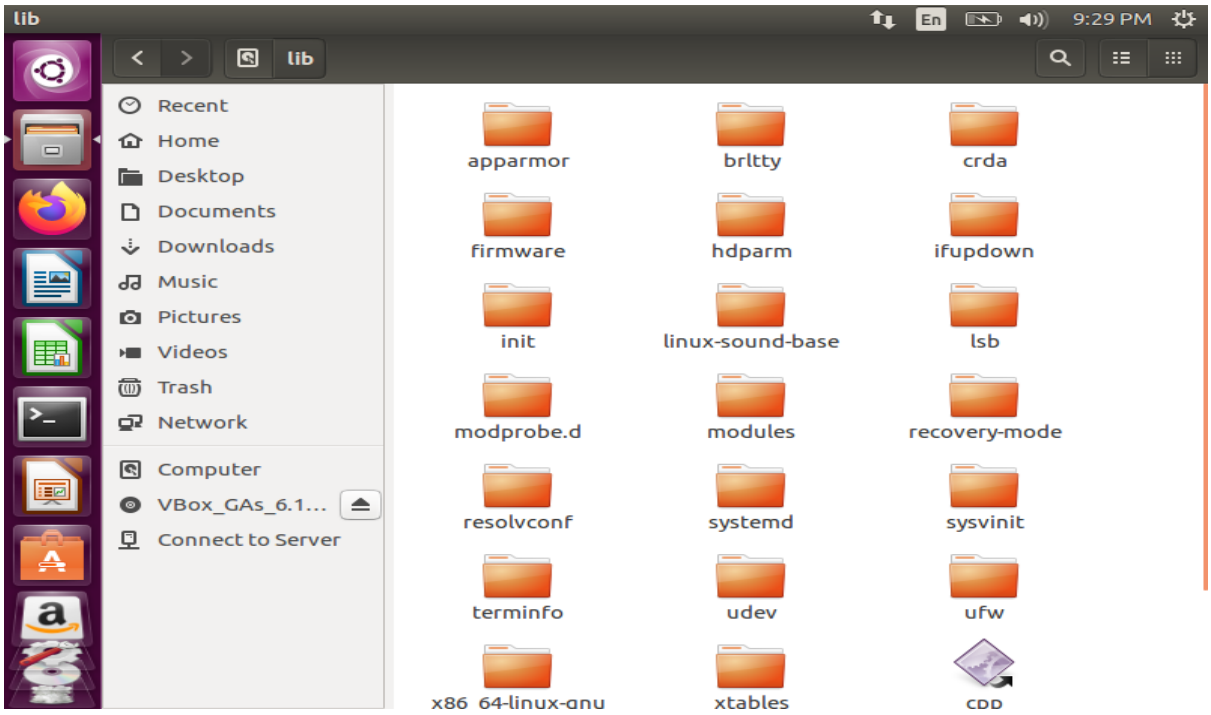
- Home directories for all users to store their personal files.
- example: /home/kishlay, /home/kv





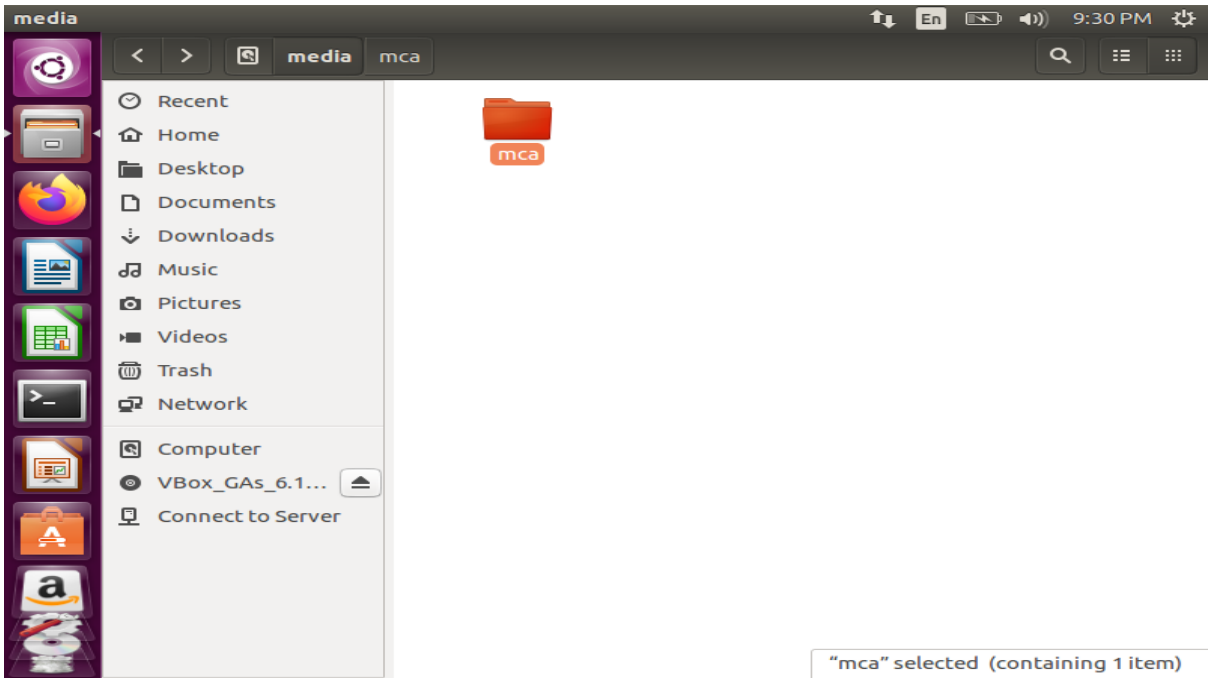
7. /lib : Libraries essential for the binaries in /bin/ and /sbin/.

- Library filenames are either ld* or lib*.so.*
- Example: ld-2.11.1.so, libncurses.so.5.7



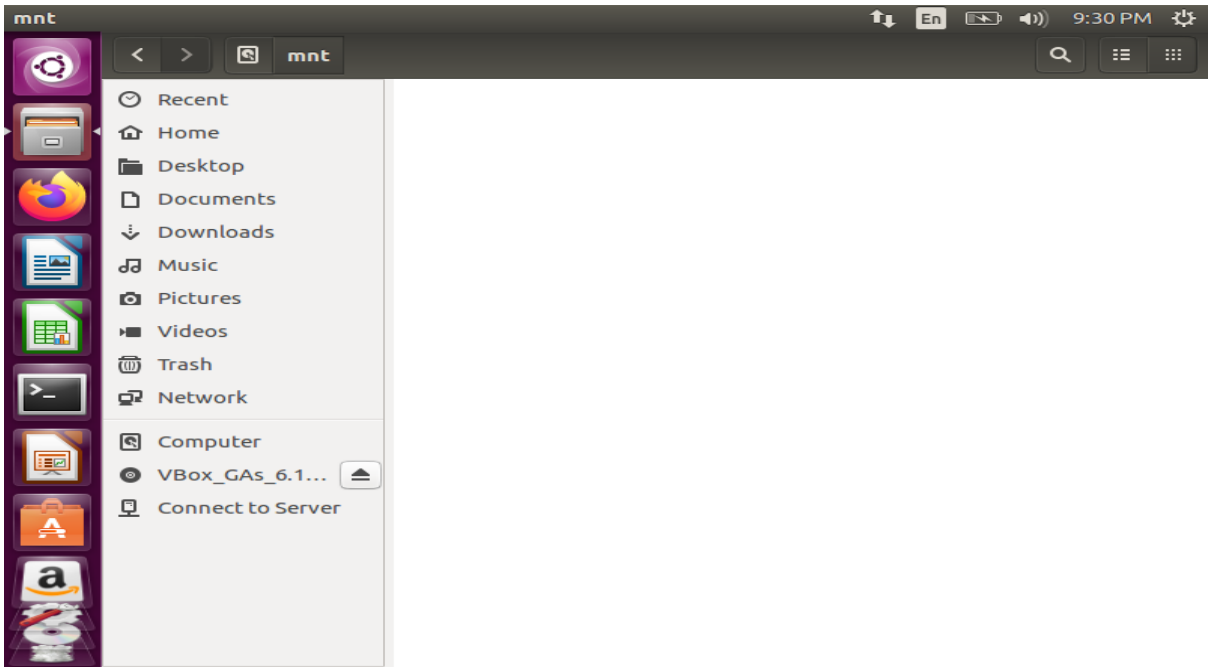
8. /media : Mount points for removable media such as CD-ROMs (appeared in FHS-2.3).

- Temporary mount directory for removable devices.
- Examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer



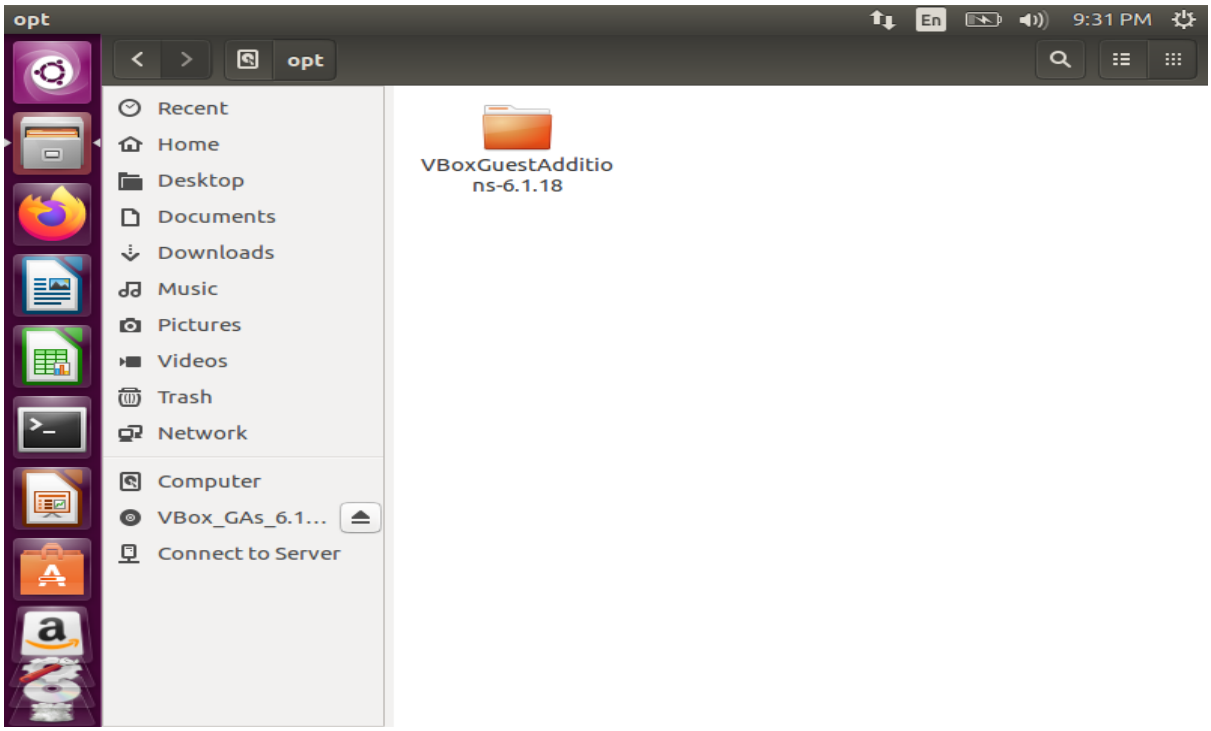
9. /mnt : Temporarily mounted filesystems.

- Temporary mount directory where sysadmins can mount filesystems.



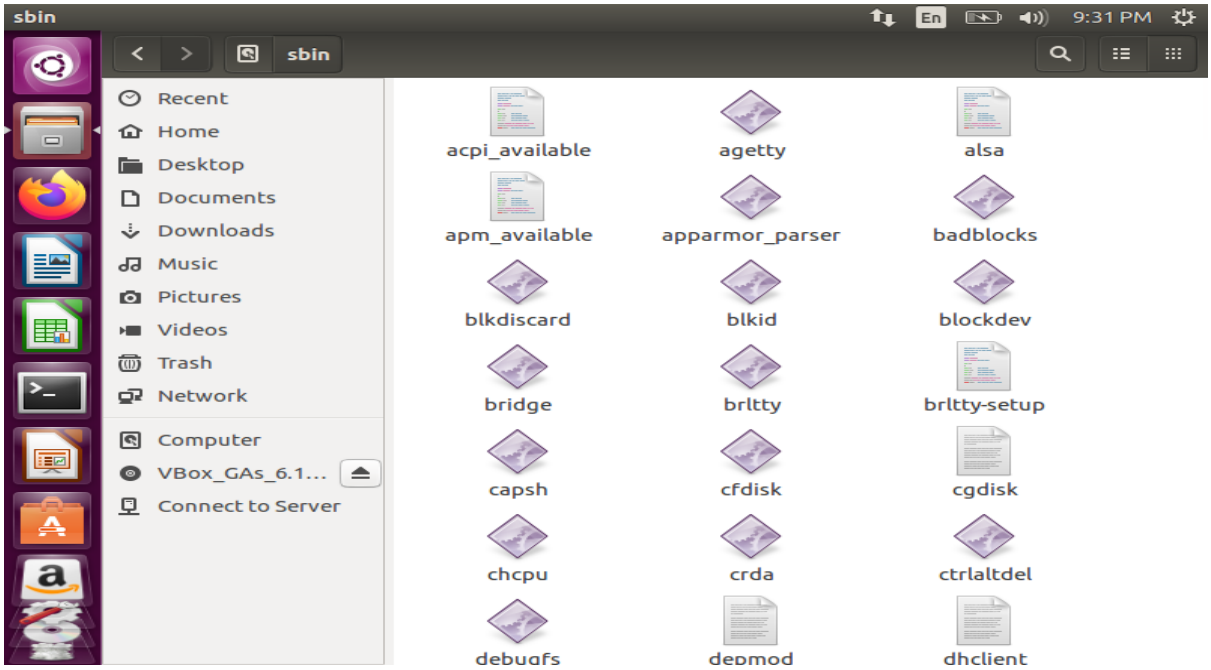
10. /opt : Optional application software packages.

- Contains add-on applications from individual vendors.
- Add-on applications should be installed under either /opt/ or /opt/ sub-directory.



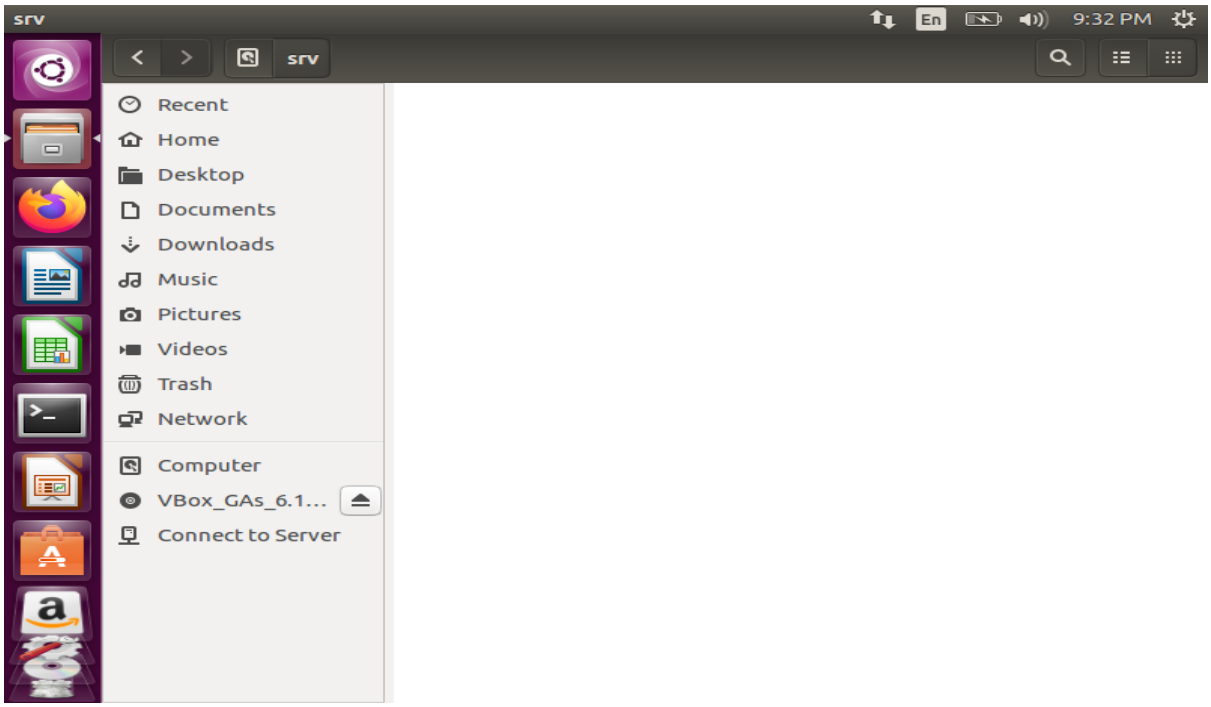
11. /sbin : Essential system binaries, e.g., fsck, init, route.

- Just like /bin, /sbin also contains binary executables.
- The linux commands located under this directory are used typically by system administrator, for system maintenance purpose.
- Example: iptables, reboot, fdisk, ifconfig, swapon



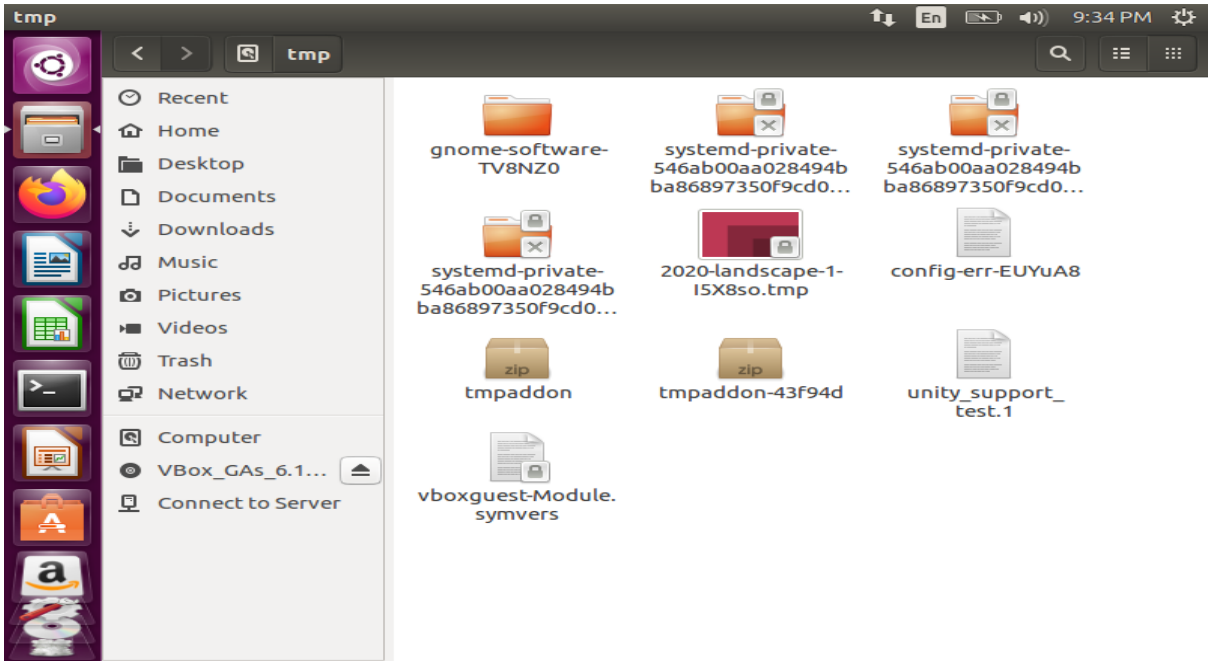
12. /srv : Site-specific data served by this system, such as data and scripts for web servers, data offered by FTP servers, and repositories for version control systems.

- srv stands for service.
- Contains server specific services related data.
- Example, /srv/cvs contains CVS related data.



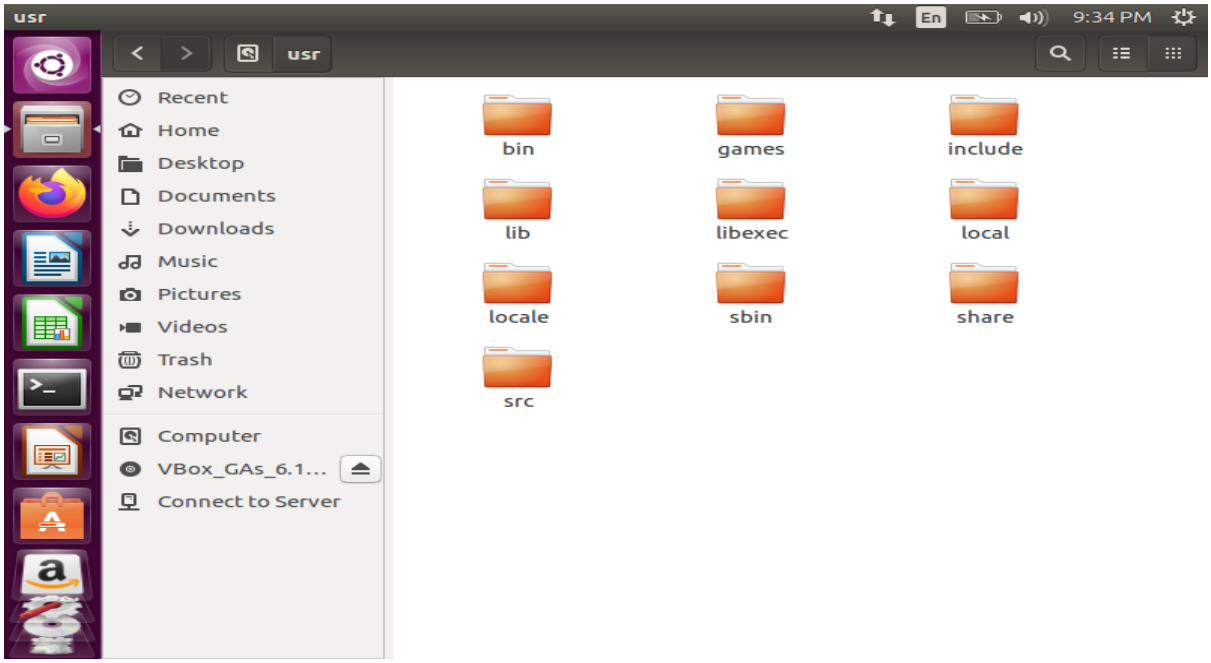
13. /tmp : Temporary files. Often not preserved between system reboots, and may be severely size restricted.

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.



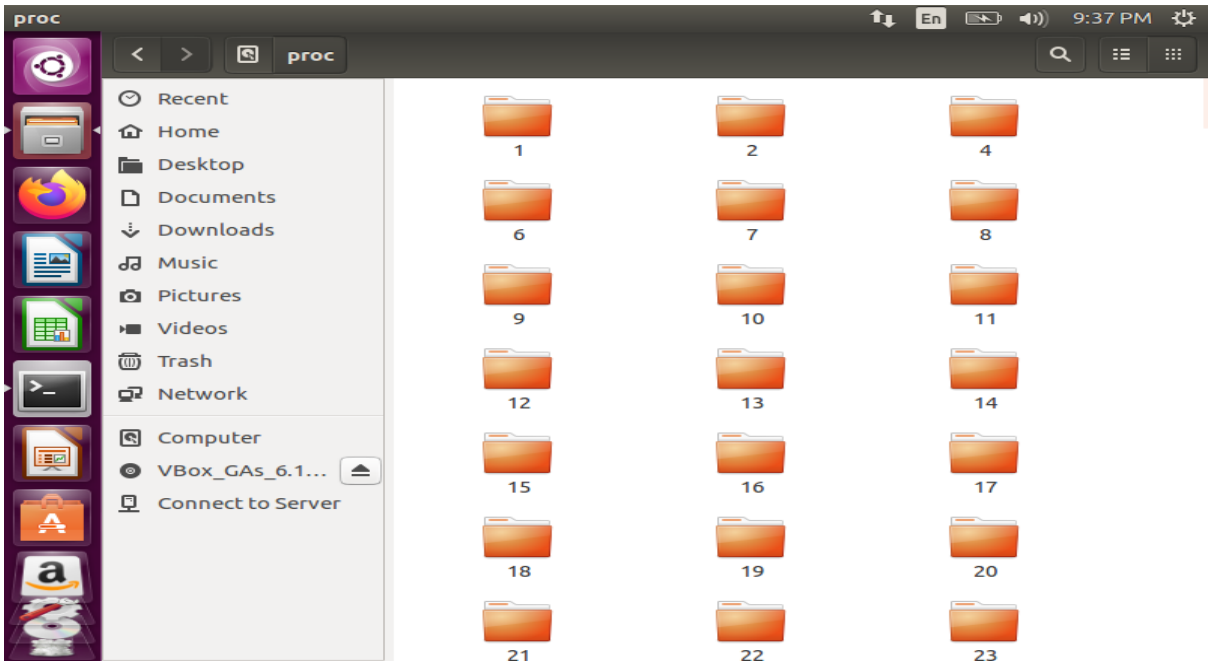
14. /usr : Secondary hierarchy for read-only user data; contains the majority of (multi-)user utilities and applications.

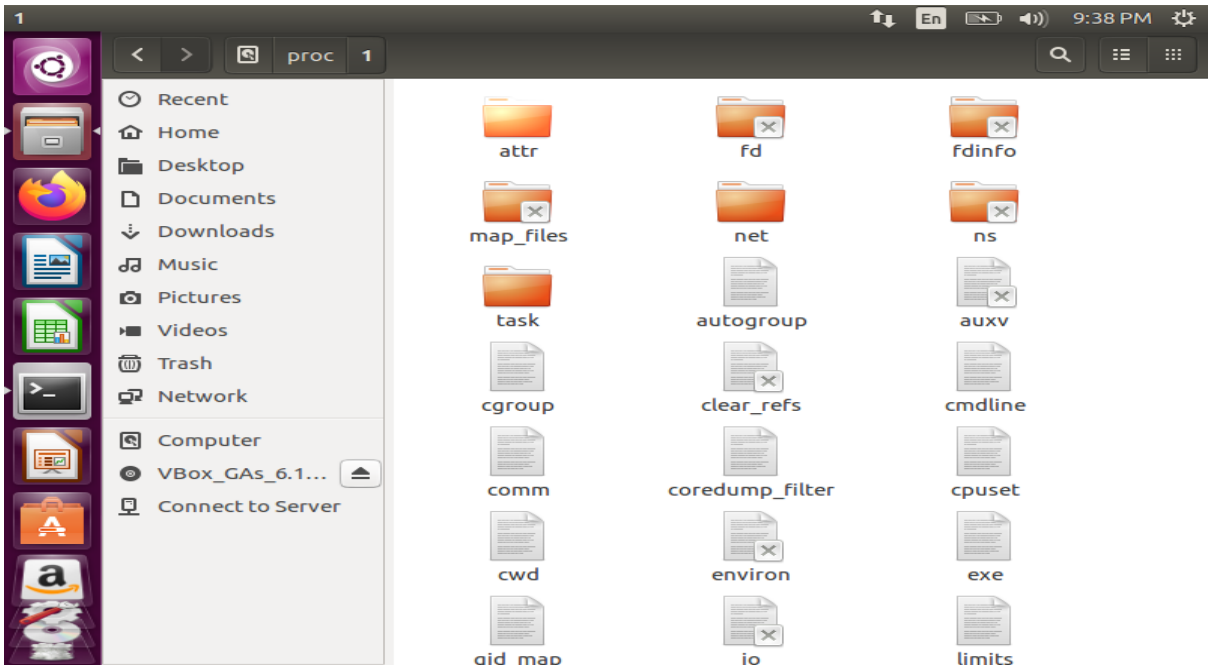
- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If you can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
- /usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2
- /usr/src holds the Linux kernel sources, header-files and documentation.



15. /proc : Virtual filesystem providing process and kernel information as files. In Linux, corresponds to a procfs mount. Generally automatically generated and populated by the system, on the fly.

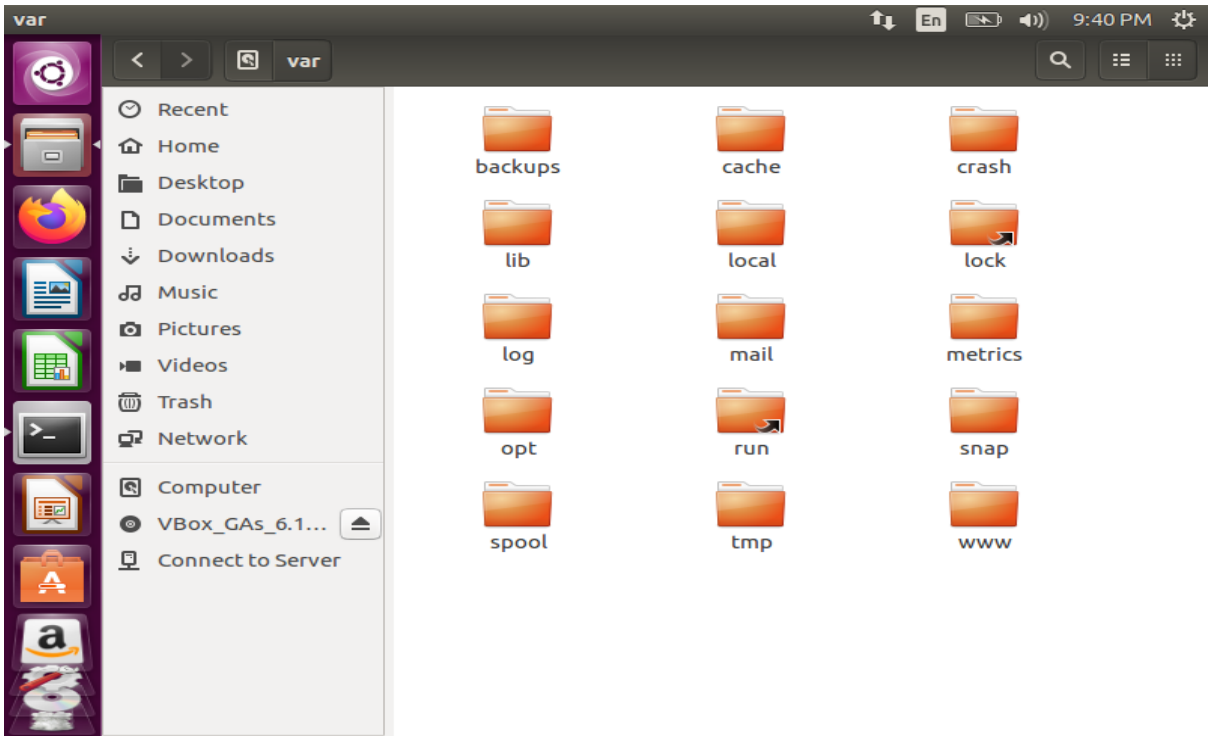
- Contains information about system process.
- This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual filesystem with text information about system resources. For example: /proc/uptime





16. /var – Variable Files

- var stands for variable files.
- Content of the files that are expected to grow can be found under this directory.
- This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);



FILE AND DEVICE PERMISSIONS

Although there are already a lot of good security features built into Linux-based systems, one very important potential vulnerability can exist when local access is granted that is file permission-based issues resulting from a user not assigning the correct permissions to files and directories. So based upon the need for proper permissions, I will go over the ways to assign permissions and show you some examples where modification may be necessary.

Ownership of Linux files

Every file and directory on your Unix/Linux system is assigned 3 types of owner, given below.

User

A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.

Group

A user- group can contain multiple users. All users belonging to a group will have the same Linux group permissions access to the file. Suppose you have a project where a number of people require access to a file. Instead of manually assigning permissions to each user, you could add all users to a group, and assign group permission to file such that only this group members and no one else can read or modify the files.

Other

Any other user who has access to a file. This person has neither created the file, nor he belongs to a usergroup who could own the file. Practically, it means everybody else. Hence, when you set the permission for others, it is also referred as set permissions for the world.

Permissions

Every file and directory in your UNIX/Linux system has following 3 permissions defined for all the 3 owners discussed above.

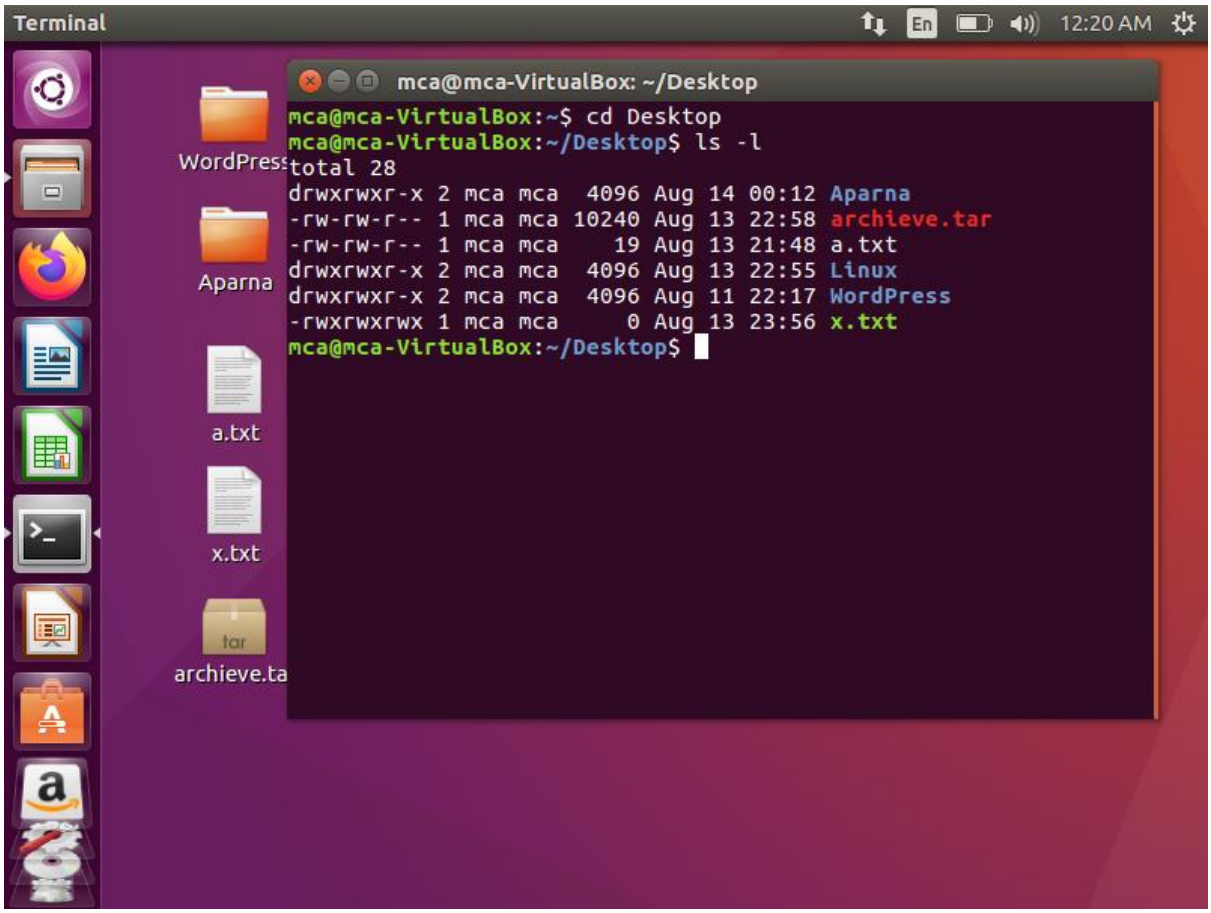
- Read: This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.

- Write: The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory. Consider a scenario where you have to write permission on file but do not have write permission on the directory where the file is stored. You will be able to modify the file contents. But you will not be able to rename, move or remove the file from the directory.

- Execute: In Windows, an executable program usually has an extension ".exe" and which you can easily run. In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code(provided read & write permissions are set), but not run it.

Let's see file permissions in Linux with examples:

ls -l used to list information about files and directories within the file system.



If the first '-' implies that we have selected a file. Else, if it were a directory, d would have been shown.

The characters are pretty easy to remember.

r read permission
w write permission
x execute permission
- no permission

The first part of the code is 'rw-'. This suggests that the owner 'Home' can:

- Read the file
- Write or edit the file
- He cannot execute the file since the execute bit is set to '-'.

The second part is 'rw-'. It for the user group 'Home' and group-members can:

- Read the file
- Write or edit the file

The third part is for the world which means any user. It says 'r--'. This means the user can only:

- Read the file

Changing file/directory permissions with 'chmod' command

We can use the 'chmod' command which stands for 'change mode'. Using the command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the world.

Syntax: *chmod permissions filename*

There are 2 ways to use the command:

1. Absolute mode
2. Symbolic mode

Absolute (Numeric) Mode

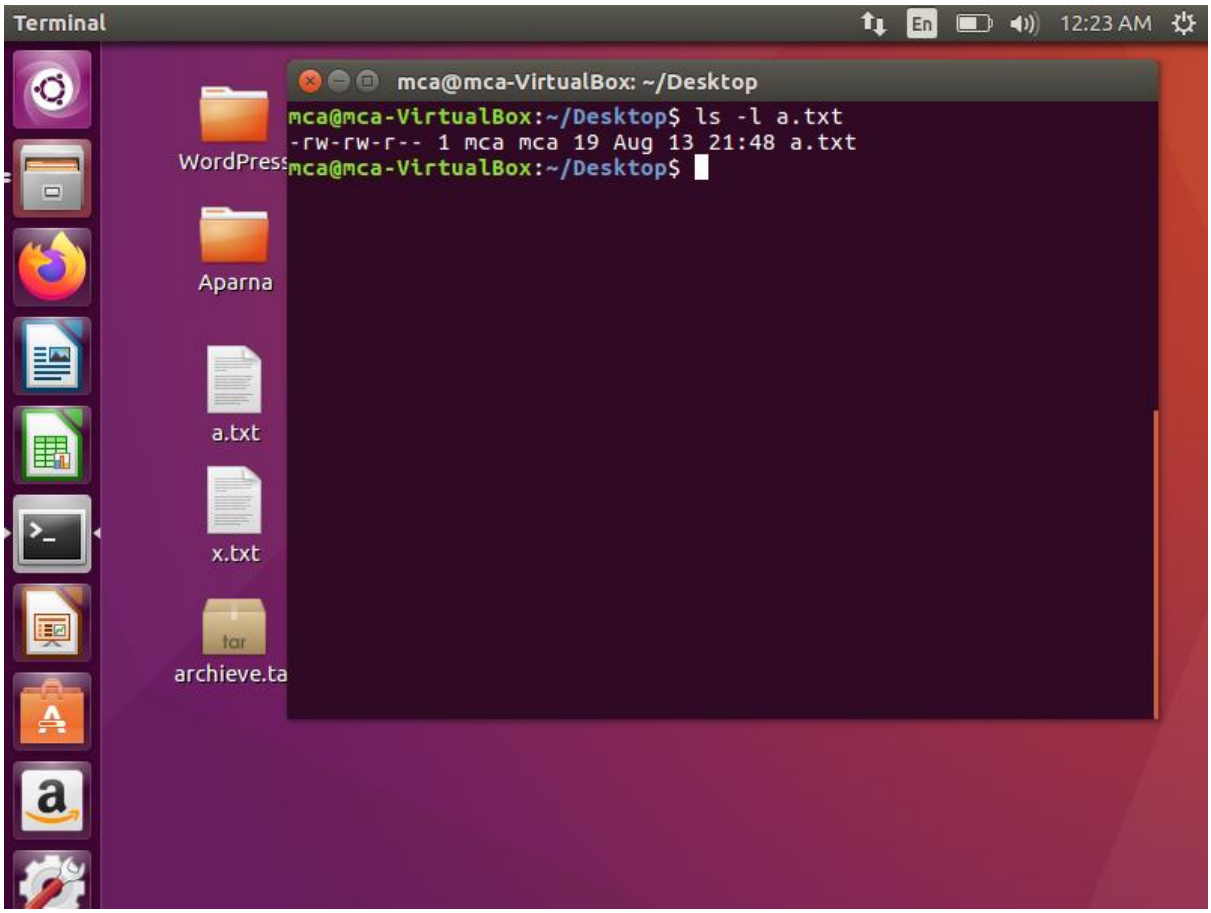
In this mode, file permissions are not represented as characters but a three-digit octal number.

The table below gives numbers for all for permissions types.\

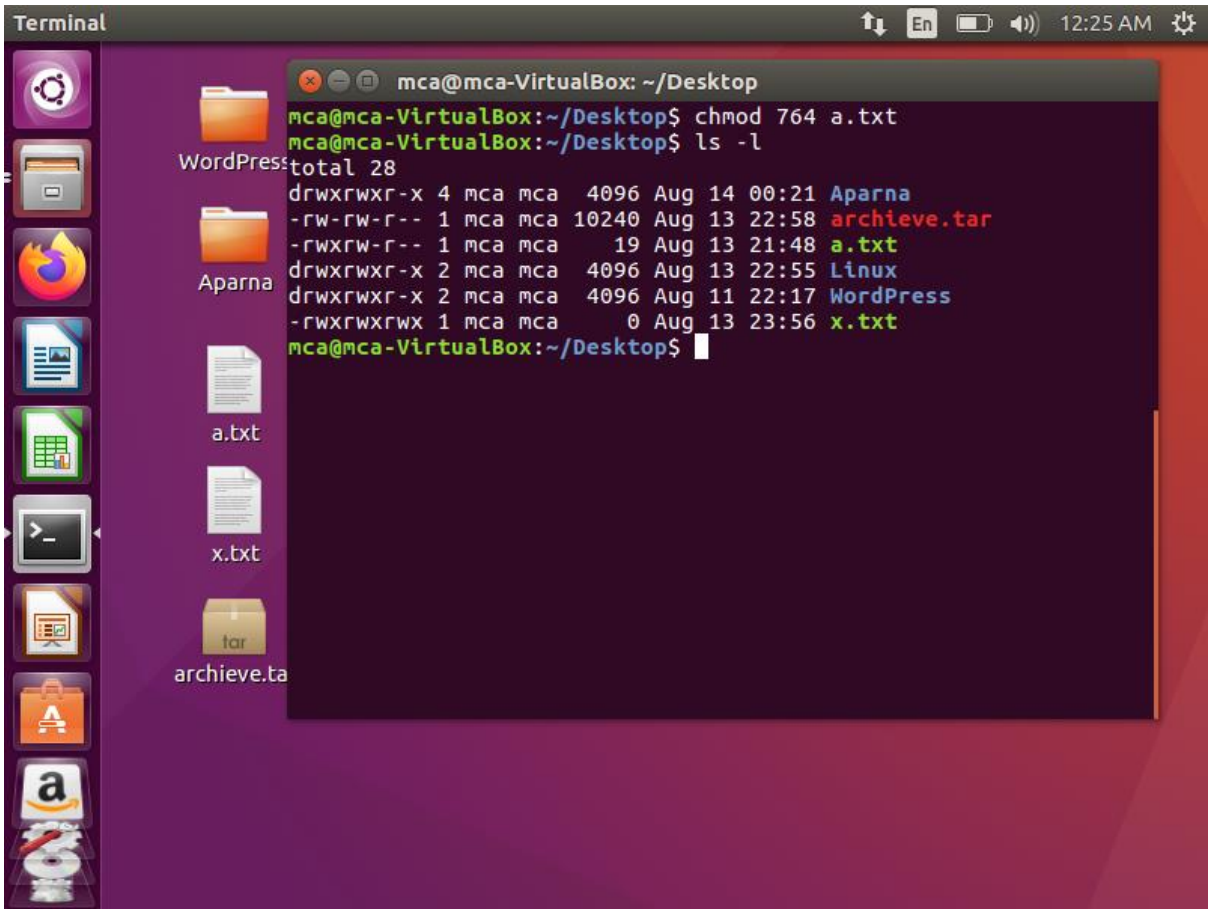
Number	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read +Write	rw-
7	Read + Write +Execute	rwx

Let's see the chmod permissions command in action.

1. Checking current file permissions



2. chmod 764 and checking file permission again



In the above-given terminal window, we have changed the permissions of the file 'sample to '764'.

'764' absolute code says the following:

Owner can read, write and execute , Usergroup can read and write , World can only read

This is shown as '-rwxrw-r—

Symbolic Mode

In the Absolute mode, you change permissions for all 3 owners. In the symbolic mode, you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the Unix file permissions.

Operator	Description
+	Adds a permission to a file or directory
-	Removes the permission

=	Sets the permission and overrides the permissions set earlier.

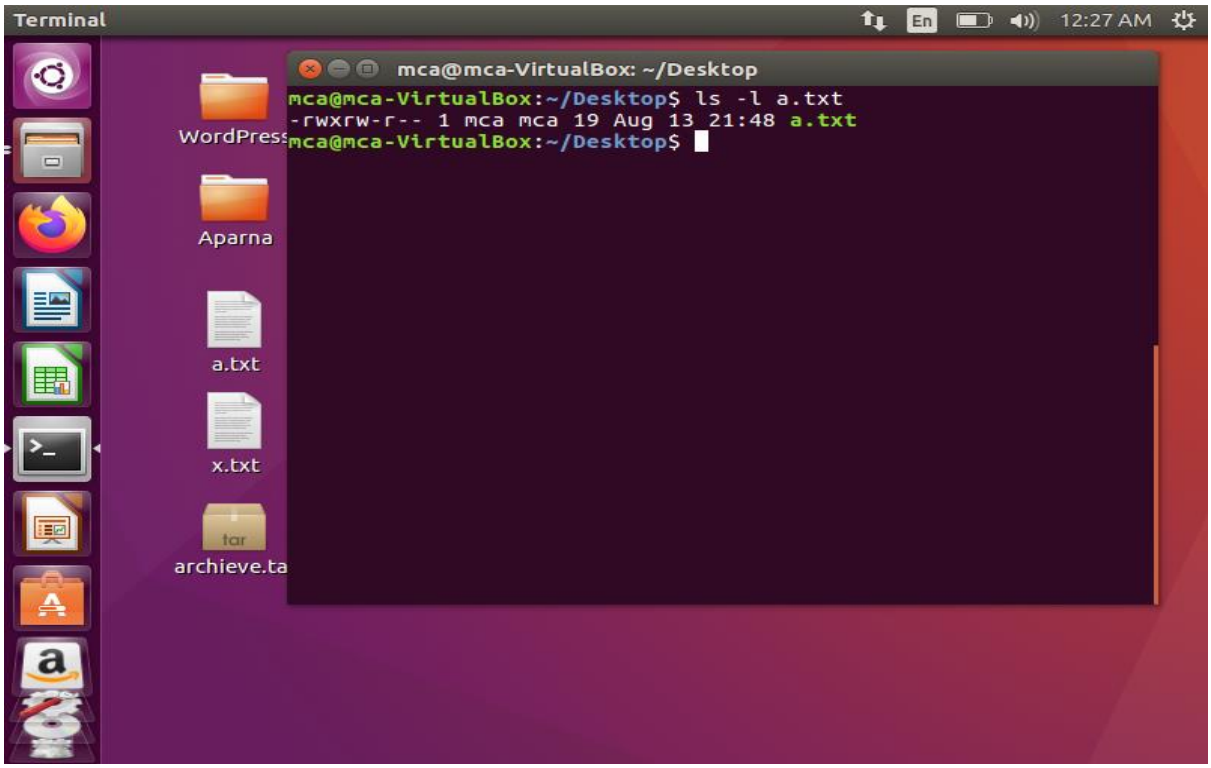
The various owners are represented as:

User Denotations:

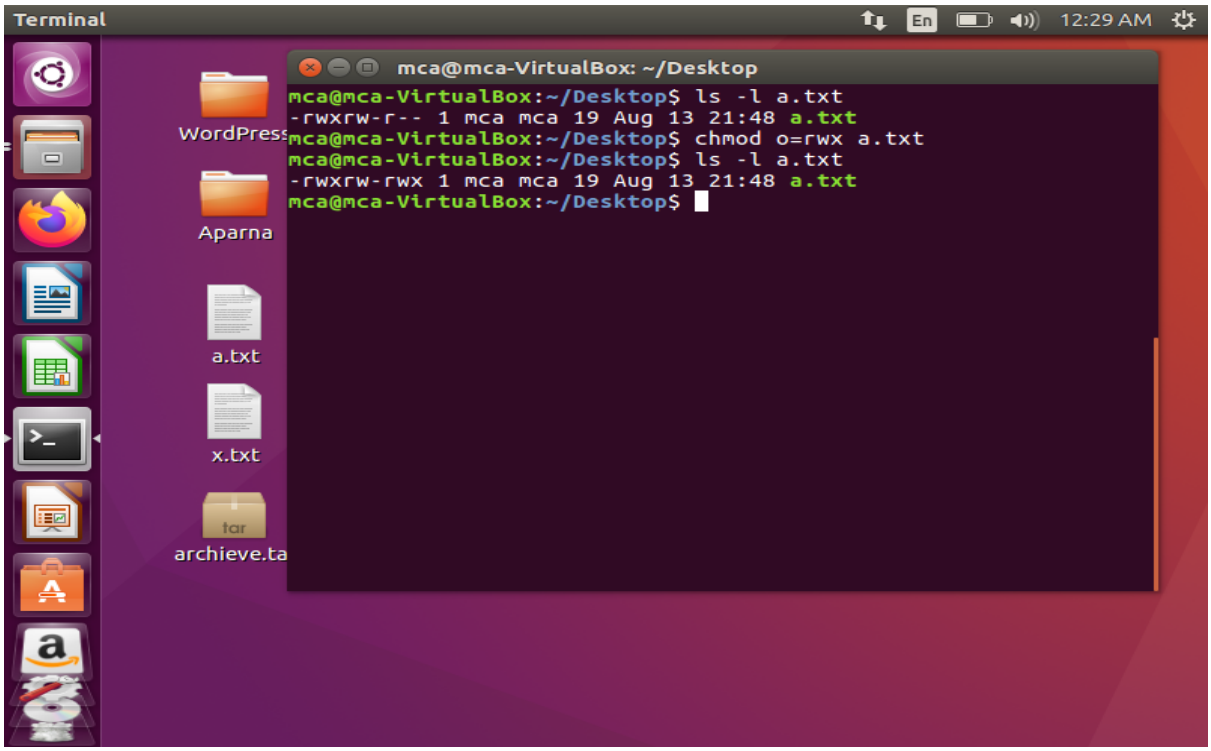
u	user/owner
g	group
o	other
a	all

We will not be using permissions in numbers like 755 but characters like rwx. Let's look into an example:

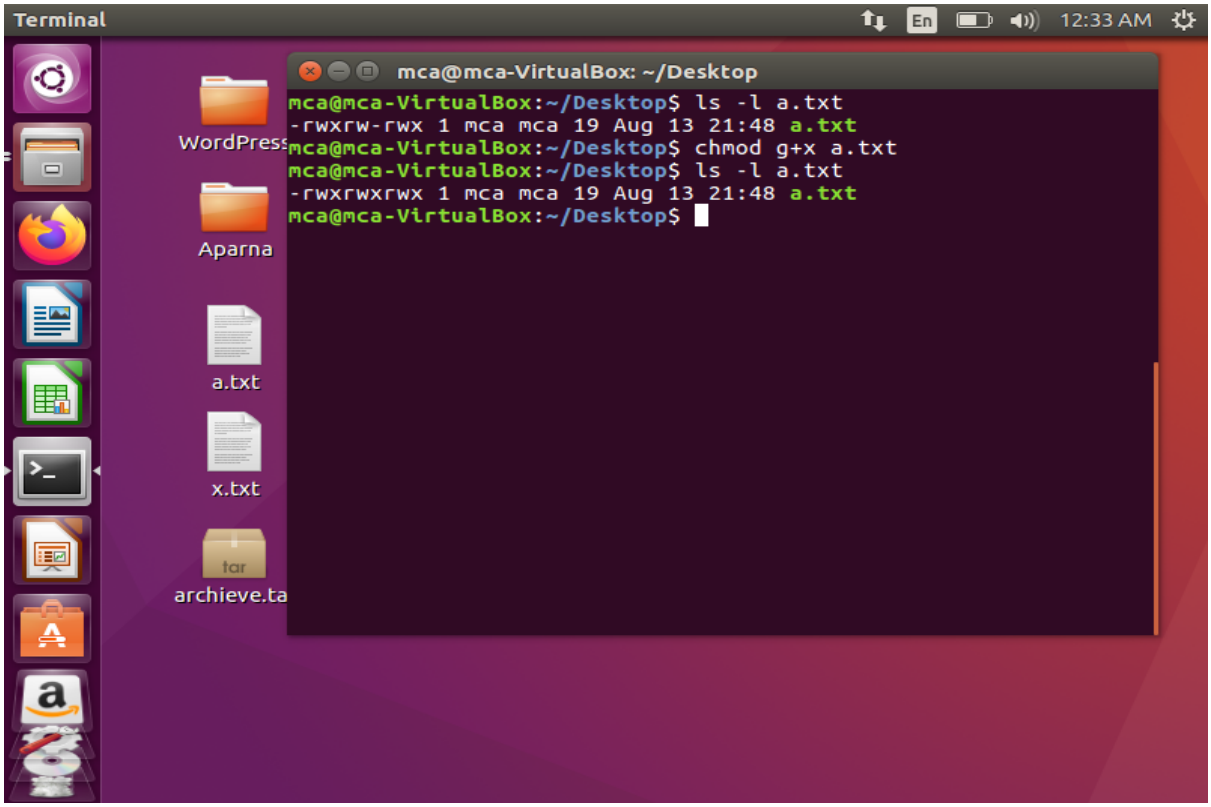
1. Current file permissions



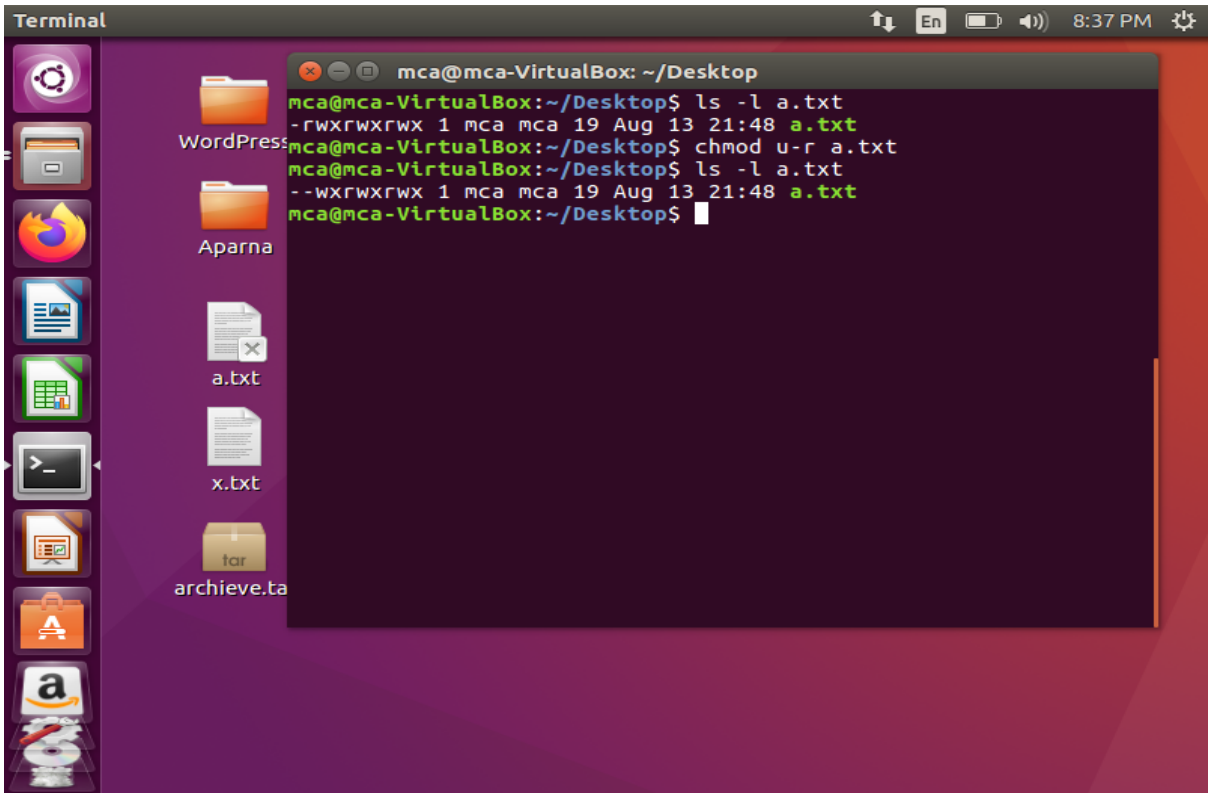
2. Setting permission to ‘other’ users



3. Adding ‘execute’ permission to usergroups



4. Removing ‘read’ permission for ‘user’



/etc : Host-specific system configuration

Purpose

The /etc hierarchy contains configuration files. A "configuration file" is a local file used to control the operation of a program; it must be static and cannot be an executable binary. It is recommended that files be stored in subdirectories of /etc rather than directly in /etc.

Requirements

No binaries may be located under /etc. The following directories, or symbolic links to directories are required in /etc:

Directory	Description
opt	Configuration for /opt

Specific Options

The following directories, or symbolic links to directories must be in /etc, if the corresponding subsystem is installed:

Directory	Description
X11	Configuration for the X Window system (optional)
sgml	Configuration for SGML (optional)
xml	Configuration for XML (optional)

The following files, or symbolic links to files, must be in /etc if the corresponding subsystem is installed:

File	Description
csh.login	Systemwide initialization file for C shell logins (optional)
exports	NFS filesystem access control list (optional)
fstab	Static information about filesystems (optional)
ftpusers	FTP daemon user access control list (optional)
gateways	File which lists gateways for routed (optional)
gettydefs	Speed and terminal settings used by getty (optional)
group	User group file (optional)
host.conf	Resolver configuration file (optional)
hosts	Static information about host names (optional)
hosts.allow	Host access file for TCP wrappers (optional)
hosts.deny	Host access file for TCP wrappers (optional)
hosts.equiv	List of trusted hosts for rlogin, rsh, rcp (optional)
hosts.lpd	List of trusted hosts for lpd (optional)
inetd.conf	Configuration file for inetd (optional)
inittab	Configuration file for init (optional)
issue	Pre-login message and identification file (optional)
ld.so.conf	List of extra directories to search for shared libraries (optional)
motd	Post-login message of the day file (optional)
mtab	Dynamic information about filesystems (optional)

File	Description
mtools.conf	Configuration file for mtools (optional)
networks	Static information about network names (optional)
passwd	The password file (optional)
printcap	The lpd printer capability database (optional)
profile	Systemwide initialization file for sh shell logins (optional)
protocols	IP protocol listing (optional)
resolv.conf	Resolver configuration file (optional)
rpc	RPC protocol listing (optional)
securetty	TTY access control for root login (optional)
services	Port names for network services (optional)
shells	Pathnames of valid login shells (optional)
syslog.conf	Configuration file for syslogd (optional)

Linux log files

Log files are the records that Linux stores for administrators to keep track and monitor important events about the server, kernel, services, and applications running on it. In this post, we'll go over the top Linux log files server administrators should monitor.

Log files are a set of records that Linux maintains for the administrators to keep track of important events. They contain messages about the server, including the kernel, services and applications running on it.

Linux provides a centralized repository of log files that can be located under the /var/log directory.

The log files generated in a Linux environment can typically be classified into four different categories:

- Application Logs
- Event Logs
- Service Logs
- System Logs\

Common Linux log files names and usage

/var/log/messages : General message and system related stuff
/var/log/auth.log : Authenication logs

/var/log/kern.log : Kernel logs
/var/log/cron.log : Crond logs (cron job)
/var/log/maillog : Mail server logs
/var/log/qmail/ : Qmail log directory (more files inside this directory)
/var/log/httpd/ : Apache access and error logs directory
/var/log/lighttpd/ : Lighttpd access and error logs directory
/var/log/nginx/ : Nginx access and error logs directory
/var/log/apt/ : Apt/apt-get command history and logs directory
/var/log/boot.log : System boot log
/var/log/mysqld.log : MySQL database server log file
/var/log/secure or /var/log/auth.log : Authentication log
/var/log/utmp or /var/log/wtmp : Login records file
/var/log/yum.log or /var/log/dnf.log : Yum/Dnf command log file.