**Lab Report No: 04**

**Name of the lab report: File operation and permission.**

Name : Binodon

ID NO : IT-17046

**Objective:** The file system is the most obvious aspect of any OS. A method had to be devised to protect the users from each other.. By this process any user can implement the method for storage and access to data as well as programs of the operating system. In this report paper we know a lot about File operating process on Linux OS.

**Q.1 What is File Operation and File Permission in Linux Operating System?**

**Answer:**

**File Operation**

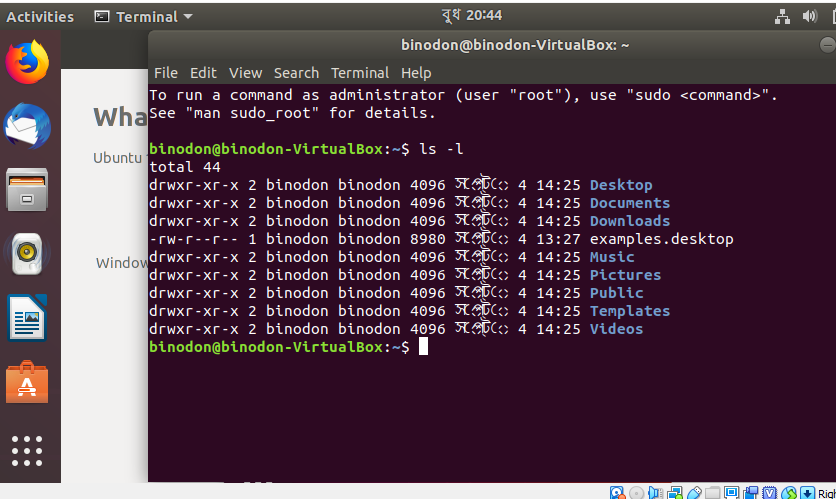
various operations a driver can perform on the devices it manages. An open device is identified internally by a file structure, and the kernel uses the file operations structure to access the driver’s functions. The structure, defined in is an array of function pointers. Each file is associated with its own set of functions (by including a field called that points to a file operations structure). The operations are mostly in charge of implementing the system calls and are thus named open*,* read*,* and so on. We can consider the file to be an “object” and the functions operating on it to be its “methods,” using object-oriented programming terminology to denote actions declared by an object to act on itself. This is the first sign of object-oriented programming we see in the Linux kernel.

**File permission**

The multi-user capability of Unix-like systems is a feature that is deeply ingrained into the design of the operating system. If you remember the environment in which Unix was created, this makes perfect sense. Years ago before computers were "personal," they were large, expensive, and centralized. A typical university computer system consisted of a large mainframe computer located in some building on campus and *terminals* were located throughout the campus, each connected to the large central computer. The computer would support many users at the same time.

In order to make this practical, a method had to be devised to protect the users from each other. After all, you could not allow the actions of one user to crash the computer, nor could you allow one user to interfere with the files belonging to another user.

**For example:**



Here the first column shows the file permissions . The third column shows the user owner of the file . The last column shows the group owner of the file.

**Q2. Implementation of File Operation and File Permission.**

**Answer:**

Permissions

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

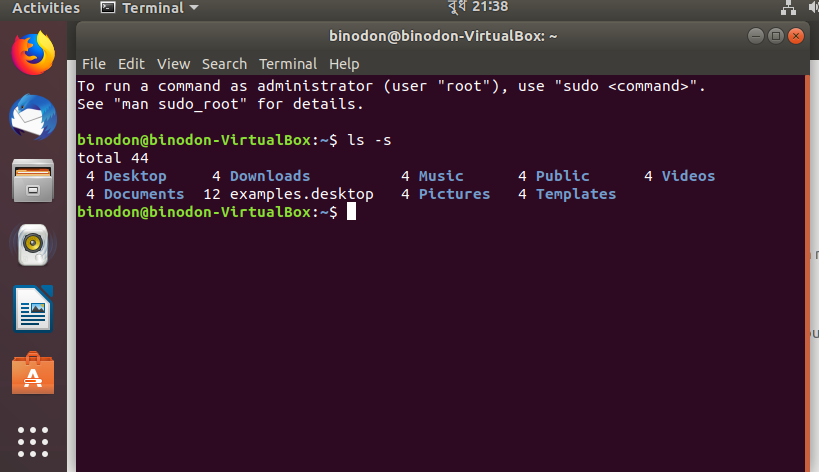
**Read:** This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.

**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

Numerous on-disk and in-memory configurations and structures are being used for implementing a file system. These structures differ based on the operating system and the file system but applying some general principles. Here they are portrayed below :A boot control block usually contains the information required by the system for booting an operating system from that volume. When the disks do not contain any operating system, this block can be treated as empty. This is typically the first chunk of a volume. In UFS, this is termed as the boot block; in NTFS, it is the partition boot sector .A volume control block holds volume or the partition details, such as the number of blocks in the partition, size of the blocks or chunks, free-block count along with free-block pointers. In UFS, it is termed as superblock; in NTFS, it is stored in the master file table .A directory structure per file system is required for organizing the files. In UFS, it held the file names and associated numbers. In NTFS, it gets stored in the master file table .The FCB contains many details regarding any file which includes file permissions, ownership; the size of file and location of data blocks. In UFS, it is called the . In NTFS, this information gets stored within the master file table that uses a relational database (RDBM) structure, using a row per file. Linux is a clone of UNIX, the **multi-user operating system** which can be accessed by many users simultaneously. Linux can also be used in mainframes and servers without any modifications. But this raises security concerns as an unsolicited or **malign user** can **corrupt, change or remove crucial data.** For effective security, Linux divides authorization into 2 levels.

Example:



**Permission Groups**

Each file and directory has three user based permission groups:

**Owner:** The Owner 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.

**All user:** The All Users permissions apply to all other users on the system, this is the permission group that you want to watch the most.

**Conclusion:**

In this lab we come to know about file operation and permission. And run file operation command in linux.By this process we can use any disk as big memory block. It's also possible to make an operating system without files. To do this lab we didnot face any problem.