**TYPES OF LINUX FILE SYSTEM**

* **Ext, Ext2, Ext3 and Ext4 file system**

The file system Ext stands for **Extended File System**. It was primarily developed for **MINIX OS**. The Ext file system is an older version, and is no longer used due to some limitations.

**Ext2** is the first Linux file system that allows managing two terabytes of data. Ext3 is developed through Ext2; it is an upgraded version of Ext2 and contains backward compatibility. The major drawback of Ext3 is that it does not support servers because this file system does not support file recovery and disk snapshot.

**Ext4** file system is the faster file system among all the Ext file systems. It is a very compatible option for the SSD (solid-state drive) disks, and it is the default file system in Linux distribution.

* **JFS File System**

JFS stands for **Journaled File System**, and it is developed by **IBM for AIX Unix.** It is an alternative to the Ext file system. It can also be used in place of Ext4, where stability is needed with few resources. It is a handy file system when [CPU](https://www.javatpoint.com/cpu-full-form) power is limited.

* **ReiserFS File System**

ReiserFS is an alternative to the Ext3 file system. It has improved performance and advanced features. In the earlier time, the ReiserFS was used as the default file system in SUSE Linux, but later it has changed some policies, so SUSE returned to Ext3. This file system dynamically supports the file extension, but it has some drawbacks in performance.

* **XFS File System**

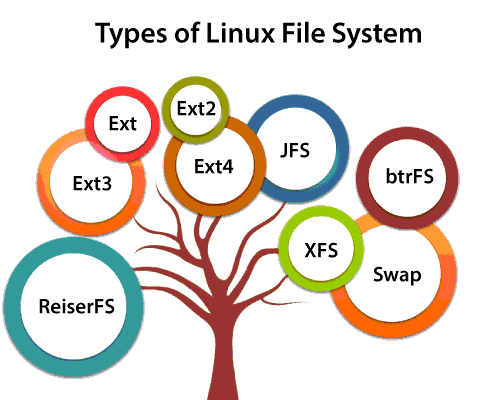
XFS file system was considered as high-speed JFS, which is developed for parallel I/O processing. NASA still using this file system with its high storage server (300+ Terabyte server).

* **Btrfs File System**

Btrfs stands for the **B tree file system**. It is used for fault tolerance, repair system, fun administration, extensive storage configuration, and more. It is not a good suit for the production system.

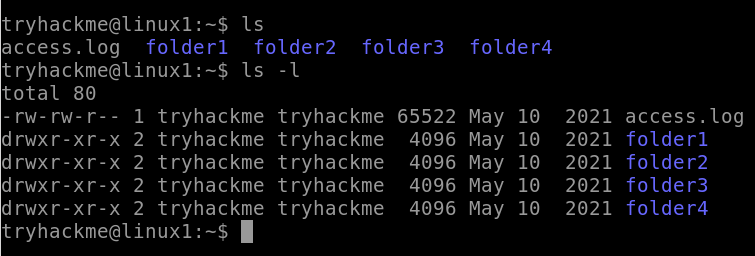
* **Swap File System**

The swap file system is used for memory paging in Linux operating system during the system hibernation. A system that never goes in hibernate state is required to have swap space equal to its [RAM](https://www.javatpoint.com/ram-full-form) size.



**PERMISSIONS IN LINUX**

Linux is a multi-user operating system, so it has security to prevent people from accessing each other’s confidential files. When you execute a “ls” command, you are not given any information about the security of the files, because by default “ls” only lists the names of files. You can get more information by using an “option” with the “ls” command. All options start with a ‘-‘. For example, to 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 a long format. For example



There’s a lot of information in those lines.

* The first character = ‘-‘, which means it’s a file

‘d’, which means it’s a directory.

* The next nine characters = (rw-r–r–) show the security
* The next column shows the owner of the file. (Here it is `root`)
* The next column shows the group owner of the file. (Here it is `root` which has 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.
* Last Column = File\_name or Directory\_name. (For example, here are: prac, snap, test, example)

Security permissions in Linux

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, and execute.

|  |  |
| --- | --- |
| Letters | Definition |
| ‘r’ | “read” the file’s contents. |
| ‘w’ | “write”, or modify, the file’s contents. |
| ‘x’ | “execute” the file. This permission is given only if the file is a program. |

Symbols: `+`, `-` and `=`

|  |  |
| --- | --- |
| Operators | Definition |
| `+` | Add permissions |
| `-` | Remove permissions |
| `=` | Set the permissions to the specified values |

User, group, and others

|  |  |  |
| --- | --- | --- |
| Reference | Class | Description |
| `u` | user | The user permissions apply only to the owner of the file or directory, they will not impact the actions of other users. |
| `g` | group | The group permissions apply only to the group that has been assigned to the file or directory, they will not affect the actions of other users. |
| `o` | others | The other permissions apply to all other users on the system, this is the permission group that you want to watch the most. |
| `a` | All three | All three (owner, groups, others) |

Reading the security permissions

For example: “rw- r-x r–“

* “rw-“: the first three characters `rw-`. This means that the owner of the file can “read” it (look at its contents) and “write” it (modify its contents). we cannot execute it because it is not a program but a text file.
* “r-x”: the second set of three characters “r-x”. This means that the members of the group can only read and execute the files.
* “r–“: The final three characters “r–” show the permissions allowed to other users who have a UserID on this Linux system. This means anyone in our Linux world can read but cannot modify or execute the files’ contents.

Changing security permissions

The command you use to change the security permissions on files is called “chmod“, which stands for “change mode” because the nine security characters are collectively called the security “mode” of the file.

For example, if you want to give “execute” permission to the world (“other”) for file “xyz.txt”, you will start by typing.

**chmod o**

Now you would type a ‘+’ to say that you are “adding” permission.

**chmod o+**

Then you would type an ‘x’ to say that you are adding “execute” permission.

**chmod o+x**

Finally, specify which file you are changing.

**chmod o+x xyz.txt**