**Comparison between Windows, Linux and MacOS**

**Introduction**

An Operating System (OS) is an interface between a 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. We are going to look at the difference between the big three operating systems, namely, Windows, MacOS and Linux.

**Use and cost**

Let's start with the key questions I ask when anyone comes to me for advice about computer purchasing:

* What do you want to do with it?
* How much do you want to spend?
* Which of those two (use versus spend) is more important?

Windows typically comes with new PCs. If you have an old PC, it almost certainly comes with Windows. Pretty much the only time you're going to pay to buy Windows is if you're building one of your own, and we'll address that later. Macs always come with MacOS, so there's no cost there. And, of course, Linux is free. The much bigger cost for all these environments will be your ongoing maintenance expenses.

If you have a budget and require a fast-blazing machine, MacOS is your best option. For limited budget and simple interface, i would suggest windows and lastly for free open-source OS but with complexity, use Linux. As simple as that, you have made your choice.

**Windows**

**Directory Structure**

The Windows OS often uses the C drive as the main drive of the computer. Every folder, file and programs are stored inside this drive, and the terminal may look like this C:\>. A user folder is created inside the C drive, meaning every user will have its own directories. Windows also uses backslashes compared to other operating systems.

**Naming Convention**

All file systems follow the same general naming conventions for an individual file: a base file name and an optional extension, separated by a period. However, each file system, such as NTFS, CDFS, exFAT, UDFS, FAT, and FAT32, can have specific and differing rules about the formation of the individual components in the path to a directory or file.

The following fundamental rules enable applications to create and process valid names for files and directories, regardless of the file system:

* Use a period to separate the base file name from the extension in the name of a directory or file.
* Use a backslash (\) to separate the *components* of a *path, for example C:\*.Does not assume case sensitivity. Oscar and OSCAR are not the same.
* Files have a maximum length limit. Cannot have a long-named file.

**Linux**

**Directory Structure**

Linux has the most complex file system and command interface. Just like Windows, Linux creates it own main drive, they call it the root directory. All the files and folders are stored in this directory. What makes the file system even more complex is that Linux has different distributions including Fedora, Ubuntu, Debian, Red Hat and many others. You have to adapt to the structure based on what you choose. Uses foward-slash implementation just like root:/

**Naming Convention**

Linux file systems (such as ext4, Btrfs, XFS, etc.) follow POSIX-compliant naming and path conventions. These rules apply across most Linux distributions:

* Use a forward slash / to separate components in a path.
* No drive letters — Linux uses a single hierarchical file system tree rooted at /.
* Case-sensitive: File.txt, file.txt, and FILE.TXT are three different files.
* File extensions are optional and not required by the system. A file is identified by its content or permissions, not the extension.
* Hidden files or directories begin with a dot. (e.g., .bashrc, .config).
* There are no restrictions on length in most modern file systems (usually up to 255 characters for file names).

**MacOS**

**Directory Structure**

The macOS file structure is UNIX based operating system. Unlike Windows, macOS does not use drive letters; instead, all files and folders exist within this unified structure. The main user-facing directory is located at /Users, where each user has their own folder (e.g., /Users/alex). This is equivalent to the C:\Users\ directory in Windows. The macOS terminal typically starts in the user’s home directory and might look like this: alex@MacBook ~ %. macOS uses **forward slashes (/)** to separate path components, consistent with UNIX conventions. Mounted drives and external volumes appear under the /Volumes directory rather than having individual drive letters like C:\ or D:\. System files and applications are stored in directories like /System, /Library, and /Applications.

**Naming Convention**

macOS uses Apple File System and supports HFS+. It follows similar UNIX-style naming conventions but with a few Mac-specific behaviors:

* Use a forward slash / to separate path components.
* Case-insensitive by default on APFS and HFS+ volumes, but this can vary:
  + file.txt and File.txt are the same file on default installations.
  + Developers can format volumes to be case-sensitive.
* File extensions are optional and not required. The system may use metadata to identify file types (like MIME types or Uniform Type Identifiers).
* File names can include almost any character, except:
  + Colon: (used internally by macOS)
  + Null character \0
* Slashes / are reserved as path separators, so they cannot be used in file names.

There are many other differences between these operating systems in regard to how they operate and take commands. These are some of the missing differences between Windows, MacOS and Linux which might be helpful when trying to identify the machine’s operating system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Windows** | **Linux** | **macOS** |
| **File System** | NTFS, FAT32, exFAT | ext4, Btrfs, XFS, FAT32, exFAT | APFS, HFS+, exFAT, FAT32 |
| **Executable File Types** | .exe, .msi, .bat | No fixed extensions (e.g., ELF binaries) | No fixed extensions (e.g., Mach-O binaries) |
| **Script Execution** | cmd, PowerShell, .bat | bash, zsh, sh, fish, etc. | zsh, bash, sh, fish, etc. |