**Advanced Linux**

**Assignment**-1

**Title:** Linux File System and Windows File System, Difference

Linux has two kinds of major partitions called data partitions and swap partitions. Because of the existence of swap partitions, you never run out of memory in Linux (like in windows).

In terms of recovery tools, only a limited number of tools can be used on Windows, while there is a large number of UNIX based recovery tools available for Linux file systems.

Both **Windows** and **Linux** organize disk-based files into a hierarchy of directories. Such directories are usually called “**folders**” when viewed in a GUI.

One whole hierarchy is called a “**file system**” on both platforms.

The architecture of Windows and Linux file systems is similar in some points, but diverges greatly when tools are considered. Here’s a quick comparison.

On hardware derived from **Intel** or **IBM** PCs, both **Windows** and **Linux** use the Master Block Record/ Master Boot Record (**MBR**). That is the bit of disk used to boot the operating system and to state whether the disk is partitioned or not. On both **Windows** and **Linux**, it is common to have at most one file system per partition.

That is about the end of the similarity.

Each **Windows** file system gets a drive letter, like “C:”. On **Linux**, each file system gets a device, like /dev/hda1 (“hard disk A part 1”), which is represented as a file. Such a file is a device file (since a disk is a device), hence the “dev” part of the path name. Also, the device file is not an ordinary text file, it is a “**special file**.” Since disks are block devices (unlike a serial mouse), such a file is fully described as a “**block special device file**.” The numbered part of the path can be a little weird to get right; it’s best to be guided by documentation there or extract the right name from a report.

On **Windows**, file systems can be **FAT16**, **FAT32** or **NTFS**, to name a few.

Recall **FAT16** is the ancient standard responsible for **Windows** file names with the “**8.3**” file name length restriction.

On **Linux**, filesystems can be “**minix**,” “**ext**,” or “**ext2**,” to name a few. Also, “**minix**” is an example of the ancient standard responsible for **UNIX** file names once being limited to 14 characters. Linux also has “**msdos**” and “**vfat**” file systems for compatibility with **Windows** and **DOS**, plus more.

Windows uses **FORMAT.EXE** to format a disk. Linux uses “**mkfs**” (“make file system”) in various specialist forms.

Each Windows file system has a File Allocation Table (**FAT**, **VFAT**, or similar) that states which disk blocks hold the topmost directory.

On **Linux**, the equivalent on most filesystems is the **superblock**. A **Linux** file system has multiple copies of the **superblock** physically saved on the disk. This provides redundancy in case of a partial disk corruption.

The **superblock** is just about always in memory on **Linux**; that is not the case for ancient **DOS**-like file systems. There are no special restrictions on files placed in the topmost directory on **Linux**, either.

On **Windows**, there is one drive letter per mounted file system: for example, C: for C: and D: for D:. On **Linux** there are no drive letters, so one file system is mounted on “/” and all other file systems are mounted on subdirectories of “/.” This arrangement is like the little-used **MS-DOS** command **SUBST**, or the **NET USE** command that supports **Novell**‘s **NetWare**. The equivalent **Linux** command is “**mount**.”

You can see all this at work on **Linux**, but for some of it you need to be logged in as **root**. Type “**cat /etc/fstab**” to see all the mountable devices, including floppy disks and CD players. Type “**df**” to see the devices currently mounted, and their free space.

In the usual case you can even see the **superblock**: try “**/sbin/dumpe2fs /dev/hda1**” where **hda1** comes from the output of “**df**.” The name “**dumpe2fs**” is a casualty of history; it replaces the older “**dumpfs**.” The information produced is really just for diagnostic purposes.

Mostly file systems work for you silently. If you want to dig into Linux further, then there are plenty of tools that can be used as inspection points.