
# Linux System Programming

# &

# UNIX Internals

# LINUX FILESYSTEM HIERARCHY

## What is file :

In UNIX or any OS file is simply a uniform unformatted stream of bytes, which is stored on our harddisk. From user point of view file contains the data that is Human readable format but from OS point of view it just a sequence of "1000101111011……". File is considered as unformatted uniform stream of bytes.

## What File System:

File system means making environment for storing our data our file on hard disk in such way that the application can retrieve it back from hard disk and display it to end user in understandable format.

Various file systems are

NTFS (New Technology File System), FAT32(File Allocation Table 32) which are windows file system.

ext3 (Extended 3) for Linux distros (i.e. Linux versions )

## Types of Files:

In UNIX or any IX there is a statement "Files have spaces And Processes Has Lives!"

In UNIX everything that is storable is file for e.g. Text File, USB Device, Network Card, Hard disk partition, directory (Folder in Windows), Devices, even Head Phone connected to machine is also a file.

But all these things are distinguishable, in two types in regular files and special files, and are simply distinguished as files those are not special files are regular files and others are special files which are Directory file and device file.

ROOT directory "/root":

As mentioned above types of file are Regular files and Special files.

One of the types of special file is "Directory file" which most important from OS point of view.

Structure for Directory file is as follows:

```
Struct DIR
{
    Name[];
    InodeNumber;
};
```

From this we introduce a field as "INODE", which is simply a number stored in the devices DILB(Disk Inode List Block) which uniquely identifies the data blocks on hard disk allocated for that file.

So we get an idea how particular file is retrieved form the hard disk. For that purpose "namei" algorithm is used.

From this we get that every file has an "InodeNumber" associated with that particular file and that entry is stored in a directory file which is the folder/Directory for that file.

But problem is arises when that file is also directory then in such case the directory file that stores the entry for this file (Directory File) is called as "Parent Directory" for this directory.

So from this scenario we get that for every file there is some parent associated with the file which stores entry for that file.

But in this case there should be a certain start point where this everything is started and in case of any IX it is "/" i.e. "root".

"/_<-inux -> It's all starts here."

This is the only file in IX which is hard coded and all the remaining files are created at runtime i.e everything in IX is starting from "root" i.e. "/".

Similarly in windows "My Computer" is the starting for the file system.

Now our file system is started but there no anything in IX world that can be put in a single file so that definitely there are multiple files in the file system exist. And we have to arrange and store them in manner such that they can satisfy the rules of hierarchical file system.

So LINUX can divide or classify them in several directories which are named as it suits for the purpose that files does.

"/bin" Directory:

bin directory contains several useful commands that are of use to both the system administrator as well as non-privileged users. It usually contains the shells like bash, csh, etc.... and commonly used commands like cp, mv, rm, cat, ls.

"/boot" Directory:

/boot directory stores data that is used before the kernel begins executing user-mode programs. This may include redundant (back-up) master boot records, sector/system map files, the kernel and other important boot files and data that is not directly edited by hand.

/boot/boot.0300

Backup master boot record.

/boot/boot.b

This is installed as the basic boot sector. In the case of most modern distributions it is actually a symbolic link to one of four files /boot/boot-bmp.b, /boot/boot-menu.b, /boot/boot-text.b, /boot/boot-compat.b which allow a user to change the boot-up schema so that it utilises a splash screen, a simple menu, a text based interface or a minimal boot loader to ensure compatibility respectively. In each case re-installation of lilo is necessary in order to complete the changes. To change the actual 'boot-logo' you can either use utilities such as fblogo or the more refined bootsplash.

/boot/chain.b

Used to boot non-Linux operating systems.

/boot/config-kernel-version

Installed kernel configuration. This file is most useful when compiling kernels on other systems or device modules.

/boot/os2_d.b

Used to boot to the 0S/2 operating system.

/boot/map

Contains the location of the kernel.

/boot/vmlinuz, /boot/vmlinuz-kernel-version

Normally the kernel or symbolic link to the kernel.

/boot/grub

This subdirectory contains the GRUB configuration files including boot-up images and sounds. GRUB is the GNU GRand Unified Bootloader, a project which intends to solve all bootup problems once and for all. One of the most interesting features, is that you don't have to install a new partition or kernel, you can change all parameters at boot time via the GRUB Console, since it knows about the filesystems.

/boot/grub/device.map

Maps devices in /dev to those used by grub. For example, (/dev/fd0) is represented by /dev/fd0 and (hd0, 4) is referenced by /dev/hda5.

/boot/grub/grub.conf, /boot/grub/menu.lst

Grub configuration file.

/boot/grub/messages

Grub boot-up welcome message.

/boot/grub/splash.xpm.gz

Grub boot-up background image.

"/dev" Directory:

/dev is the location of special or device files. It is a very interesting directory that highlights one important aspect of the Linux filesystem - everything is a file or a directory. Look through this directory and you should hopefully see hda1, hda2 etc.... which represent the various partitions on the first master drive of the system. /dev/cdrom and /dev/fd0 represent your CD-ROM drive and your floppy drive. This may seem strange but it will make sense if you compare the characteristics of files to that of your hardware. Both can be read from and written to. Take /dev/dsp, for instance. This file represents your speaker device. Any data written to this file will be re-directed to your speaker. If you try 'cat /boot/vmlinuz> /dev/dsp' (on a properly configured system) you should hear some sound on the speaker. That's the sound of your kernel! A file sent to /dev/lp0 gets printed. Sending data to and reading from /dev/ttyS0 will allow you to communicate with a device attached there - for instance, your modem.

The majority of devices are either block or character devices; however other types of devices exist and can be created. In general, 'block devices' are devices that store or hold data, 'character devices' can be thought of as devices that transmit or transfer data. For example, diskette drives, hard drives and CD-ROM drives are all block devices while serial ports, mice and parallel printer ports are all character devices.

"/etc" Directory:

This directory contains all system related configuration files in here or in its sub-directories. A "configuration file" is defined as a local file used to control the operation of a program; it must be static and cannot be an executable binary. For this reason, it's a good idea to backup this directory regularly. It will definitely save you a lot of re-configuration later if you re-install or lose your current installation.

"/home" Directory:

Linux is a multi-user environment so each user is also assigned a specific directory that is accessible only to them and the system administrator. These are the user home directories, which can be found under '/home/$USER' (~/). It is your playground: everything is at your command, you can write files, delete them, install programs, etc.... Your home directory contains your personal configuration files, the so-called dot files (their name is preceded by a dot). Personal configuration files are usually 'hidden', if you want to see them, you either have to turn on the appropriate option in your file manager or run ls with the -a switch. If

there is a conflict between personal and system wide configuration files, the settings in the personal file will prevail.

Dotfiles most likely to be altered by the end user are probably your .xsession and .bashrc files. The configuration files for X and Bash respectively. They allow you to be able to change the window manager to be startup upon login and also aliases, user-specified commands and environment variables respectively. Almost always when a user is created their dotfiles will be taken from the /etc/skel directory where system administrators place a sample file that user's can modify to their hearts content.

/home can get quite large and can be used for storing downloads, compiling, installing and running programs, your mail, your collection of image or sound files etc.

**"/initrd" Directory:**

This directory contains files which are needed to boot(start) the Operating System. Initrd provides the capability to load a RAM disk by the boot
loader. This RAM disk can then be mounted as the root file system and programs can be run from it. Afterwards, a new root file system can be mounted from a different device. The previous root (from initrd) is then moved to a directory and can be subsequently unmounted.

{

initrd is mainly designed to allow system startup to occur in two phases, where the kernel comes up with a minimum set of compiled-in drivers, and where additional modules are loaded from initrd.

}

**"/lib" Directory:**

This directory contains files which useful or required for computer programmer in development of an application. The /lib directory contains kernel modules and those shared library images (the C programming code library) needed to boot the system and run the commands in the root filesystem, ie. bybinaries in /bin and /sbin. Libraries are readily identifiable through their filename extension of *.so. Windows equivalent to a shared library would be a DLL (dynamically linked library) file. They are essential for basic system functionality. Kernel modules (drivers) are in the subdirectory /lib/modules/'kernel-version'. To ensure proper module compilation you should ensure that /lib/modules/'kernel-

version'/kernel/build points to /usr/src/'kernel-version' or ensure that the Makefile knows where the kernel source itself are located.

## "/media" Directory:

This directory contains subdirectories which are used as mount points for
removeable media such as floppy disks, cdroms and zip disks.

The motivation for the creation of this directory has been that historically there have been a number of other different places used to mount removeable
media such as /cdrom, /mnt or /mnt/cdrom. Placing the mount points for all
removeable media directly in the root directory would potentially result in a large number of extra directories in /. Although the use of subdirectories
in /mnt as a mount point has recently been common, it conflicts with a much
older tradition of using /mnt directly as a temporary mount point.

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

floppyFloppy drive (optional)
cdrom CD-ROM drive (optional)
cdrecorder CD writer (optional)
zipZip drive (optional)

On systems where more than one device exists for mounting a certain type of media, mount directories can be created by appending a digit to the name of those available above starting with '0', but the unqualified name must also exist.

A compliant implementation with two CDROM drives might have /media/cdrom0
and /media/cdrom1 with /media/cdrom a symlink to either of these.

## "/mnt" Directory:

This is a generic mount point under which you mount your filesystems or devices. Mounting is the process by which you make a filesystem available to the system. After mounting your files will be accessible under the mount-point. This directory usually contains mount

points or sub-directories where you mount your floppy and your CD. You can also create additional mount-points here if you wish. Standard mount points would include /mnt/cdrom and /mnt/floppy. There is no limitation to creating a mount-point anywhere on your system but by convention and for sheer practicality do not litter your file system with mount-points. It should be noted that some distributions like Debian allocate /floppy and /cdrom as mount points while Redhat and Mandrake puts them in /mnt/floppy and /mnt/cdrom respectively.

Mounting and unmounting

Before one can use a filesystem, it has to be *mounted*. The operating system then does various bookkeeping things to make sure that everything works. Since all files in UNIX are in a single directory tree, the mount operation will make it look like the contents of the new filesystem are the contents of an existing subdirectory in some already mounted filesystem.

```
$mount /dev/hda2 /home
$mount /dev/hda3 /usr
$
```

The **mount** command takes two arguments. The first one is the device file corresponding to the disk or partition containing the filesystem. The second one is the directory below which it will be mounted. After these commands the contents of the two filesystems look just like the contents of the /home and /usr directories, respectively. One would then say that ``/dev/hda2*is mounted on*/home'', and similarly for /usr. To look at either filesystem, one would look at the contents of the directory on which it has been mounted, just as if it were any other directory. Note the difference between the device file, /dev/hda2, and the mounted-on directory, /home. The device file gives access to the raw contents of the disk, the mounted-on directory gives access to the files on the disk. The mounted-on directory is called the *mount point*.

Linux supports many filesystem types. **mount** tries to guess the type of the filesystem. You can also use the -t fstype option to specify the type directly; this is sometimes necessary, since the heuristics **mount** uses do not always work. For example, to mount an MS-DOS floppy, you could use the following command:

```
$mount -t msdos /dev/fd0 /floppy
```

The mounted-on directory need not be empty, although it must exist. Any files in it, however, will be inaccessible by name while the filesystem is

mounted. (Any files that have already been opened will still be accessible. Files that have hard links from other directories can be accessed using those names.) There is no harm done with this, and it can even be useful. For instance, some people like to have /tmp and /var/tmp synonymous, and make /tmpbe a symbolic link to /var/tmp. When the system is booted, before the /varfilesystem is mounted, a /var/tmp directory residing on the root filesystem is used instead. When /var is mounted, it will make the /var/tmp directory on the root filesystem inaccessible. If /var/tmp didn't exist on the root filesystem, it would be impossible to use temporary files before mounting /var.

If you don't intend to write anything to the filesystem, use the -r switch for **mount** to do a *read-only mount*. This will make the kernel stop any attempts at writing to the filesystem, and will also stop the kernel from updating file access times in the inodes. Read-only mounts are necessary for unwritable media, e.g., CD-ROMs.

The alert reader has already noticed a slight logistical problem. How is the first filesystem (called the *root filesystem*, because it contains the root directory) mounted, since it obviously can't be mounted on another filesystem? Well, the answer is that it is done by magic.

For more information, see the kernel source or the Kernel Hackers' Guide.

The root filesystem is magically mounted at boot time, and one can rely on it to always be mounted. If the root filesystem can't be mounted, the system does not boot. The name of the filesystem that is magically mounted as root is either compiled into the kernel, or set using LILO or **rdev**.

The root filesystem is usually first mounted read-only. The startup scripts will then run **fsck** to verify its validity, and if there are no problems, they will *re-mount* it so that writes will also be allowed. **fsck** must not be run on a mounted filesystem, since any changes to the filesystem while **fsck** is running *will* cause trouble. Since the root filesystem is mounted read-only while it is being checked, **fsck** can fix any problems without worry, since the remount operation will flush any metadata that the filesystem keeps in memory.

On many systems there are other filesystems that should also be mounted automatically at boot time. These are specified in the /etc/fstab file; see the fstab man page for details on the format. The details of exactly when the extra filesystems are mounted depend on many factors, and can be configured by each administrator if need be.

When a filesystem no longer needs to be mounted, it can be unmounted with **umount**.

It should of course be **unmount**, but the n mysteriously disappeared in the 70s, and hasn't been seen since. Please return it to Bell Labs, NJ, if you find it.

**umount** takes one argument: either the device file or the mount point. For example, to unmount the directories of the previous example, one could use the commands

```
$umount /dev/hda2
$umount /usr
```

See the man page for further instructions on how to use the command. It is imperative that you always unmount a mounted floppy. *Don't just pop the floppy out of the drive!* Because of disk caching, the data is not necessarily written to the floppy until you unmount it, so removing the floppy from the drive too early might cause the contents to become garbled. If you only read from the floppy, this is not very likely, but if you write, even accidentally, the result may be catastrophic.

Mounting and unmounting requires super user privileges, i.e., only root can do it. The reason for this is that if any user can mount a floppy on any directory, then it is rather easy to create a floppy with, say, a Trojan horse disguised as /bin/sh, or any other often used program. However, it is often necessary to allow users to use floppies, and there are several ways to do this:

- Give the users the root password. This is obviously bad security, but is the easiest solution. It works well if there is no need for security anyway, which is the case on many non-networked, personal systems.
- Use a program such as **sudo** to allow users to use mount. This is still bad security, but doesn't directly give super user privileges to everyone.
- Make the users use **mtools**, a package for manipulating MS-DOS filesystems, without mounting them. This works well if MS-DOS floppies are all that is needed, but is rather awkward otherwise.
- List the floppy devices and their allowable mount points together with the suitable options in /etc/fstab.

The last alternative can be implemented by adding a line like the following to the /etc/fstab file:

```
  /dev/fd0 /floppy
msdosuser,noauto 0 0
```

The columns are: device file to mount, directory to mount on, filesystem type, options, backup frequency (used by **dump**), and **fsck** pass number

(to specify the order in which filesystems should be checked upon boot; 0 means no check).

The noauto option stops this mount to be done automatically when the system is started (i.e., it stops **mount -a** from mounting it). The user option allows any user to mount the filesystem, and, because of security reasons, disallows execution of programs (normal or setuid) and interpretation of device files from the mounted filesystem. After this, any user can mount a floppy with an msdosfilesystem with the following command:

```
$mount /floppy
```

The floppy can (and needs to, of course) be unmounted with the corresponding **umount** command.

If you want to provide access to several types of floppies, you need to give several mount points. The settings can be different for each mount point. For example, to give access to both MS-DOS and ext2 floppies, you could have the following to lines in /etc/fstab:

```
/dev/fd0 /dosfloppymsdosuser,noauto 0 0 /dev/fd0
  /ext2floppy ext2 user,noauto 0 0
```

### "/opt" Directory:

This directory is reserved for all the software and add-on packages that are not part of the default installation. All third party applications should be installed in this directory. Any package to be installed here must locate its static files (ie. extra fonts, clipart, database files) must locate its static files in a separate /opt/'package' or /opt/'provider' directory tree (similar to the way in which Windows will install new software to its own directory tree C:\Windows\Progam Files\"Program Name"), where 'package' is a name that describes the software package and 'provider' is the provider's LANANA registered name.
Under no circumstances are other package files to exist outside the /opt, /var/opt, and /etc/opt hierarchies except for those package files that must reside in specific locations within the filesystem tree in order to function properly.

### "/proc" Directory:

/proc is very special in that it is also a virtual filesystem. It's sometimes referred to as a process information pseudo-file system. It doesn't contain 'real' files but runtime system information (e.g. system

memory, devices mounted, hardware configuration, etc). For this reason it can be regarded as a control and information centre for the kernel. In fact, quite a lot of system utilities are simply calls to files in this directory. For example, 'lsmod' is the same as 'cat /proc/modules' while 'lspci' is a synonym for 'cat /proc/pci'. By altering files located in this directory you can even read/change kernel parameters (sysctl) while the system is running.

## "/sbin" Directory:

Linux discriminates between 'normal' executables and those used for system maintenance and/or administrative tasks. The latter reside either here or - the less important ones - in /usr/sbin. Locally installed system administration programs should be placed into /usr/local/sbin.

Programs executed after /usr is known to be mounted (when there are no problems) are generally placed into /usr/sbin. This directory contains binaries that are essential to the working of the system. These include system administration as well as maintenance and hardware configuration programs. You may find lilo, fdisk, init, ifconfig, etc.... here.

Another directory that contains system binaries is /usr/sbin. This directory contains other binaries of use to the system administrator. This is where you will find the network daemons for your system along with other binaries that (generally) only the system administrator has access to, but which are not required for system maintenance and repair. Normally, these directories are never part of normal user's $PATHs, only of roots (PATH is an environment variable that controls the sequence of locations that the system will attempt to look in for commands).

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

fastboot Reboot the system without checking the disks (optional)
fasthalt Stop the system without checking the disks (optional)
fdisk Partition table manipulator (optional)
fsck File system check and repair utility (optional)
fsck.* File system check and repair utility for a specific filesystem (optional)
getty The getty program (optional)
halt Command to stop the system (optional)
ifconfig Configure a network interface (optional)
init Initial process (optional)
mkfs Command to build a filesystem (optional)
mkfs.* Command to build a specific filesystem (optional)
mkswap Command to set up a swap area (optional)

reboot Command to reboot the system (optional)
route IP routing table utility (optional)
swapon Enable paging and swapping (optional)
swapoff Disable paging and swapping (optional)
update Daemon to periodically flush filesystem buffers (optional)

**"/usr" Directory:**

/usr usually contains by far the largest share of data on a system. Hence, this is one of the most important directories in the system as it contains all the user binaries, their documentation, libraries, header files, etc.... X and its supporting libraries can be found here. User programs like telnet, ftp, etc.... are also placed here. In the original Unix implementations, /usr was where the home directories of the users were placed (that is to say, /usr/someone was then the directory now known as /home/someone). In current Unices, /usr is where user-land programs and data (as opposed to 'system land' programs and data) are. The name hasn't changed, but it's meaning has narrowed and lengthened from "everything user related" to "user usable programs and data". As such, some people may now refer to this directory as meaning 'User System Resources' and not 'user' as was originally intended.

**"/var" Directory:**

Contains variable data like system logging files, mail and printer spool directories, and transient and temporary files. Some portions of /var are not shareable between different systems. For instance, /var/log, /var/lock, and /var/run. Other portions may be shared, notably /var/mail, /var/cache/man, /var/cache/fonts, and /var/spool/news. Why not put it into /usr? Because there might be circumstances when you may want to mount /usr as read-only, e.g. if it is on a CD or on another computer. '/var' contains variable data, i.e. files and directories the system must be able to write to during operation, whereas /usr should only contain static data. Some directories can be put onto separate partitions or systems, e.g. for easier backups, due to network topology or security concerns. Other directories have to be on the root partition, because they are vital for the boot process. 'Mountable' directories are: '/home', '/mnt', '/tmp', '/usr' and '/var'. Essential for booting are: '/bin', '/boot', '/dev', '/etc', '/lib', '/proc' and '/sbin'.

The following directories, or symbolic links to directories, must be in /var, if the corresponding subsystem is installed:
account Process accounting logs (optional)
crash System crash dumps (optional)
games Variable game data (optional)
mail User mailbox files (optional)

yp Network Information Service (NIS) database files (optional)

## "/tmp" Directory:

This directory contains mostly files that are required temporarily. Many programs use this to create lock files and for temporary storage of data. Do not remove files from this directory unless you know exactly what you are doing! Many of these files are important for currently running programs and deleting them may result in a system crash. Usually it won't contain more than a few KB anyway. On most systems, this directory is cleared out at boot or at shutdown by the local system. The basis for this was historical precedent and common practice. However, it was not made a requirement because system administration is not within the scope of the FSSTND. For this reason people and programs must not assume that any files or directories in /tmp are preserved between invocations of the program.