File System:

What are filesystems?
A filesystem is the methods and data structures that an operating system uses to keep track of files on a disk or partition; that is, the way the files are organized on the disk. The word is also used to refer to a partition or disk that is used to store the files or the type of the filesystem. Thus, one might say ``I have two filesystems'' meaning one has two partitions on which one stores files, or that one is using the ``extended filesystem'', meaning the type of the filesystem.

The difference between a disk or partition and the filesystem it contains is important. A few programs (including, reasonably enough, programs that create filesystems) operate directly on the raw sectors of a disk or partition; if there is an existing file system there it will be destroyed or seriously corrupted. Most programs operate on a filesystem, and therefore won't work on a partition that doesn't contain one (or that contains one of the wrong types).

Before a partition or disk can be used as a filesystem, it needs to be initialized, and the bookkeeping data structures need to be written to the disk. This process is called making a filesystem.

Most LINUX filesystem types have a similar general structure, although the exact details vary quite a bit. The central concepts are superblock, inode , data block, directory block , and indirection block. The superblock contains information about the filesystem as a whole, such as its size (the exact information here depends on the filesystem). An inode contains all information about a file, except its name. The name is stored in the directory, together with the number of the inode. A directory entry consists of a filename and the number of the inode which represents the file. The inode contains the numbers of several data blocks, which are used to store the data in the file. There is space only for a few data block numbers in the inode, however, and if more are needed, more space for pointers to the data blocks is allocated dynamically. These dynamically allocated blocks are indirect blocks; the name indicates that in order to find the data block, one has to find its number in the indirect block first.

LINUX filesystems usually allow one to create a hole in a file (this is done with the lseek() system call; check the manual page), which means that the filesystem just pretends that at a particular place in the file there is just zero bytes, but no actual disk sectors are reserved for that place in the file (this means that the file will use a bit less disk space). This happens especially often for small binaries, Linux shared libraries, some databases, and a few other special cases. (Holes are implemented by storing a special value as the address of the data block in the indirect block or inode. This special address means that no data block is allocated for that part of the file, ergo, there is a hole in the file.)

Comparing Filesystem Features

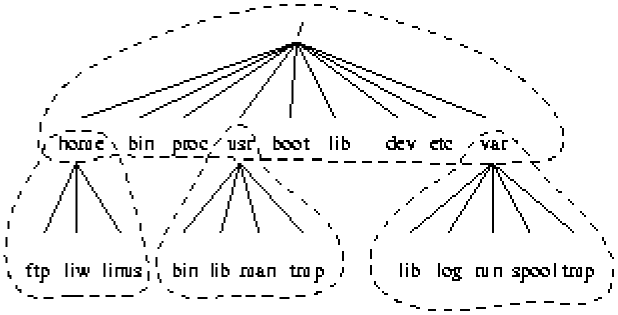
FS Name	Year Introduced	Original OS	Max File Size	Max FS Size	Journaling
FAT16	1983	MSDOS V2	4GB	16MB to 8GB	N
FAT32	1997	Windows 95	4GB	8GB to 2TB	N

FS Name	Year Introduced	Original OS	Max File Size	Max FS Size	Journaling
HPFS	1988	OS/2	4GB	2TB	N
NTFS	1993	Windows NT	16EB	16EB	Y
HFS+	1998	Mac OS	8EB	?	N
UFS2	2002	FreeBSD	512GB to 32PB	1YB	N
ext2	1993	Linux	16GB to 2TB4	2TB to 32TB	N
ext3	1999	Linux	16GB to 2TB4	2TB to 32TB	Y
ReiserFS3	2001	Linux	8TB8	16TB	Y
ReiserFS4	2005	Linux	?	?	Y
XFS	1994	IRIX	9EB	9EB	Y
JFS	?	AIX	8EB	512TB to 4PB	Y
VxFS	1991	SVR4.0	16EB	?	Y
ZFS	2004	Solaris 10	1YB	16EB	N

This topic is loosely based on the *Filesystems Hierarchy Standard* (FHS), which attempts to set a standard for how the directory tree in a Linux system should be organized. Such a standard has the advantage that it will be easier to write or port software for Linux, and to administer Linux machines, since everything should be in standardized places. There is no authority behind the standard that forces anyone to comply with it, but it has gained the support of many Linux distributions. It is not a good idea to break with the FHS without very compelling reasons. The FHS attempts to follow Linux tradition and current trends, making Linux systems familiar to those with experience with other Linux systems, and vice versa.

The full directory tree is intended to be breakable into smaller parts, each capable of being on its own disk or partition, to accommodate to disk size limits and to ease backup and other system administration tasks. The major parts are the root (/), /usr , /var , and /home filesystems (*see the following figure*). Each part has a different purpose. The directory tree has been designed so that it works well in a network of Linux machines which may share some parts of the filesystems over a read-only device (e.g., a CD-ROM), or over the network with NFS.

Parts of a Linux directory tree. Dashed lines indicate partition limits



The roles of the different parts of the directory tree are described below

- The root filesystem is specific for each machine (it is generally stored on a local disk, although it could be a ramdisk or network drive as well) and contains the files that are necessary for booting the system up, and to bring it up to such a state that the other filesystems may be mounted. The contents of the root filesystem will therefore be sufficient for the single user state. It will also contain tools for fixing a broken system, and for recovering lost files from backups.

- The /usr filesystem contains all commands, libraries, manual pages, and other unchanging files needed during normal operation. No files in /usr should be specific for any given machine, nor should they be modified during normal use. This allows the files to be shared over the network, which can be cost-effective since it saves disk space (there can easily be hundreds of megabytes, increasingly multiple gigabytes in /usr). It can make administration easier (only the master /usr needs to be changed when updating an application, not each machine separately) to have /usr network mounted. Even if the filesystem is on a local disk, it could be mounted read-only, to lessen the chance of filesystem corruption during a crash.

- The /var filesystem contains files that change, such as spool directories (for mail, news, printers, etc), log files, formatted manual pages, and temporary files. Traditionally everything in /var has been somewhere below /usr , but that made it impossible to mount /usr read-only.

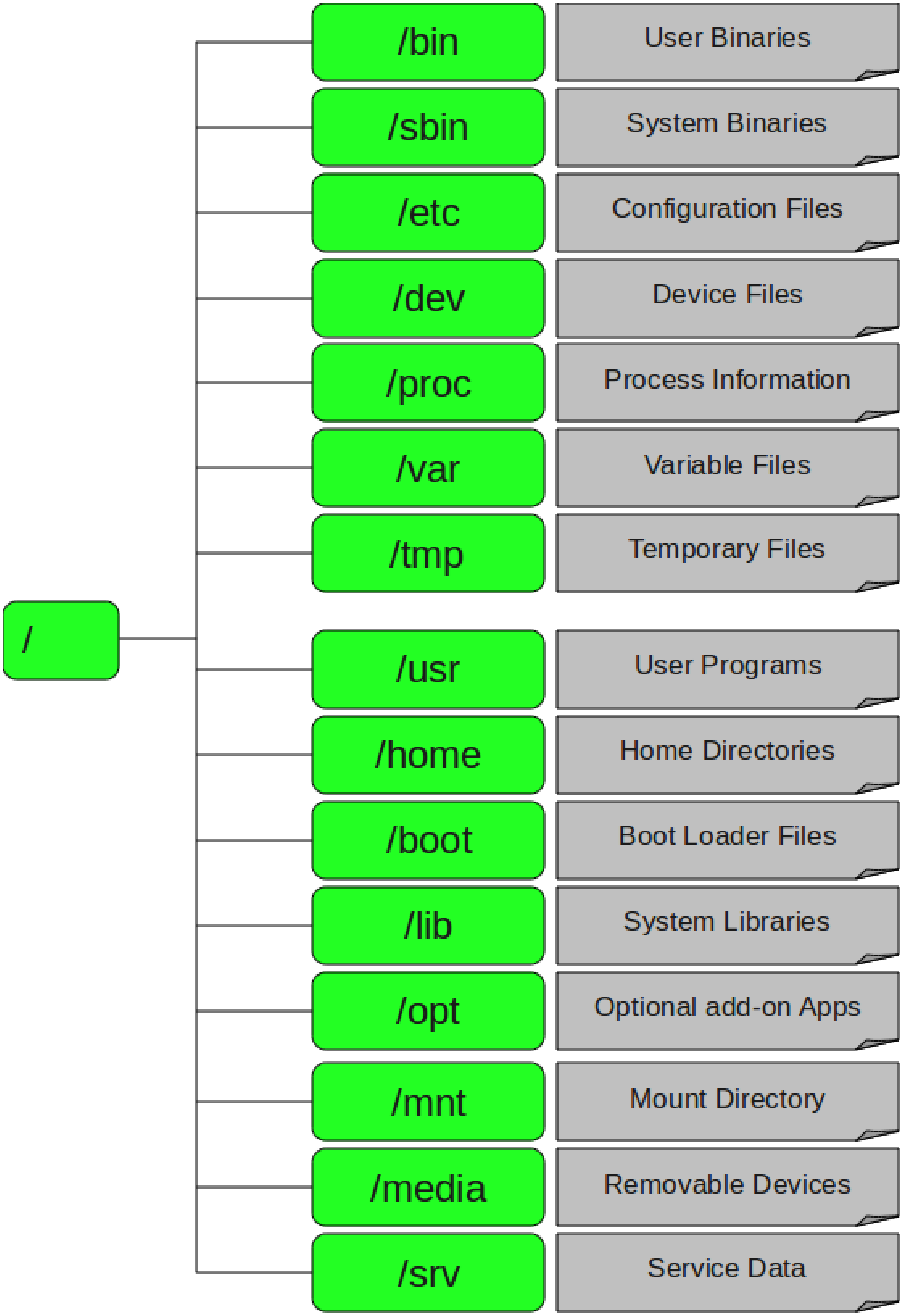
- The /home filesystem contains the users' home directories, i.e., all the real data on the system. Separating home directories to their own directory tree or filesystem makes backups easier; the other parts often do not have to be backed up, or at least not as often as they seldom change. A big /home might have to be broken across several filesystems, which requires adding an extra naming level below /home, for example /home/students and /home/staff.

Although the different parts have been called filesystems above, there is no requirement that they actually be on separate filesystems. They could easily be kept in a single one if the system is a small single-user system and the user wants to keep things simple. The directory tree might also be divided into filesystems differently, depending on how large the disks are, and how space is allocated for various purposes. The important part, though, is that all the standard *names* work; even if, say, /var and /usr are actually on the same partition, the names /usr/lib/libc.a and /var/log/messages must work, for example by moving files below /var into /usr/var, and making /var a symlink to /usr/var.

The Linux filesystem structure groups files according to purpose, i.e., all commands are in one place, all data files in another, documentation in a third, and so on. An alternative would be to group files according to the program they belong to, i.e., all Emacs files would be in one directory, all TeX in another, and so on. The problem with the latter approach is that it makes it difficult to share files (the program directory often contains both static and sharable and changing and non-sharable files), and sometimes to even find the files (e.g., manual pages in a huge number of places, and making the manual page programs find all of them is a maintenance nightmare).

The root filesystem should generally be small, since it contains very critical files and a small, infrequently modified filesystem has a better chance of not getting corrupted. A corrupted root filesystem will generally mean that the system becomes unbootable except with special measures (e.g., from a floppy), so you don't want to risk it.

The root directory generally doesn't contain any files, except perhaps on older systems where the standard boot image for the system, usually called /vmlinuz was kept there. (Most distributions have moved those files the the /boot directory.



/

/bin	User Binaries
/sbin	System Binaries
/etc	Configuration Files
/dev	Device Files
/proc	Process Information
/var	Variable Files
/tmp	Temporary Files
/usr	User Programs
/home	Home Directories
/boot	Boot Loader Files
/lib	System Libraries
/opt	Optional add-on Apps
/mnt	Mount Directory
/media	Removable Devices
/srv	Service Data

1. / – Root

- Every single file and directory starts from the root directory.
- Only root user has write privilege under this directory.
- Please note that /root is root user's home directory, which is not same as /.

2. /bin – User Binaries

- Contains binary executables.
- Common linux commands you need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here.
- For example: ps, ls, ping, grep, cp.

3. /sbin – System Binaries

- Just like /bin, /sbin also contains binary executables.
- But, the linux commands located under this directory are used typically by system aministrator, for system maintenance purpose.
- For example: iptables, reboot, fdisk, ifconfig, swapon

4. /etc – Configuration Files

- Contains configuration files required by all programs.
- This also contains startup and shutdown shell scripts used to start/stop individual programs.
- For example: /etc/resolv.conf, /etc/logrotate.conf

5. /dev – Device Files

- Contains device files.
- These include terminal devices, usb, or any device attached to the system.
- For example: /dev/tty1, /dev/usbmon0

6. /proc – Process Information

- Contains information about system process.
- This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual filesystem with text information about system resources. For example: /proc/uptime

7. /var – Variable Files

- var stands for variable files.

- Content of the files that are expected to grow can be found under this directory.
- This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

8. /tmp – Temporary Files

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.

9. /usr – User Programs

- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If you can’t find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
- /usr/sbin contains binary files for system administrators. If you can’t find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2

10. /home – Home Directories

- Home directories for all users to store their personal files.
- For example: /home/john, /home/nikita

11. /boot – Boot Loader Files

- Contains boot loader related files.
- Kernel initrd, vmlinux, grub files are located under /boot
- For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

12. /lib – System Libraries

- Contains library files that supports the binaries located under /bin and /sbin
- Library filenames are either ld* or lib*.so.*
- For example: ld-2.11.1.so, libncurses.so.5.7

13. /opt – Optional add-on Applications

- opt stands for optional.
- Contains add-on applications from individual vendors.
- add-on applications should be installed under either /opt/ or /opt/ sub-directory.

14. /mnt – Mount Directory

- Temporary mount directory where sysadmins can mount filesystems.

15. /media – Removable Media Devices

- Temporary mount directory for removable devices.
- For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer

16. /srv – Service Data

- srv stands for service.
- Contains server specific services related data.
- For example, /srv/cvs contains CVS related data