The Filesystem

In Linux, everything is a file

- Processes
- Audio devices
- Kernel data structures and tuning parameters
- Interprocess communication channels

Main components

- A namespace
- An API
- Security models
- An implementation

Pathnames

- Single unified hierarchy start at root: `/`
- Absolute path: path name starts from root
- Relative path: path name starts from current directory: `.` or subdirectory name

Mounting and unmounting

- The root filesystem is composed of smaller trunks (smaller filesystems)
- Smaller filesystems are attached to the tree with the `mount` command, which ...
  - Maps a directory within the existing filesystem tree, called the mount point, to the root of the newly attached filesystem.

Recalling the procedure to export id_rsa.pub

https://gist.github.com/estorgio/1d679f962e8209f8a9232f7593683265

Who is doing what on which filesystem?

```
-  $ man fuser
-  $ fuser -cv /home
```

Instead of rebooting, perhaps unmounting/remounting of offending device drivers ...

Organization of the filesystem tree

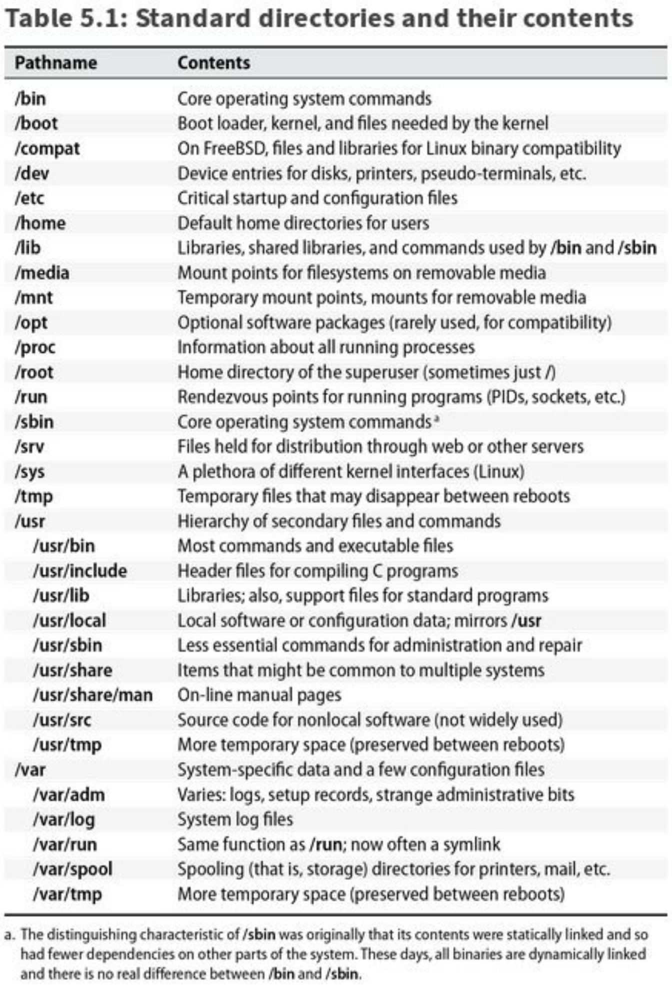

Table 5.1: Standard directories and their contents

Pathname	Contents
/bin	Core operating system commands
/boot	Boot loader, kernel, and files needed by the kernel
/compat	On FreeBSD, files and libraries for Linux binary compatibility
/dev	Device entries for disks, printers, pseudo-terminals, etc.
/etc	Critical startup and configuration files
/home	Default home directories for users
/lib	Libraries, shared libraries, and commands used by **/bin** and **/sbin**
/media	Mount points for filesystems on removable media
/mnt	Temporary mount points, mounts for removable media
/opt	Optional software packages (rarely used, for compatibility)
/proc	Information about all running processes
/root	Home directory of the superuser (sometimes just /)
/run	Rendezvous points for running programs (PIDs, sockets, etc.)
/sbin	Core operating system commands[a]
/srv	Files held for distribution through web or other servers
/sys	A plethora of different kernel interfaces (Linux)
/tmp	Temporary files that may disappear between reboots
/usr	Hierarchy of secondary files and commands
/usr/bin	Most commands and executable files
/usr/include	Header files for compiling C programs
/usr/lib	Libraries; also, support files for standard programs
/usr/local	Local software or configuration data; mirrors **/usr**
/usr/sbin	Less essential commands for administration and repair
/usr/share	Items that might be common to multiple systems
/usr/share/man	On-line manual pages
/usr/src	Source code for nonlocal software (not widely used)
/usr/tmp	More temporary space (preserved between reboots)
/var	System-specific data and a few configuration files
/var/adm	Varies: logs, setup records, strange administrative bits
/var/log	System log files
/var/run	Same function as **/run**; now often a symlink
/var/spool	Spooling (that is, storage) directories for printers, mail, etc.
/var/tmp	More temporary space (preserved between reboots)

[a] The distinguishing characteristic of **/sbin** was originally that its contents were statically linked and so had fewer dependencies on other parts of the system. These days, all binaries are dynamically linked and there is no real difference between **/bin** and **/sbin**.

Filetype encoding

- Character/block device file: standard communication interface provided by device drivers.
- Local domain sockets: connections between processes that allow them to communicate hygienically.
- Named pipes allow communication between two processes running on the same host.
- Symbolic links: point to a file by name
- Hard links: create an illusion that a file exists in more than one place at the same time.

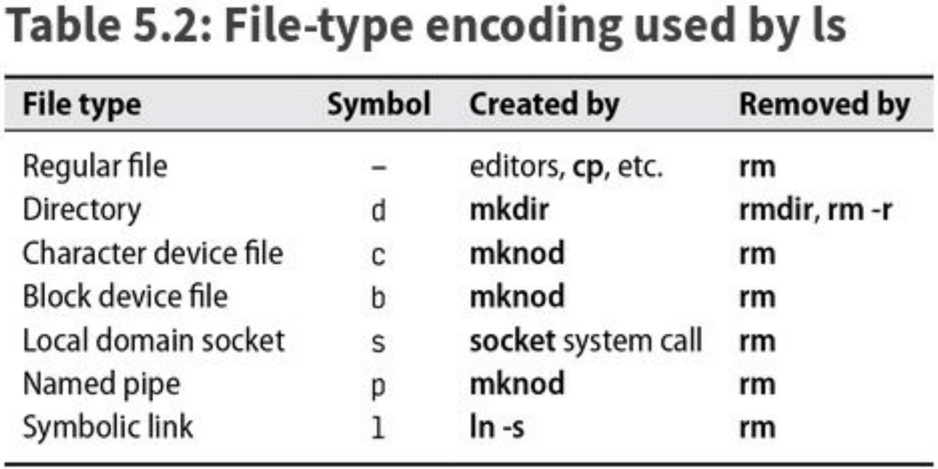

Table 5.2: File-type encoding used by ls

File type	Symbol	Created by	Removed by
Regular file	-	editors, **cp**, etc.	**rm**
Directory	d	**mkdir**	**rmdir, rm -r**
Character device file	c	**mknod**	**rm**
Block device file	b	**mknod**	**rm**
Local domain socket	s	**socket** system call	**rm**
Named pipe	p	**mknod**	**rm**
Symbolic link	l	**ln -s**	**rm**

File Attributes

- Traditionally 12 bits for each file: the file's mode (plus 4 more bits : file's type)
- 9 permission bits - read, write, execute for owner, group, others
- setuid & setgid bits (4000 , 2000)
  - setgid on directory - newly created file has group ownership of the directory (not group ownership of a user creating it)
- sticky bit (1000)
  - on regular files ignored (original meaning: keep program text on swap device)
  - on directories - only the owner of the file and the owner of that directory may remove the file from that directory

```
-   ls -l (ls -ld)
-   chmod
-   chown user.group file (-R)
-   umask encoding
```

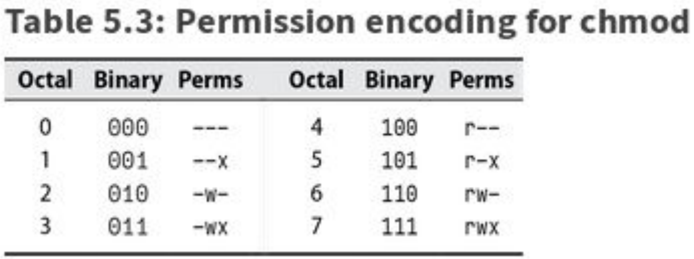

Table 5.3: Permission encoding for chmod

Octal	Binary	Perms	Octal	Binary	Perms
0	000	---	4	100	r--
1	001	--x	5	101	r-x
2	010	-w-	6	110	rw-
3	011	-wx	7	111	rwx

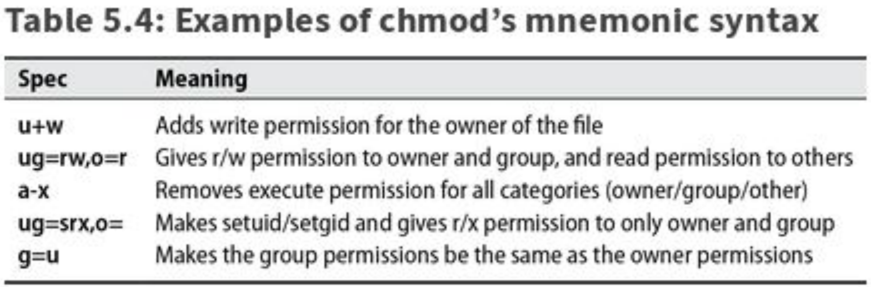

Table 5.4: Examples of chmod's mnemonic syntax

Spec	Meaning
u+w	Adds write permission for the owner of the file
ug=rw,o=r	Gives r/w permission to owner and group, and read permission to others
a-x	Removes execute permission for all categories (owner/group/other)
ug=srx,o=	Makes setuid/setgid and gives r/x permission to only owner and group
g=u	Makes the group permissions be the same as the owner permissions

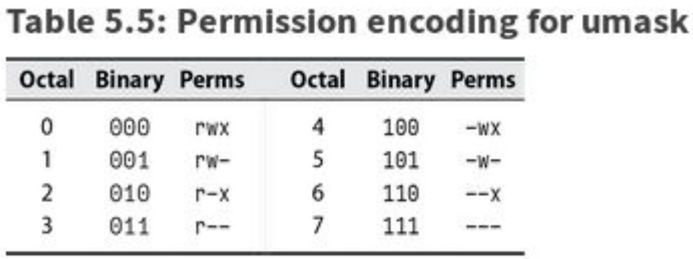

Table 5.5: Permission encoding for umask

Octal	Binary	Perms	Octal	Binary	Perms
0	000	rwx	4	100	-wx
1	001	rw-	5	101	-w-
2	010	r-x	6	110	--x
3	011	r--	7	111	---

Access Control Lists

- supported for ext2, ext3, ext4, reiserfs, XFS, JFS
- `$ mount -o [no]acl`
- allows rwx to be set independently for any user.group combination
- `getfacl`, `setfacl` (plus `man acl`)
- NFSv4 - superset of POSIX ACLs plus all permission bits and most semantics from Windows