CST8177

File Systems

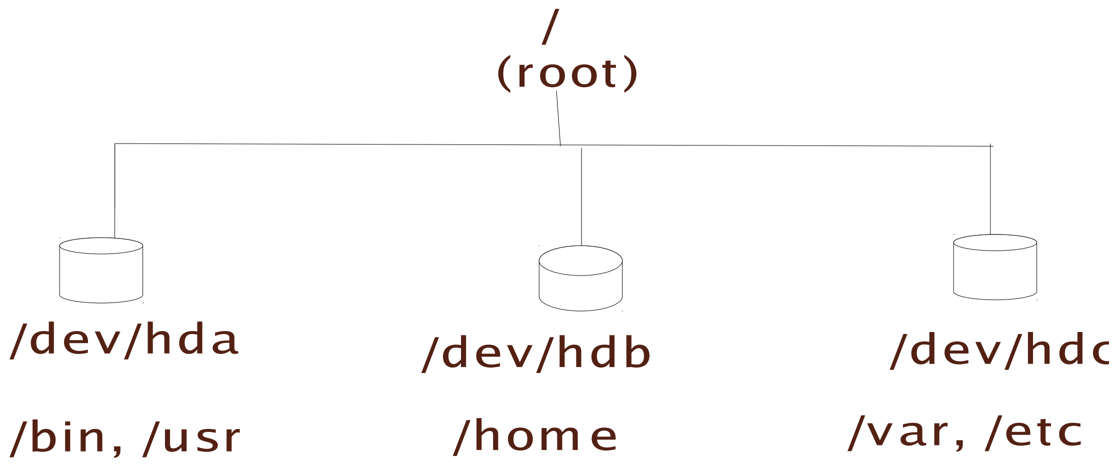
Linux File Systems

Layout of the file system:

- Each **physical drive** can be divided into several **partitions**
- Each partition can contain one **file system**
- Each file system contains:
    1. boot block(s);
    2. superblock;
    3. inode list;
    4. data blocks.

- A **boot block** may contain the bootstrap code that is read into the machine upon booting.
- A **superblock** describes the state of the file system:
  - how large it is;
  - how many files it can store;
  - where to find free space on the file system;
  - who has ownership of it;
  - and more.
- The **inode list** is an array of "**i**nformation **node**s" analogous to the FAT (File Allocation Table) system or VFAT in MS-DOS and Windows.
- **data blocks** start at the end of the inode list and contain file data and directory blocks.

The directory "tree" usually spans many disks and/or partitions by means of **mount points**. For example, in many versions of Linux, there are pre-defined mount points for floppy disks and CD-ROMs at **`floppy`** and **`cdrom`** in **`/mnt`**. USB devices may be mounted in **`/mnt`** or in **`/media`**. See also the files **`fstab`** and **`mtab`** in **`/etc`**.



Some Linux-supported File Systems

minix is the filesystem used in the Minix operating system, the first to run under Linux. It has a number of shortcomings, mostly in being small. It led to ext.

ext is now gone, completely replaced by ext2.

ext2 is a disk filesystem used by Linux for both hard drives and floppies. ext2, designed as an extension to ext, has in its turn generated a successor, ext3.

ext3 offers improved (in terms of speed and CPU usage) combined with data security of the file systems supported under Linux due to its journaling feature.

ext4 enhances ext3 and is now the default.

msdos is the filesystem used by MS-DOS and early Windows. msdos filenames are limited to the 8 + 3 form.

vfat extends msdos to be compatible with Microsoft Windows' support for long filenames. Often used on USB keys for inter-system portability.

ntfs replaces Window's VFAT file systems with reliability, performance, and space-utilization enhancements plus features like ACLs, journaling, encryption, and so on.

proc is a pseudo-filesystem which is used as an interface to kernel data. Its files do not use disk space. See proc(5).

iso9660 is a CD-ROM filesystem type conforming to the ISO 9660 standard, including both High Sierra and Rock Ridge.

nfs is a network filesystem used to access remote disks, mostly on TCP/IP networks.

nfs4 is the current version of nfs.

smb is a network filesystem used by Windows.

Partition Structure

Boot Block(s)

Blocks on a Linux (and often a Unix) filesystem are 512 bytes in length, but may be longer or shorter. The blocks are normally a power of 2 in size (512 is 2 to the 9th power). Some systems use 1024 bytes (2 to the 10th) but 2048 and 4096 are also seen.

The first few blocks on any partition are reserved for a boot program, a short program for loading the kernel of the operating system and launching it. Often on Linux systems it will be controlled by GRUB, allowing booting of multiple operating systems. It's quite common to be able to boot both Linux and a flavour of Windows.

Superblock

The boot blocks are followed by the unique superblock, which contains information about the geometry of the physical disk, the layout of the partition, number of inodes and data blocks, and much more.

Inode Blocks

Disk space allocation is managed by thousands of inodes (**i**nformation **node**), which are created by the **`mkfs(1)`** ("make filesystem") command. Inodes cannot be manipulated directly, but are changed by many commands, such as **`touch(1)`**, **`cp(1)`**, **`mv(1)`**, and **`rm(1)`**, and can be read by **`ls(1)`** and **`stat(1)`**. Both **`chmod(1)`** and **`chown(1)`** can change inode contents.

Data Blocks

This is where the file data itself is stored. Since a directory is simply a specially formatted file, directories are also contained in the many, many data blocks. An allocated data block can belong to one and only one file in the system. If a data block is not allocated to a file, it is free and available for the system to allocate when needed.

Some parts of a super block

Inodes count	- how many inodes
Blocks count	- how many data blocks
Reserved blocks count	- spare data blocks
Free blocks count	- unused space
Free inodes count	- unused inodes
First Data Block	- address of 1st data block
Block size	- size of a "block"
Mount time	- when was it mounted
Write time	- when it was last written
Mount count	- how many mounts
Maximal mount count	- max possible mounts
Magic signature	- a superblock identifier
time of last check	- last fsck
max time between checks	- max time between fscks
First non-res. Inode	- first non-reserved inode
size of inode structure	- how big is an inode
volume name	- volume label
dir where last mounted	- where was it last mounted

Structure of an inode on the disk

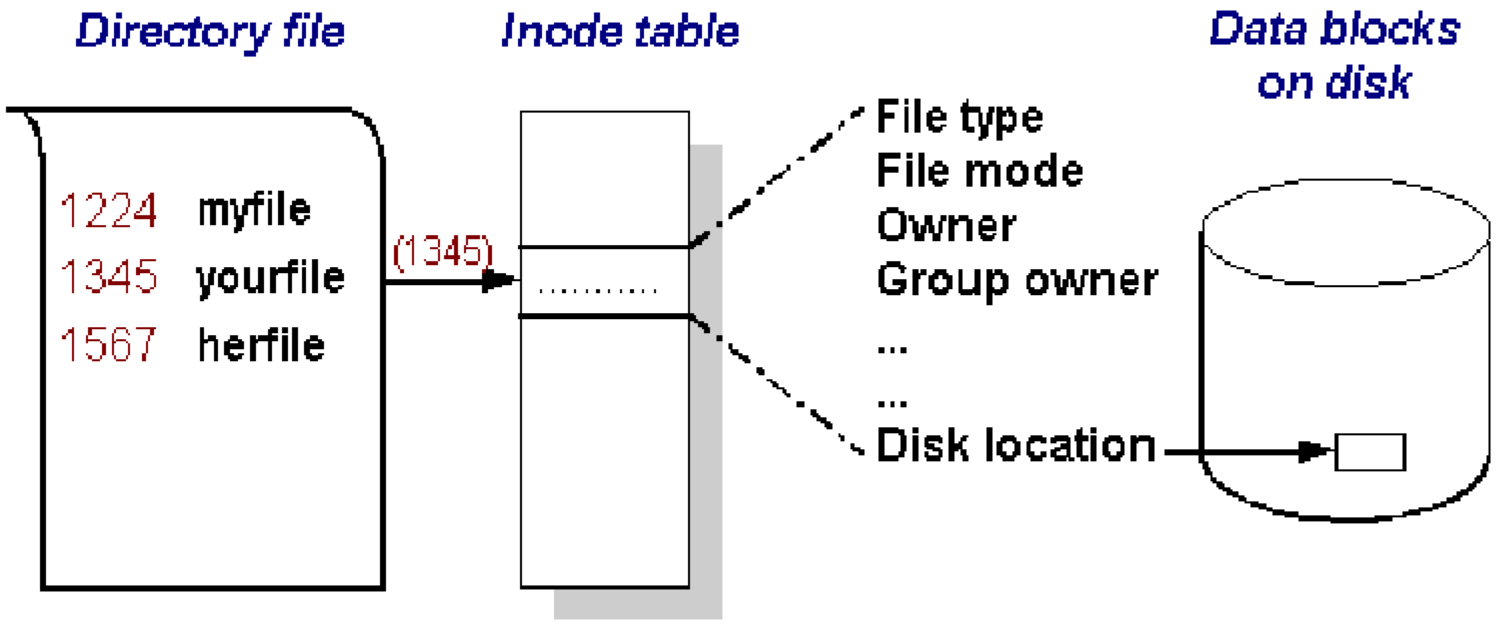
Each file (that is, a unique collection of data blocks) has only 1 inode, which completely defines the file except for its name(s). The filenames are actually links in the directory structure to the inode for the file.

A file of data may have many names, but only one inode.

Since filenames are only kept in the directory tree, any directory entry is a link to the actual description of the file, the inode.

Directory Tree		**Inode List**		**Data blocks**
1 or more names	=>	an inode	=>	the file contents

ext Sample


Directory file
Inode table
Data blocks on disk
1224 myfile
1345 yourfile
1567 herfile
(1345)
File type
File mode
Owner
Group owner
...
...
Disk location

Sample Inode Fields

file type and permissions	- mode flags
device number	- relates to /dev types
number of links	- hard link count
owners user id	- UID
owners group id	- GID
size in bytes	- actual size
time of last Access	- 3 time fields: atime
time of last Modify	- mtime
time of status Change	- ctime
number of disk blocks	- number of blocks
List of data blocks	- in sequence

Directory entry

Use the path of the directory plus the filename in a call as the absolute path to the file.

The directory entry is basically just these fields:

- `Inode number`
- `Filename`

File location = <directory path> + / + <filename>

The first two directory entries are the `.` (self) and `..` (parent) entries, which are always present. For the `/` directory only, both `.` and `..` refer to the same inode number.

The link count field of an inode for a directory holds the number of directory references to that inode. Therefore, the minimum for a directory is a link count of 2 (self and parent's name entry) and increases by 1 for each direct sub-directory.

Sample directory dir-root

inode number	filename
201	.
150	..
202	file1
203	file2
204	dir1
--	--
--	--

Sample directory dir1

inode number	filename
204	.
201	..
207	dir1-1
208	dir1-2
--	--
--	--
--	--

Sample directory dir1-1

inode number	filename
207	.
204	..
209	file1
--	--
--	--
--	--
--	--

Sample directory dir1-2

inode number	filename
208	.
204	..
210	file1
--	--
--	--
--	--
--	--

Access Permissions

- permissions are made up of "**r**", "**w**", "**x**", for read, write and executable (access for directories), in 3 categories:
  user, **g**roup, and **o**ther
- permissions are changed by the **`chmod(1)`** command.
- Often represented in 3-digit octal numbers: **`0755`**, **`0644`**
- Can use symbolic notation: from **`0666`**, the command **`chmod g-w,o=    <filename>`** results in **`0640`**
- Permission “**`x`**” has different meanings:
  - For files: executable
  - For directories: access (axxess, if that helps).

Files

- A collection of data blocks, as listed in the inode
- If the list in an inode is full, a second (third, 4th, etc.) inode is used for more data blocks
- A file's name is **<u>only</u>** in the directory (or in several directories)
- Have as many links as they have names (hard links)

Directories

- A collection of data blocks, as listed in the inode
- Always contain at least two entries: "`.`" (self) and "`..`" (parent)
- Have 2 links plus the total number of its subdirectories
- The root directory is different in that the <u>self</u> and <u>parent</u> entries are absolutely identical

Filenames

- A filename is the name field in a directory entry, and can refer to a data file, a directory, or other things not yet seen.

Linking files

In Linux and Unix, a data file is a bunch of data blocks on a disk, managed by an inode. Its name is stored only in the directory. Or in many directories. This is the concept of linking. Both "soft" (symbolic) links and "hard" links can be made using the **ln(1)** command. Below we create a second name for a file as a hard link:

Original file

```
System Prompt: ls -l
-rw-r--r-- 1 allisor allisor 0 D/T abc
```

Create hard link

```
System Prompt: ln abc h-abc

System Prompt: ls -l
-rw-r--r-- 2 allisor allisor 0 D/T abc
-rw-r--r-- 2 allisor allisor 0 D/T h-abc
```

Symbolic (soft) links don't create a second name like hard links, they create an alias or shortcut to an existing name:

Create soft link

```
System Prompt: ls -l
-rw-r--r-- 2 allisor allisor 0 D/T abc
-rw-r--r-- 2 allisor allisor 0 D/T h-abc

System Prompt: ln -s abc s-abc

System Prompt: ls -l
-rw-r--r-- 2 allisor allisor 0 D/T abc
lrwxrwxrwx 1 allisor allisor 3 D/T s-abc -> abc
-rw-r--r-- 2 allisor allisor 0 D/T h-abc
```

Examine inodes

```
System Prompt: ls -i1
25263 abc
25263 h-abc
25265 s-abc
```

Remove original filename (not the file)

```
System Prompt: ls -l
-rw-r--r-- 2 allisor staff 0 D/T abc
lrwxr-xr-x 1 allisor staff 3 D/T s-abc -> abc
-rw-r--r-- 2 allisor staff 0 D/T h-abc

System Prompt: rm abc

System Prompt: ls -l
-rw-r--r-- 1 allisor staff 0 D/T h-abc
lrwxr-xr-x 1 allisor staff 3 D/T s-abc -> abc
```

Removing one hard link has no effect on another hard link, but it can break a soft link so it no longer points to a file.

Here, the original file still exists although its name (which was just another hard link) is gone. Its second name is still there and provides access to the inode and data content.

The soft (symbolic) link **`s-abc`**, however, is now broken, and is no longer of any use.

Link counts

Always: the number of times that inode's number appears anywhere in the directory tree in that filesystem.

Directory: starts at 2, and adds one for every direct sub-directory.

Sub-directory: increases its direct parent's link count by 1.

File: starts at 1, and increases by one for each hard link (never increases for a soft link).

Hard link: another filename for a file, completely the same as any other name for that file. Increases the file's link count by 1. The file is removed only when the link count is zero.

Soft link: also known as a symbolic link is an alias or shortcut to a file or directory. It has no effect on the file's link count. The soft link contains a path and name for its target, and is broken if that target is moved, renamed, or deleted. Nothing happens to the target for any change to its soft link(s) – no change to its link count.