



Linux Administration

The Filesystem

Lecturer:
Di Dio Lavore, Galtarossa
Politecnico di Milano
www.polimi.it

© 2003 - Di Dio Lavore, Galtarossa

Summary



-
- Introduction
 - Types of filesystems
 - The organization of the file tree
 - ▶ Filesystem Hierarchy Standard (FHS)
 - ▶ / (root) view
 - Mounting and Unmounting
 - ▶ **mount** command
 - ▶ **umount** command
 - ▶ **/etc/fstab** file
 - **fsck** command
 - Making a filesystem
 - Tools for all filesystems
 - Other tools for the **ext2** filesystem
 - X-based Interfaces
 - Appendix - File types

Introduction



- A filesystem is the methods and data structures that an operating system uses to keep track of files
- The filesystem can be thought of as comprising four main components:
 - ▶ A name space
 - ▶ An API
 - ▶ A security model
 - ▶ An implementation
- Most filesystems are disk partitions but they can be anything that obeys the proper API

Introduction (2)



- The difference between a disk partition and the filesystem it contains is important

NOTE:

- ▶ Few programs operate directly on the raw sectors of a disk partition
 - ▶ Most programs operate on a filesystem, and therefore won't work on a partition that doesn't contain one
- Before a disk partition 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*



- Not all disks or partitions are used as filesystems
 - ▶ A **swap partition** will not have a filesystem on it
 - ▶ Many floppies are used in a tape-drive emulating fashion, so that a file is written directly on the raw disk, without a filesystem
 - ▶ Linux boot floppies don't contain a filesystem, only the raw kernel

Types of filesystems



- Linux supports several types of native filesystems
 - ▶ ***minix***
The oldest (at most 64 MB per filesystem)
 - ▶ ***xia***
A modified version of the ***minix***
 - ▶ ***ext2***
The most popular of the native Linux filesystems
 - ▶ ***ext***
An older version of ***ext2***

Types of filesystems (2)



- ▶ **ext3**
It is a remarkable extension of **ext2** and it is the actual **standard**
It adds journaling capability that makes data loss less likely

- ▶ **reiserfs**
A robust filesystem with journaling capability

NOTE:

- ▶ Journaling is a mechanism whereby a record is kept of transaction which are to be performed, or which have been performed
- ▶ This allows the filesystem to reconstruct itself fairly easily after damage caused by, for example, improper shutdowns
- ▶ **ext2**, the most popular filesystem, has special tools

Types of filesystems (3)



- In addition Linux can work with many foreign filesystems
 - ▶ **msdos**
Compatibility with MS-DOS FAT filesystems

 - ▶ **vfat**
This is an extension of the FAT filesystem known as FAT32

 - ▶ **ntfs**
The Windows NT, 2000 and XP filesystem (read only access)

 - ▶ **hpfs**
The OS/2 “high-performance” filesystem

Types of filesystems (4)



- ▶ **sysv**
Unix System V filesystem
- ▶ **iso9660**
The standard CD-ROM filesystem
- ▶ ...

NOTE:

These foreign filesystems work just like native ones, except that they may be lacking in some usual Linux/Unix features, or have curious limitations, or other oddities

Types of filesystems (special)



- ▶ **nfs**
A *networked* filesystem that allows sharing a filesystem between many computers to allow easy access to the files from all of them
- ▶ **smbfs**
A *networked* filesystem which allows sharing of a filesystem with an MS Windows computer
- ▶ **proc**
This is not really a filesystem at all, even though it looks like one
 - The **proc** filesystem makes it easy to access certain kernel data structures, such as the process list
 - It makes these data structures look like a filesystem, and that filesystem can be manipulated with all the usual file tools
- ▶ **devpts**
This filesystem is a pseudo filesystem (pseudo terminal)

The organization of the file tree



- The overall layout of the Linux filesystem (the *file tree*) can be thought like a set of smaller *filesystems*
- The *file tree* is presented as a single unified hierarchy that starts at the directory */* (**root**) and continues downward through an arbitrary number of subdirectories
- Before one can use a *filesystem*, it has to be *mounted* to the *file tree*
- */* (**root**) is the first *filesystem* to be mounted (usually an **ext3 filesystem**) and it represents the initial *file tree*

The organization of the file tree (2)



- 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
- The **root filesystem** is usually first *mounted* read-only
- The startup scripts will then run **fsck** (filesystem consistency check) to verify its validity, and if there are no problems, they will *re-mount* it so that writes will also be allowed



- The organization of the *file tree* is based on the *Filesystems Hierarchy Standard (FHS)* version 2.1
- The *FHS* attempts to follow Unix tradition and current trends, making Linux systems familiar to those with experience with other Unix systems, and vice versa
- It has gained the support of many Linux distributions
- 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
- **A system Administrator should read the full *FHS* for a complete understanding**

FHS (2)



The file tree standard directory and their contents:

- ***/bin***
Commands needed for minimal system operability
They might be used by normal users
- ***/sbin***
Commands for booting, repairing, or recovering the system
These commands are not intended for normal users, although they may use them if necessary and allowed
It will be in root's default path
- ***/root***
The home directory of the superuser
This is usually not accessible to other users on the system



- ***/etc***
Critical startup and configuration files

- ***/lib***
Shared libraries and parts of the C compiler
 - ▶ ***/lib/modules***
It contains the loadable kernel modules, especially those that are needed to boot the system when recovering from disasters

- ***/tmp***
Temporary files that disappear between reboots

- ***/home***
It contains the home directories of the users



- ***/dev***
It contains the special device files for all the devices (terminals, disks, modems,...)
 - ▶ ***/dev/hda5***
This is the logical partition, inside extended partition, of the master IDE drive on the primary IDE controller

- ***/boot***
Kernel and files needed to load the kernel
Files used by the bootstrap loader (LILO or GRUB)

- ***/mnt***
Mount point for temporary mounts by the system Administrator
/mnt might be divided into subdirectories (*/mnt/dos* might be the floppy drive using an *msdos filesystem*)



- **/usr**
 Hierarchy of secondary files and commands
 All programs are installed there
 All files in /usr usually come from a Linux distribution
- **/var**
 System-specific data and configuration files
- **/proc**
 Images of all running processes
 This directory contains a *illusionary filesystem*
 It does not exist on a disk, the kernel creates it in memory
 It is used to provide information about the system

/ (root) view



```
[root@localhost /]# ls -l
```

```
drwxr-xr-x    2 root    root          4096 May 13 19:21 bin/
drwxr-xr-x    3 root    root          4096 Jun  4 12:45 boot/
drwxr-xr-x    1 root    root           0 Jan  1 1970 dev/
drwxr-xr-x   55 root    root          4096 Jun  4 13:01 etc/
drwxr-xr-x    5 root    root          4096 May 13 21:30 home/
drwxr-xr-x    2 root    root          4096 May 13 17:41 initrd/
drwxr-xr-x    9 root    root          4096 May 13 19:12 lib/
drwxr-xr-x   12 root    root          4096 Jun  4 12:43 mnt/
drwxr-xr-x    2 root    root          4096 Aug 23 1999 opt/
dr-xr-xr-x   77 root    root           0 Jun  4 12:45 proc/
drwx-----  11 root    root          4096 Jun  4 13:11 root/
drwxr-xr-x    2 root    root          4096 May 13 19:13/sbin/
drwxrwxrwt   13 root    root          4096 Jun  4 12:46 tmp/
drwxr-xr-x   12 root    root          4096 May 13 19:29 usr/
drwxr-xr-x   16 root    root          4096 May 13 19:11 var/
```

- **Mandrake distribution**
 - ▶ **/initrd**: information for booting
 - ▶ **/opt**: typically contains extra and third party software

Mounting and Unmounting



- A *filesystem* must be *mounted* before it becomes available to Linux processes
- The *mount point* for a supported *filesystem* can be any directory (but *FSH* shows the best)
- *Filesystems* are attached to the tree with the **mount** command and are detached with **umount** command
 - ▶ These commands maintain a list of currently *mounted filesystems* in the file **/etc/mtab**
- A list of the *filesystems* that are customarily mounted on a particular system is kept in the **/etc/fstab** file (file system table)

mount command



- **mount** maps a directory within the existing *file tree*, called the *mount point*, to the root of the newly attached *filesystem*
- The standard form of the command, is:

mount -t type device dir
 - ▶ This tells the kernel to attach the *filesystem* found on device (which is of type **type**) at the directory **dir**
 - ▶ The pathname **dir** refers to the root of the *filesystem* on device
- Most devices are indicated by a file name (of a block special device), like `/dev/hda5`

mount command (2)



- The *proc filesystem* is not associated with a special device, and when mounting it, an arbitrary keyword, such as **proc**, can be used instead of a device specification
- In the case of an *nfs* mount, device may look like
`knuth.cwi.nl:/dir`
- An example: `# mount /dev/hda5 /mnt/data`

This command installs the *filesystem* stored on the disk partition represented by `/dev/hda5` under the path `/mnt/data`

NOTE:

The previous contents of the *mount point* become inaccessible as long as another *filesystem* is mounted there

mount command (3)



- **mount -a** mounts all filesystems listed in the **fstab** file
- The **-t** flag constrains the operation to filesystems of a certain type (*iso9660,vfat,...*)
- Options are specified with a **-o** flag followed by a comma separated string of options. Some options are the following:
 - ▶ **exec**: permit execution of binaries
 - ▶ **rw**: mount the file system read-write
 - ▶ **ro**: mount the file system read-only
 - ▶ **nouser**: forbid an ordinary user to mount the filesystem
 - ▶ **users**: allow every user to mount and unmount the filesystem
 - ▶ **suid**: allow set-user or set-group identifier bits to take effect
 - ▶ ...

mount command (4)



- Some options are only useful when they appear in the `/etc/fstab` file:
 - ▶ **auto**: can be mounted with the `-a` option
 - ▶ **noauto**: can only be mounted explicitly
 - ▶ **defaults**: use default options `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, and `async`
 - ▶ ...

NOTE:

Many supported filesystems have specific options, for example:

- ▶ **(iso9660) unhide**: also show hidden and associated files
- ▶ **(vfat) iocharset=value**: character set to use for converting between 8 bit characters and 16 bit Unicode characters

mount command (5)



- If `/etc/fstab` file is corrupted the system mounts the `/` (**root**) *filesystem* in read-only mode
- In this case we are obliged to *re-mount* the *filesystem* in read-write mode to modify `fstab` file:

```
# mount -n -o remount,rw /
```

- ▶ **-n**: mount without writing in `/etc/mtab`
- ▶ **remount**: attempt to remount an already-mounted *filesystem* It does not change device or *mount point*

umount command



- The **umount** command has the same syntax of **mount** command
- **umount** detaches the *filesystem* mentioned from the file hierarchy
- The *filesystem* is specified by giving the directory where it has been mounted

NOTE:

- ▶ Giving the special device on which the *filesystem* lives may also work, but it will fail in case this device was mounted on more than one directory
- ▶ You cannot unmount a *filesystem* that is “busy” (open files...)
- ▶ **fuser** command, when invoked with the **-mv** flags and a *mount point*, displays every process that’s using a file or directory on that *filesystem*

/etc/fstab file



- The **/etc/fstab** file contains descriptive information about the filesystems:
 - ▶ Lists the *filesystems* mounted automatically at startup by the **mount -a** command (in **/etc/rc** or equivalent startup file)
 - ▶ Contains information about swap areas used automatically by **swapon -a**
 - ▶ Serves as documentation for the layout of the *filesystems* on disk
 - ▶ Enables short commands
When mounting a *filesystem* mentioned in **fstab**, it suffices to give only the device, or only the *mount point*
- **fstab** is only read by programs, and not written

/etc/fstab file (2)



- Each *filesystem* is described on a separate line
- There are 6 fields per line separated by spaces:
 1. *fs_spec*: describes the block special device or remote *filesystem* to be mounted
 2. *fs_file*: describes the mount point for the *filesystem*
 3. *fs_vfstype*: describes the type of the *filesystem*
 4. *fs_mntops*: describes the mount options associated with the *filesystem*
 5. *fs_freq*: is used by the **dump** command
 6. *fs_passno*: is used by the **fsck** command

/etc/fstab file (3)



- It is the duty of the system Administrator to properly create and maintain this file
- For example, when **fstab** contains the **users** option on a line anybody can mount the corresponding system

- ▶ Thus, given a line:

```
/dev/cdrom /cd iso9660 ro,users,noauto,unhide
```

any user can mount the **iso9660 filesystem** found on his CDROM using the command:

```
mount /dev/cdrom or mount /cd
```

/etc/fstab file : an example



```
/dev/hda9 / ext3 defaults 1 1

none /dev/pts devpts mode=0620 0 0

/dev/hda8 /home ext3 defaults 1 2

none /mnt/cdrom supermount dev=/dev/scd0,fs=auto,ro,--,iocharset=iso8859-1,codepage=850,umask=0 0 0

/dev/hda5 /mnt/data vfat iocharset=iso8859-1,codepage=850,umask=0 0 0

none /mnt/floppy supermount dev=/dev/fd0,fs=auto,--,iocharset=iso8859-1,sync,codepage=850,umask=0 0 0

none /mnt/zip supermount dev=/dev/sda4,fs=auto,--,iocharset=iso8859-1,codepage=850,umask=0 0 0

none /proc proc defaults 0 0

/dev/hda7 swap swap defaults 0 0
```

fsck command



- *Filesystems* can become damage or inconsistent in a number of ways (power fails, kernel panics,...)
- The 5 most common types of damage are:
 - ▶ Unreference inodes
 - ▶ Inexplicably large link counts
 - ▶ Unused data blocks not recorded in the block maps
 - ▶ Data blocks listed as free that are also used in a file
 - ▶ Incorrect summary information in the superblock
- **fsck** command can safely and automatically fix these problems

fsck command (2)



- Disks are normally checked at boot time with **fsck -p**, which examines all local *filesystem* listed in **fstab** file
- If some form of journaling is enabled, **fsck** simply rolls up the log to the last consistent state

NOTE:

- ▶ **fsck** is simply a front-end for the various file system checkers (**fsck.fstype**) available under Linux
- ▶ **e2fsck** is the version of **fsck** for the **ext2 filesystem** (also supports **ext3**)

Making a filesystem



- The tool used for partitioning the disk is **fdisk**
 - ▶ It is interactive (pressing **m** displays a list of all its commands)
- *Filesystem* is created with the **mkfs** command
 - ▶ It is just a front end that runs the appropriate program depending on the desired filesystem type
 - ▶ The type is selected with the **-t** option
- New *filesystem* can be mounted as soon as its *mount point* is created with **mkdir** command

NOTE:

- ▶ **mke2fs** is the version of **mkfs** for the **ext2 filesystem**
- ▶ **mke2fs -j** creates an **ext3 filesystem**



- Some tools are useful for managing *filesystems*:
 - ▶ **df** shows the free disk space on one or more *filesystems*
 - ▶ **du** shows how much disk space a directory and all its files contain
 - ▶ **sync** forces all unwritten blocks in the buffer cache to be written to disk (the daemon process **update** does this automatically)

Other tools for the *ext2* filesystem



- In addition to the *filesystem* creator (**mke2fs**) and checker (**e2fsck**) the *ext2 filesystem* has some additional tools that can be useful:
 - ▶ **tune2fs** adjusts *filesystem* parameters
It can convert, with **-j** option, an *ext2* to an *ext3 filesystem*
 - ▶ **dumpe2fs** shows information about an *ext2 filesystem*, mostly from the superblock
 - ▶ **debugfs** is a *filesystem* debugger. It allows direct access to the *filesystem* data structures stored on disk and can thus be used to repair a disk that is so broken that **fsck** can't fix it automatically
It has also been known to be used to recover deleted files

X-based Interfaces



General browsable tool: **Webmin** (<https://localhost:10000/>)

The screenshot shows two windows from the Webmin interface. The left window, titled 'Disk and Network Filesystems', displays a table of mounted filesystems. The right window, titled 'Edit Mount', shows configuration options for a Windows 95 Filesystem.

Mounted As	Type	Location
/	New Linux Native Filesystem	IDE device A
/dev/pts	PTS Filesystem	none
/home	New Linux Native Filesystem	IDE device A
/mnt/cdrom	SUPERMOUNT	none
/mnt/cdrom2	SUPERMOUNT	none
/mnt/data	Windows 95 Filesystem	IDE device A
/mnt/floppy	SUPERMOUNT	none
/mnt/winme	Windows 95 Filesystem	IDE device A
/mnt/zip	SUPERMOUNT	none
/proc	Kernel Filesystem	proc
/Virtual Memory	Virtual Memory	IDE device A
/proc/bus/usb	USB Devices	none
/dev	DEVFS	none

Windows 95 Filesystem Mount Details

Mounted As: /mnt/data Size: 40955680 kB / Free: 34974560 kB

Save Mount? Save and mount at boot Save Don't save

Mount now? Mount Unmount

Windows 95 Filesystem: Disk Floppy disk 0

Advanced Mount Options

Read-only? Yes No

Allow device files? Yes No

Disallow setuid programs? Yes No

User files are owned by: []

File naming rules: Default

File permissions mask: Default 0

Translate Unicode characters? Yes No

Buffer writes to filesystem? Yes No

Allow execution of binaries? Yes No

Allow users to mount this filesystem? Yes No

Group files are owned by: []

DOS-Unix newline conversion: None

Report errors on chown attempts? Yes No

Allow names that differ only in case? Yes No

X-based Interfaces (2)



General tool: **Linuxconf**

The screenshot shows three windows from the Linuxconf interface. The 'Filesystem configurator' window has buttons for 'Access local drive', 'Access nfs volume', and 'Configure swap files and partitions'. The 'Local volume' window shows a table of mounted volumes. The 'Volume specification' window is open, showing options for configuring a volume.

Source	Mount point	Filesystem	Size	Options	Status
/dev/hda9	/	ext3	1992M	Linux(83)	Mounted
/dev/hda8	/home	ext3	1992M	Linux(83)	Mounted
/dev/hda5	/mnt/data	vfat	40005M	Dos (0b)	Mounted
/dev/hda1	/mnt/winme	vfat	10001M	Dos (0b)	Mounted

Volume specification

You must enter the specification of a volume or partition and the position (mount point) where you want to install this volume in the directory structure of this workstation

Base | Options | Dos options | Misc |

Other options: iocharset=iso8859-1,codepage=850

Comment: []

Buttons: Accept, Cancel, Del, Mount, Unmount, Help

X-based Interfaces (3)



Specific KDE tool to mount and unmount disk partitions:
KDiskFree

Icon	Device	Type	Size	Mount point	Free	Full %	Usage
	/dev/hda1	vfat	9.8 GB	/mnt/winme	7.5 GB	23.1%	
	/dev/hda5	vfat	39.1 GB	/mnt/data	33.4 GB	14.6%	
	/dev/hda8	ext3	1.9 GB	/home	1.9 GB	1.8%	
	/dev/hda9	ext3	4.8 GB	/	3.4 GB	29.0%	

Appendix - File types



- Linux defines seven types of files:
 - ▶ Regular files
 - ▶ Directories
 - ▶ Character device files
 - ▶ Block device files
 - ▶ Local domain sockets
 - ▶ Named pipes (FIFOs)
 - ▶ Symbolic links
- **ls -ld** command shows the type of an existing file
In fact the first character of the **ls** output encodes the type:
- regular, **d** directory, **c** character, **b** block, **s** socket, **p** pipe, **l** link
- Different files are created by different tools but **rm** command is the universal tool for deleting files

Appendix - File types (2)



- *Regular file*: is just a bag o' bytes
- *Directory*: contains named reference to other files. More than one directory can refer to a file at one time, and the reference can have different names (*hard link*)

NOTE:

you can create directories with `mkdir` command and *hard links* with `ln` command

- *Character and Block device file*: allow programs to communicate with the system's hardware and peripherals

NOTE:

you can create device files with `mknod` command

Appendix - File types (3)



- *Local domain sockets*: are connections between processes that allow them to communicate in a hygienic manner. They cannot be read from or written to by processes not involved in the connection
- *Named pipes (FIFOs)*: allow communication between two processes running on the same host
- *Symbolic link*: points to a file by name
 - ▶ Hard link is a direct reference, whereas a symbolic link is a reference by name
 - ▶ Symbolic link are distinct from the files they point to

NOTE:

you can create a symbolic link with `ln -l` command

Reference



- “UNIX System Administration Handbook Third Edition” by Evi Nemeth, Garth Snyder, Scott Seebass, Trent R. Hein
- “Linux Administration Handbook” by Evi Nemeth, Garth Snyder, Trent R. Hein
- “Linux: The Complete Reference, Fifth Edition” by Richard L. Petersen
- “The Linux System Administrator's Guide Version 0.7” by Lars Wirzenius, Joanna Oja, Stephen Stafford (<http://www.tldp.org/guides.html>)
- Filesystems HOWTO (<http://www.tldp.org/>)

System Commands



- `mount` - mount a file system
- `umount` - unmount file systems
- `fdisk` - Partition table manipulator for Linux
- `mkfs` - build a Linux file system
- `mke2fs` - create a Linux second extended file system
- `tune2fs` - adjust tunable filesystem parameters on second extended filesystems

System Commands (2)



- `dumpe2fs` - dump filesystem information
- `fsck` - check and repair a Linux file system
- `e2fsck` - check a Linux second extended file system
- `df` - report filesystem disk space usage
- `du` - estimate file space usage
- `debugfs` - ext2 file system debugger

System Commands (3)



- `fuser` - identify processes using files or sockets
- `swapon`, `swapoff` - enable/disable devices and files for paging and swapping
- `ls` - list directory contents
- `mknod` - make block or character special files
- `mkdir` - make directories
- `sync` - flush filesystem buffers

System Commands (4)



- `rm` - remove files or directories
- `ln` - make links between files