1.2 Files architecture

One of the fundamental criteria of a UNIX system is that everything is a file, with the only exception of the interfaces of network devices.

1.2.1 The Virtual File System and the characteristics of the files

On a UNIX system, all files are the same, plus extensions are just conventions and mean nothing to the kernel, which reads the data the same way.

Actually the system expects different types of files, but in another sense.

These are:

- Regular file: a file that contains data (what is usually meant by a file). It is indicated with "-";
- Directory: a file that contains a list of names associated with inodes. It is indicated with "d";
- Symbolic link: a file that contains a reference to another file or directory. It is indicated with "l";
- Char device: a file that identifies a character access device. It is indicated with "c";
- Block device: a file that identifies a block access device. It is indicated with "b";
- Fifo or pipe: a special file that identifies a one-way communication line. It is indicated with "p";
- Socket: a special file that identifies a two-way communication line. It is indicated with "s".

The fifos and the sockets are communication channels made available to the kernel to let the processes communicate with each other, they are two different ways to realize the communication between processes:

- Fifo: one process can write on it and another process that has opened it will read on the other end what the first one has written, nothing will be saved on disk, but it will all pass through the kernel that allows this communication as through a pipe;
- Socket: they do the same thing but allow two-way communication, where the second process can write back, and the first can read.

The main file operations are:

- Open: opens the file;
- Read: read the file;
- Write: writes to file;
- Llseek: moves within the file;
- Ioctl: access control operations;

- Readdir: reads the contents of a directory.

One of the basic programs for managing files is **ls** (**lis**t files), which displays (with no arguments) the list of all files in the current directory.

Using the -l option it is possible to obtain an extended list, with the various properties of the files. The information is:

- File type: the first character of the first column;
- File permissions: the rest of the first column;
- Number of Hard Links to the file: the second column;
- The user and the owner group of the file: in the third and fourth columns;
- Size (in bytes): in the fifth column;
- Last file modification: in the sixth column;
- File name: in the last column.

Other options of the **ls** command are:

- -a: show invisible files;
- -i: write the inode number;
- -R: execute the list recursively for all subdirectories;
- -c: use the time of last file change, that is the changes on file properties;
- -u: use the last access time to the file;
- -d: it only displays the name and not the content when referring to a directory, and does not execute symbolic links.

To operate on the associated times of a file, use the command **touch**, and takes various options:

- -a: change the last access time;
- -m: change the time of last modification;
- -d: to specify a date;
- -r: to get values from another file.

1.2.2 The architecture of a filesystem and the properties of files

The structure that uniquely identifies a single file within the filesystem is the so-called inode: each file is associated with an inode in which all the information concerning it is kept (the information that ls provides comes from the inode), the only information not contained is the name of the file.

You can have multiple entries in different directories pointing to the same inode. This introduces the concept of **Hard Link** (direct link): two files pointing to the same inode are actually the same file, no specific property allows them to be distinguished, as access for both is through the same inode.

Since the same inode can be referenced in multiple directories, a file can have multiple names, even completely unrelated to each other. For this reason each inode keeps a counter that indicates the reference number that has been made to it (**link count**), it is reported in the second column of the **ls -l** command.

The generic command to create a link is **ln** (**lin**k), and it takes as arguments the original file and the name of the new link. In UNIX systems, links are of two types:

- •Hard Link: those just illustrated;
- •Symbolic Link: which are the most similar to Windows shortcuts. A limitation of Hard Links is that you can only refer to an inode present in the same filesystem as the directory. To overcome this limitation, symbolic links have been created, which are created with the **ln** command using the **-s** option (the link will have a different inode than the original file).

Other options of the **ln** command:

- •-f: forces overwriting of the new file if it already exists;
- •-i: requests confirmation in case of overwriting;
- •-d: create a hard link to a directory (not usable in Linux);
- •-b: backs up the destination if it already exists.

A second feature of symbolic links is that they can also create links for directories and, if you delete the link, the file will not be deleted (unlike hard links).

The command to remove files is **rm** (**rem**ove file), but the function to delete in link is actually **unlink**.

Only when the number of references of an inode is canceled, is the data in the file actually removed from the kernel disk. In case a process is using the file at the time of removal, the disk space will not be released, and as long as the process has not finished the data will remain, even if only for it, available.

This statement is partially true, as with the deletion the data of a file are not touched, only the space occupied by them is declared available. You need to use **shred** to work around this as, in addition to deleting the reference, it overwrites the file data.

The **rm** command takes as arguments the list of files to be deleted, used with the option:

- •-i: confirmation is requested;
- •-f: it cancels any previous -i and no errors are printed for non-existent files;
- •-r (or -R): allows the recursive deletion of a directory and all its contents, is to be used with caution (especially when combined with -f).

The command used for moving files is **mv** (**m**o**v**e file).

If the file is moved without changing the filesystem, there is no need to move the contents of the file and just create a new entry for the inode in question by removing the old one.

When you have to move to a different filesystem it becomes necessary to copy the contents first and then delete the original (you will have to deal with a different inode).

The **mv** command has two forms, it can take as arguments a list of files followed by a directory, to move all the files within the directory, or two files (or two directories), to rename the first to the second. The main options are:

- -i: requires confirmation in case of overwriting;
- -f: force overwrite if file already exists;
- -u: moves only if the destination is older than the source (or does not exist);
- -b: backs up the destination if it already exists.

If the move occurs within the same filesystem, the times of the file are not changed.

When you want to duplicate a file, the command to use is **cp** (**cop**y file). Like **mv** the command can take a list of files followed by a directory, to copy the list of files into the directory, or two files (or directories), to copy the first to the second. The main options are:

- -f: force overwrite the destination if it already exists;
- -i: requires confirmation in case of overwriting;
- -p: preserve the times, permissions and owners of the file (otherwise it modifies them);
- -l: create hard links instead of copies;
- -s: create symbolic links instead of copies;
- -d: copy the symbolic link instead of the file indicated by it;
- -r: recursively copies all the contents of a directory;
- -a: combine options -dpr;
- -L: always follows symbolic links;
- -b: backs up the destination if it already exists;
- -u: copy only if the destination is older than the source (or does not exist).

The convention used in all UNIX-like systems is that file names are indicated with a pathname or path, which is the path that must be taken in the tree to reach files through directories, where the directory names are separated from each other with "/".

The path can be indicated in the following way:

- Absolute: ie starting from the root directory, so the pathname will start with /;
- Relative: therefore starting from the current working directory.

The working directory can be changed with the **cd** (**c**hange **d**irectory) command followed by the directory pathname. To find out what the current working directory is, use the **pwd** (**p**rint **w**orking **d**irectory) command.

Each directory will always contain at least two directories:

- .: which refers to itself;
- ..: which refers to the directory in which the current one is contained.

Since the name starts with '.' they are invisible.

The command for creating directories is **mkdir** (**m**a**k**e **dir**ectory) with argument a list of directories. To create all directories that do not exist within the pathname use the **-p** option, you can also set permissions with the **-m** option.

To remove directories, use the **rmdir** (**rem**ove **dir**ectory) command with argument the list of empty directories to be removed. If the directories are not empty the command fails. With the **-p** option you delete the entire path of the directories (which must be empty).

1.2.3 Filesystem Hierarcy Standard

At boot the kernel mounts everything called the root directory of the tree, which is denoted by "/", which is not contained in any directory.

The **Hierarcy Standard Filesystem** describes in detail the structure of the tree. Directories are divided according to criteria:

- First criterion: Possibility to contain files whose content can be modified (read / write) or not (read);
- Second criterion: Ability to contain files such as system programs that can be shared between multiple workstations or files that are local and specific to the machine in question;
- Third criterion: possibility of containing or not commands and files (configurations and device files) that are needed to boot the system, and which therefore must be located on the filesystem used for the root directory, since they would not be available if placed in different file systems (as they can only be mounted after system startup).

The standard stipulates that there must be subdirectories of / which are:

- /bin: contains essential system commands, which must be available even when there are no other filesystems mounted besides the root one, for example at boot or when in single user mode. It must have no subdirectories and cannot reside on a filesystem other than the root's;
- /boot: it contains all the files necessary for the boot process (images of the kernel, bootloader, ramdisk, etc ...) except the configuration files and the programs for setting up the boot process, which go to /sbin. It can fit on any filesystem as long as it is visible to the bootloader
- /dev: contains device files, which allow access to peripherals. It originally had to be on the same filesystem as the root, as device files are needed for the boot process. Today the content is in most cases dynamically generated and is mounted on a temporary filesystem which is populated at boot time via **udev**;
- /etc: contains system configuration files and startup scripts. It must not contain binary programs and cannot reside on a filesystem other than the root one. The files can in turn be grouped into directories; the standard provides that, if installed, the directories **/etc/opt** (optional packages), **/etc/X11** (for the X Window configuration), **/etc/sgml** (for SGML configuration) and **/etc/xml** (for XML configuration) are present;

•/home: contains the users home directory, the only part of the filesystem (except **/tmp** and **/var/tmp**) that users have write rights to, which is used to keep their personal files. It can be mounted on any filesystem;

•/lib: contains essential shared libraries, used by **/bin** and **/sbin** programs, and must be on the same filesystem as the root. If a modular kernel is installed the modules must be installed in **/lib/modules**;

•/media: contains mountpoints for removable devices. In older distributions, floppies and CDROMs have dedicated directories, but with the introduction of automatic detection mechanisms for removable devices, this directory has been defined in which the appropriate subdirectories are created on which they are then mounted in a manner automatic;

•/mnt: contains mountpoints for temporary mounts for use by the system administrator, filesystems of permanent devices with removable media such as floppies, and CDROMs that were previously kept both in this directory and directly under / must be moved under **/media**. It is normally empty and must be created directly under the root;

•/opt: contains any additional software packages. It can be under any filesystem. A package must install in the **/opt/package** directory where package is the name of the package. The administrator is allowed to use some optional directories: **/opt/bin**, **/opt/doc**, **/opt/include**, **/opt/info**, **/opt/lib** and **/opt/man**. Variable files pertaining to the above packages must be installed in **/var/opt** and the configuration files in **/etc/opt**, no package pertaining files must be installed outside these directories;

•/proc: is the standard mountpoint of the virtual proc filesystem. This is a special filesystem that allows you to access a whole series of internal kernel variables (related to parameters and settings of all kinds) with the files interface. So if you want information about the interrupts and the DMA channels used by the system, you can read the **/proc/interrupts** and **/proc/dma** files, while you can set various system characteristics by writing to the files under **/proc/sys**;

•/root: is the administrator's home directory. Normally you keep it in the same filesystem as the root; its use is optional but this is the recommended location;

•/run: present only in the most recent distributions, it is used to make available from the start a directory on which a temporary filesystem is mounted where the so-called volatile run-time data are kept (such as the files that record the PIDs of the service programs, lock files and other files whose use mainly concerns the execution of programs) which do not need to be maintained through a restart. The directory allows you to go a place where even startup programs can keep their data, since in general **/var/run** and **/var/lock** may not be available when **/var** is kept on a separate filesystem. It also allows you to unify all this data in a single directory and on a single temporary filesystem, since if the contents of /var/run and /var/lock are used, they are returned to it;

•/sbin: contains essential programs for use by the system administrator (such as init and fsck). It must be on the root filesystem. Only the essential programs for booting the system, recovering and maintaining the filesystem should be placed in this directory;

•/srv: it was introduced to keep the data relating to the various services that may have been installed on a machine (such as the pages served by a webserver) that were previously installed directly under **/var**. A further subdivision is not defined, but generally we tend to create a subdirectory for each service;

•/tmp: it is used to keep temporary files. It is cleared on each reboot, and programs do not have to assume that files are kept between two successive runs;

•/usr: is the root directory that contains all programs, files and non-variable data, which can also be shared among multiple workstations. It can be mounted on a separate filesystem from / and can be mounted read-only, although this choice may make it difficult to use some environments. It provides a further directory hierarchy in which the various files must be organized; the standard requires the following subdirectories:

> •bin: contains user-used programs installed directly from the system (or from the original distribution). It cannot be shared further;
>
> •include: contains all library function declaration files used by the compiler and by C and C++ programs;
>
> •lib: contains the libraries, object and binary files related to the **/bin** and **/sbin** programs. It can contain subdirectories with all files relating to individual programs;
>
> •local: contains a replica of the **/usr** hierarchy dedicated to files installed locally by the administrator. Generally, programs compiled from source and everything that is not part of the official distribution are installed here;
>
> •sbin: it contains the programs for the management of the system for the use of the administrator not essential to the start
>
> •share: it contains a hierarchy in which all files and data that do not depend on the hardware architecture are organized: the **man** (for man page) and **misc** directories are always required; if installed you can also find a series of directories such as: **dict** (for dictionaries), **doc** (for documentation), **games** (for the static data of the games), **info** (for the files of the relative help system), **terminfo** (for the database with database information) and **zoneinfo** (for time zone data). Anything not classified in the other optional directories must be placed in **misc**;
>
> •src: its use is optional and contains the package sources for reference;

•/var: contains variable files, spool directories, log files, transient and temporary data, so that **/usr** can be mounted read-only. It is preferable to mount it on a separate filesystem and some directories cannot be shared. Also in this case the files are organized in a further standardized hierarchy which foresees the presence of the following subdirectories:

> •cache: contains support and temporary storage data for applications;
>
> •lib: contains status information and application data;
>
> •local: contains variable data relating to packages in **/usr/local**;
>
> •lock: contains lock files, if used **/run** is a symbolic link to **/run/lock**;
>
> •log: contains application log files;
>
> •opt: contains variable files for packages in **/opt**;

- spool: contains directories for queues containing application transit data (print jobs, e-mails, etc ...)
- tmp: contains temporary files not deleted when the system is restarted.