**IT5423**

**Module 2: File systems**

**Learning objectives**:

* Explain file systems in operating systems
* Apply file related commands to access and modify files

Contents

Contents

[**File System** 1](#_Toc500164454)

[**Directory Structure** 2](#_Toc500164455)

[**Inode** 4](#_Toc500164456)

[How do I see file inode number? 6](#_Toc500164457)

[**Types of files** 6](#_Toc500164458)

[**Directory** 10](#_Toc500164459)

[**Understanding UNIX / Linux symbolic (soft) and hard links** 11](#_Toc500164460)

[**Navigating the File System** 13](#_Toc500164461)

[**The df Command** 15](#_Toc500164462)

[**The du Command** 16](#_Toc500164463)

[**Mounting the File System** 17](#_Toc500164464)

[**Unmounting the File System** 18](#_Toc500164465)

[**User and Group Quotas** 18](#_Toc500164466)

[**References** 19](#_Toc500164467)

# **File System**

A file system is a logical collection of files on a partition or disk. A partition is a container for information and can span an entire hard drive if desired.

Your hard drive can have various partitions which usually contain only one file system, such as one file system housing the **/file system** or another containing the **/home file system**.

One file system per partition allows for the logical maintenance and management of differing file systems.

Everything in Unix is considered to be a file, including physical devices such as DVD-ROMs, USB devices, and floppy drives.

# **Directory Structure**

Unix uses a hierarchical file system structure, much like an upside-down tree, with root (/) at the base of the file system and all other directories spreading from there.

A Unix filesystem is a collection of files and directories that has the following properties −

* It has a root directory (**/**) that contains other files and directories.
* Each file or directory is uniquely identified by its name, the directory in which it resides, and a unique identifier, typically called an **inode**.
* By convention, the root directory has an **inode** number of **2** and the **lost+found** directory has an **inode** number of **3**. Inode numbers **0**and **1** are not used. File inode numbers can be seen by specifying the **-i option** to **ls command**.
* It is self-contained. There are no dependencies between one filesystem and another.

The directories have specific purposes and generally hold the same types of information for easily locating files. Following are the directories that exist on the major versions of Unix −

|  |  |
| --- | --- |
| **S.No.** | **Directory & Description** |
| 1 | **/**  This is the root directory which should contain only the directories needed at the top level of the file structure |
| 2 | **/bin**  This is where the executable files are located. These files are available to all users |
| 3 | **/dev**  These are device drivers |
| 4 | **/etc**  Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages |
| 5 | **/lib**  Contains shared library files and sometimes other kernel-related files |
| 6 | **/boot**  Contains files for booting the system |
| 7 | **/home**  Contains the home directory for users and other accounts |
| 8 | **/mnt**  Used to mount other temporary file systems, such as **cdrom** and **floppy** for the **CD-ROM** drive and **floppy diskette drive**, respectively |
| 9 | **/proc**  Contains all processes marked as a file by **process number** or other information that is dynamic to the system |
| 10 | **/tmp**  Holds temporary files used between system boots |
| 11 | **/usr**  Used for miscellaneous purposes, and can be used by many users. Includes administrative commands, shared files, library files, and others |
| 12 | **/var**  Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data |
| 13 | **/sbin**  Contains binary (executable) files, usually for system administration. For example, ***fdisk*** and ***ifconfig*** utlities |
| 14 | **/kernel**  Contains kernel files |

# **Inode**

Inodes store information about files and directories (folders), such as file ownership, access mode (read, write, execute permissions), and file type. On many types of file system implementations, the maximum number of inodes is fixed at file system creation, limiting the maximum number of files the file system can hold. A typical allocation heuristic for inodes in a file system is one percent of total size.

The inode number indexes a table of inodes in a known location on the device. From the inode number, the kernel's file system driver can access the inode contents, including the location of the file - thus allowing access to the file.

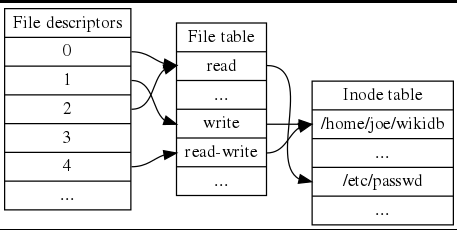
A file's inode number can be found using the ls -i command. The ls -i command prints the i-node number in the first column of the report.

Some Unix-style file systems such as [ReiserFS](https://en.wikipedia.org/wiki/ReiserFS" \o "ReiserFS) omit an inode table, but must store equivalent data in order to provide equivalent capabilities. The data may be called stat data, in reference to the [stat](https://en.wikipedia.org/wiki/Stat_(Unix)) [system call](https://en.wikipedia.org/wiki/System_call) that provides the data to programs.

File names and directory implications:

* Inodes do not contain its hardlink names, only other file metadata.
* Unix directories are lists of association structures, each of which contains one filename and one inode number.
* The file system driver must search a directory looking for a particular filename and then convert the filename to the correct corresponding inode number.

The operating system kernel's in-memory representation of this data is called struct inode in [Linux](https://en.wikipedia.org/wiki/Linux). Systems derived from [BSD](https://en.wikipedia.org/wiki/BSD) use the term vnode, with the **v** of **vnode** referring to the kernel's [virtual file system](https://en.wikipedia.org/wiki/Virtual_file_system) layer.



The **inode (index node)** is a fundamental concept in the Linux and UNIX filesystem. Each object in the filesystem is represented by an inode. But what are the objects? Let us try to understand it in simple words. Each and every file under Linux (and UNIX) has following attributes:

=> File type (executable, block special etc)  
=> Permissions (read, write etc)  
=> Owner  
=> Group  
=> File Size  
=> File access, change and modification time (remember UNIX or Linux never stores file creation time, this is favorite question asked in UNIX/Linux sys admin job interview)  
=> File deletion time  
=> Number of links (soft/hard)  
=> Extended attribute such as append only or [no one can delete file](https://www.cyberciti.biz/tips/linux-password-trick.html) including [root user (immutability)](https://www.cyberciti.biz/tips/howto-write-protect-file-with-immutable-bit.html)  
=> Access Control List (ACLs)

All the above information stored in an inode. In short the inode identifies the file and its attributes (as above) . Each inode is identified by a unique inode number within the file system. Inode is also know as index number.

## How do I see file inode number?

You can use ls -i command to see inode number of file  
$ ls -i /etc/passwd  
Sample Output

32820 /etc/passwd

You can also use stat command to find out inode number and its attribute:  
$ stat /etc/passwdOutput:

File: `/etc/passwd'

Size: 1988 Blocks: 8 IO Block: 4096 regular file

Device: 341h/833d Inode: 32820 Links: 1

Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)

Access: 2005-11-10 01:26:01.000000000 +0530

Modify: 2005-10-27 13:26:56.000000000 +0530

Change: 2005-10-27 13:26:56.000000000 +0530

# **Types of files**

Most files are just files, called regular files; they contain normal data, for example text files, executable files or programs, input for or output from a program and so on.

While it is reasonably safe to suppose that everything you encounter on a Linux system is a file, there are some exceptions.

* Directories: files that are lists of other files.
* Special files: the mechanism used for input and output. Most special files are in /dev, we will discuss them later.
* Links: a system to make a file or directory visible in multiple parts of the system's file tree. We will talk about links in detail.
* (Domain) sockets: a special file type, similar to TCP/IP sockets, providing inter-process networking protected by the file system's access control.
* Named pipes: act more or less like sockets and form a way for processes to communicate with each other, without using network socket semantics.

The -l option to **ls** displays the file type, using the first character of each input line:

|  |
| --- |
| jaime:~/Documents> **ls -l**  total 80  -rw-rw-r-- 1 jaime jaime 31744 Feb 21 17:56 intro Linux.doc  -rw-rw-r-- 1 jaime jaime 41472 Feb 21 17:56 Linux.doc  drwxrwxr-x 2 jaime jaime 4096 Feb 25 11:50 course |

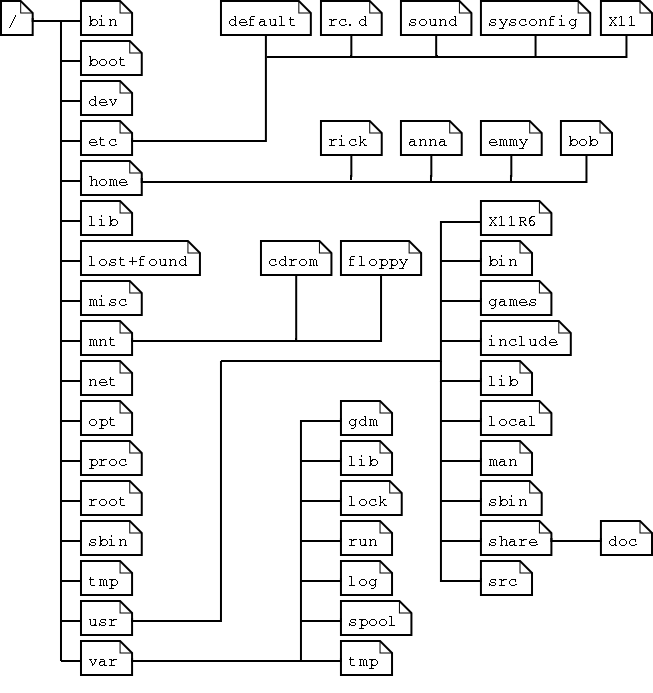
This table gives an overview of the characters determining the file type:

**Table 3-1. File types in a long list**

| **Symbol** | **Meaning** |
| --- | --- |
| - | Regular file |
| d | Directory |
| l | Link |
| c | Special file |
| s | Socket |
| p | Named pipe |
| b | Block device |

For convenience, the Linux file system is usually thought of in a tree structure. On a standard Linux system you will find the layout generally follows the scheme presented below.

**Figure 3-1. Linux file system layout**



This is a layout from a RedHat system. Depending on the system admin, the operating system and the mission of the UNIX machine, the structure may vary, and directories may be left out or added at will. The names are not even required; they are only a convention.

The tree of the file system starts at the trunk or slash, indicated by a forward slash (/). This directory, containing all underlying directories and files, is also called the root directory or "the root" of the file system.

Directories that are only one level below the root directory are often preceded by a slash, to indicate their position and prevent confusion with other directories that could have the same name. When starting with a new system, it is always a good idea to take a look in the root directory. Let's see what you could run into:

|  |
| --- |
| emmy:~> **cd /**  emmy:/> **ls**  bin/ dev/ home/ lib/ misc/ opt/ root/ tmp/ var/  boot/ etc/ initrd/ lost+found/ mnt/ proc/ sbin/ usr/ |

**Table 3-2. Subdirectories of the root directory**

| **Directory** | **Content** |
| --- | --- |
| /bin | Common programs, shared by the system, the system administrator and the users. |
| /boot | The startup files and the kernel, vmlinuz. In some recent distributions also grub data. Grub is the GRand Unified Boot loader and is an attempt to get rid of the many different boot-loaders we know today. |
| /dev | Contains references to all the CPU peripheral hardware, which are represented as files with special properties. |
| /etc | Most important system configuration files are in /etc, this directory contains data similar to those in the Control Panel in Windows |
| /home | Home directories of the common users. |
| /initrd | (on some distributions) Information for booting. Do not remove! |
| /lib | Library files, includes files for all kinds of programs needed by the system and the users. |
| /lost+found | Every partition has a lost+found in its upper directory. Files that were saved during failures are here. |
| /misc | For miscellaneous purposes. |
| /mnt | Standard mount point for external file systems, e.g. a CD-ROM or a digital camera. |
| /net | Standard mount point for entire remote file systems |
| /opt | Typically contains extra and third party software. |
| /proc | A virtual file system containing information about system resources. More information about the meaning of the files in proc is obtained by entering the command **man *proc*** in a terminal window. The file proc.txt discusses the virtual file system in detail. |
| /root | The administrative user's home directory. Mind the difference between /, the root directory and /root, the home directory of the root user. |
| /sbin | Programs for use by the system and the system administrator. |
| /tmp | Temporary space for use by the system, cleaned upon reboot, so don't use this for saving any work! |
| /usr | Programs, libraries, documentation etc. for all user-related programs. |
| /var | Storage for all variable files and temporary files created by users, such as log files, the mail queue, the print spooler area, space for temporary storage of files downloaded from the Internet, or to keep an image of a CD before burning it. |

How can you find out which partition a directory is on? Using the **df** command with a dot (.) as an option shows the partition the current directory belongs to, and informs about the amount of space used on this partition:

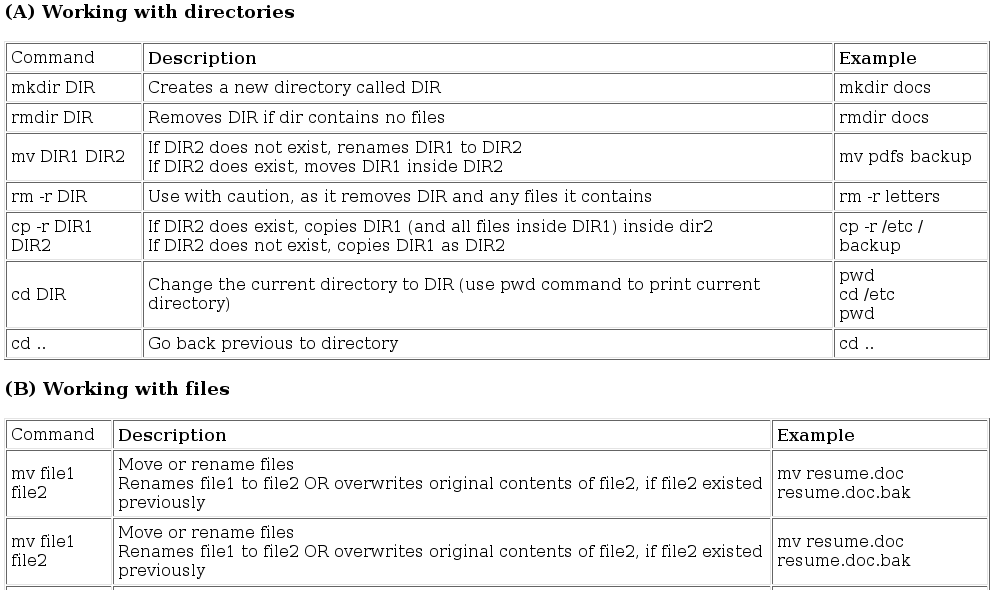
|  |
| --- |
| sandra:/lib> **df -h .**  Filesystem Size Used Avail Use% Mounted on  /dev/hda7 980M 163M 767M 18% / |

As a general rule, every directory under the root directory is on the root partition, unless it has a separate entry in the full listing from **df** (or **df -h** with no other options).

# **Directory**

A directory contained inside another directory is called a subdirectory. At the end the directories form a tree structure. Use tree command to see directory tree structure:  
$ tree /etc | less  
Again a directory has an inode just like a file. It is a specially formatted file containing records which associate each name with an inode number. Please note the following limitation of directories under ext2/3 file system:

* There is an upper limit of 32768 subdirectories in a single directory.
* There is a “soft” upper limit of about 10-15k files in a single directory

However according to official documentation of ext2/3 file system points that â€œUsing a hashed directory index (which is under development) allows 100k-1M+ files in a single directory without performance problems’. Here are my two favorite alias commands related to directory :  
$ alias ..='cd ..'  
alias d='ls -l | grep -E "^d"'  


# **Understanding UNIX / Linux symbolic (soft) and hard links**

Inodes are associated with precisely one directory entry at a time. However, with hard links it is possible to associate multiple directory entries with a single inode. To create a hard link use ln command as follows:  
# ln /root/file1 /root/file2  
# ls -l  
Above commands create a link to file1. Symbolic links refer to:

A symbolic path indicating the abstract location of another file.

Hard links refer to:

The specific location of physical data.

**Hard link vs. Soft link in Linux or UNIX**

* Hard links cannot link directories.
* Cannot cross file system boundaries.

Soft or symbolic links are just like hard links. It allows to associate multiple filenames with a single file. However, symbolic links allows:

* To create links between directories.
* Can cross file system boundaries.

These links behave differently when the source of the link is moved or removed.

* Symbolic links are not updated.
* Hard links always refer to the source, even if moved or removed.

**How do I create symbolic link?**

You can create symbolic link with ln command:  
$ ln -s /path/to/file1.txt /path/to/file2.txt  
$ ls -ali  
Above command will create a symbolic link to file1.txt.

**Symbolic link creation and deletion**

Let us create a directory called foo, enter:  
$ mkdir foo  
$ cd foo  
Copy /etc/resolv.conf file, enter:  
$ cp /etc/resolv.conf .  
View inode number, enter:  
$ ls -ali  
Sample output:

total 152

1048600 drwxr-xr-x 2 vivek vivek 4096 2008-12-09 20:19 .

1015809 drwxrwxrwt 220 root root 143360 2008-12-09 20:19 ..

1048601 -rwxr-xr-x 1 vivek vivek 129 2008-12-09 20:19 resolv.conf

Now create soft link to resolv.conf, enter:  
$ ln -s resolv.conf alink.conf  
$ ls -ali  
  
Sample output:

total 152

1048600 drwxr-xr-x 2 vivek vivek 4096 2008-12-09 20:24 .

1015809 drwxrwxrwt 220 root root 143360 2008-12-09 20:19 ..

1048602 lrwxrwxrwx 1 vivek vivek 11 2008-12-09 20:24 alink.conf -> resolv.conf

1048601 -rwxr-xr-x 1 vivek vivek 129 2008-12-09 20:19 resolv.conf

The reference count of the directory has not changed (total 152). Our symbolic (soft) link is stored in a different inode than the text file (1048602). The information stored in resolv.conf is accessible through the alink.conf file. If we delete the text file resolv.conf, alink.conf becomes a broken link and our data is lost:  
$ rm resolv.conf  
$ ls -ali  
If alink.conf was a hard link, our data would still be accessible through alink.conf. Also, if you delete the soft link itself, the data would still be there. Read man page of ln for more information.

# **Navigating the File System**

Now that you understand the basics of the file system, you can begin navigating to the files you need. The following commands are used to navigate the system −

|  |  |
| --- | --- |
| **S.No.** | **Command & Description** |
| 1 | **cat filename**  Displays a filename |
| 2 | **cd dirname**  Moves you to the identified directory |
| 3 | **cp file1 file2**  Copies one file/directory to the specified location |
| 4 | **file filename**  Identifies the file type (binary, text, etc) |
| 5 | **find filename dir**  Finds a file/directory |
| 6 | **head filename**  Shows the beginning of a file |
| 7 | **less filename**  Browses through a file from the end or the beginning |
| 8 | **ls dirname**  Shows the contents of the directory specified |
| 9 | **mkdir dirname**  Creates the specified directory |
| 10 | **more filename**  Browses through a file from the beginning to the end |
| 11 | **mv file1 file2**  Moves the location of, or renames a file/directory |
| 12 | **pwd**  Shows the current directory the user is in |
| 13 | **rm filename**  Removes a file |
| 14 | **rmdir dirname**  Removes a directory |
| 15 | **tail filename**  Shows the end of a file |
| 16 | **touch filename**  Creates a blank file or modifies an existing file or its attributes |
| 17 | **whereis filename**  Shows the location of a file |
| 18 | **which filename**  Shows the location of a file if it is in your PATH |

You can use [Manpage Help](https://www.tutorialspoint.com/unix/unix-manpage-help.htm) to check complete syntax for each command mentioned here.

## **The df Command**

The first way to manage your partition space is with the **df (disk free)**command. The command **df -k (disk free)** displays the **disk space usage in kilobytes**, as shown below −

$df -k

Filesystem 1K-blocks Used Available Use% Mounted on

/dev/vzfs 10485760 7836644 2649116 75% /

/devices 0 0 0 0% /devices

$

Some of the directories, such as **/devices**, shows 0 in the kbytes, used, and avail columns as well as 0% for capacity. These are special (or virtual) file systems, and although they reside on the disk under /, by themselves they do not consume disk space.

The **df -k** output is generally the same on all Unix systems. Here's what it usually includes −

|  |  |
| --- | --- |
| **S.No.** | **Column & Description** |
| 1 | **Filesystem**  The physical file system name |
| 2 | **kbytes**  Total kilobytes of space available on the storage medium |
| 3 | **used**  Total kilobytes of space used (by files) |
| 4 | **avail**  Total kilobytes available for use |
| 5 | **capacity**  Percentage of total space used by files |
| 6 | **Mounted on**  What the file system is mounted on |

You can use the **-h (human readable) option** to display the output in a format that shows the size in easier-to-understand notation.

## **The du Command**

The **du (disk usage) command** enables you to specify directories to show disk space usage on a particular directory.

This command is helpful if you want to determine how much space a particular directory is taking. The following command displays number of blocks consumed by each directory. A single block may take either 512 Bytes or 1 Kilo Byte depending on your system.

$du /etc

10 /etc/cron.d

126 /etc/default

6 /etc/dfs

...

$

The **-h** option makes the output easier to comprehend −

$du -h /etc

5k /etc/cron.d

63k /etc/default

3k /etc/dfs

...

$

## **Mounting the File System**

A file system must be mounted in order to be usable by the system. To see what is currently mounted (available for use) on your system, use the following command −

$ mount

/dev/vzfs on / type reiserfs (rw,usrquota,grpquota)

proc on /proc type proc (rw,nodiratime)

devpts on /dev/pts type devpts (rw)

$

The **/mnt** directory, by the Unix convention, is where temporary mounts (such as CDROM drives, remote network drives, and floppy drives) are located. If you need to mount a file system, you can use the mount command with the following syntax −

mount -t file\_system\_type device\_to\_mount directory\_to\_mount\_to

For example, if you want to mount a **CD-ROM** to the directory **/mnt/cdrom**, you can type −

$ mount -t iso9660 /dev/cdrom /mnt/cdrom

This assumes that your CD-ROM device is called **/dev/cdrom** and that you want to mount it to **/mnt/cdrom**. Refer to the mount man page for more specific information or type mount **-h** at the command line for help information.

After mounting, you can use the cd command to navigate the newly available file system through the mount point you just made.

## **Unmounting the File System**

To unmount (remove) the file system from your system, use the **umount**command by identifying the mount point or device.

For example, **to unmount cdrom**, use the following command −

$ umount /dev/cdrom

The **mount command** enables you to access your file systems, but on most modern Unix systems, the **automount function** makes this process invisible to the user and requires no intervention.

## **User and Group Quotas**

The user and group quotas provide the mechanisms by which the amount of space used by a single user or all users within a specific group can be limited to a value defined by the administrator.

Quotas operate around two limits that allow the user to take some action if the amount of space or number of disk blocks start to exceed the administrator defined limits −

* **Soft Limit** − If the user exceeds the limit defined, there is a grace period that allows the user to free up some space.
* **Hard Limit** − When the hard limit is reached, regardless of the grace period, no further files or blocks can be allocated.

There are a number of commands to administer quotas −

|  |  |
| --- | --- |
| **S.No.** | **Command & Description** |
| 1 | **quota**  Displays disk usage and limits for a user of group |
| 2 | **edquota**  This is a quota editor. Users or Groups quota can be edited using this command |
| 3 | **quotacheck**  Scans a filesystem for disk usage, creates, checks and repairs quota files |
| 4 | **setquota**  This is a command line quota editor |
| 5 | **quotaon**  This announces to the system that disk quotas should be enabled on one or more filesystems |
| 6 | **quotaoff**  This announces to the system that disk quotas should be disabled for one or more filesystems |
| 7 | **repquota**  This prints a summary of the disc usage and quotas for the specified file systems |

# **References**

<https://www.tutorialspoint.com/unix/unix-file-system.htm>

<http://tldp.org/LDP/intro-linux/html/sect_03_01.html>

<https://www.cyberciti.biz/tips/understanding-unixlinux-filesystem-directories.html>