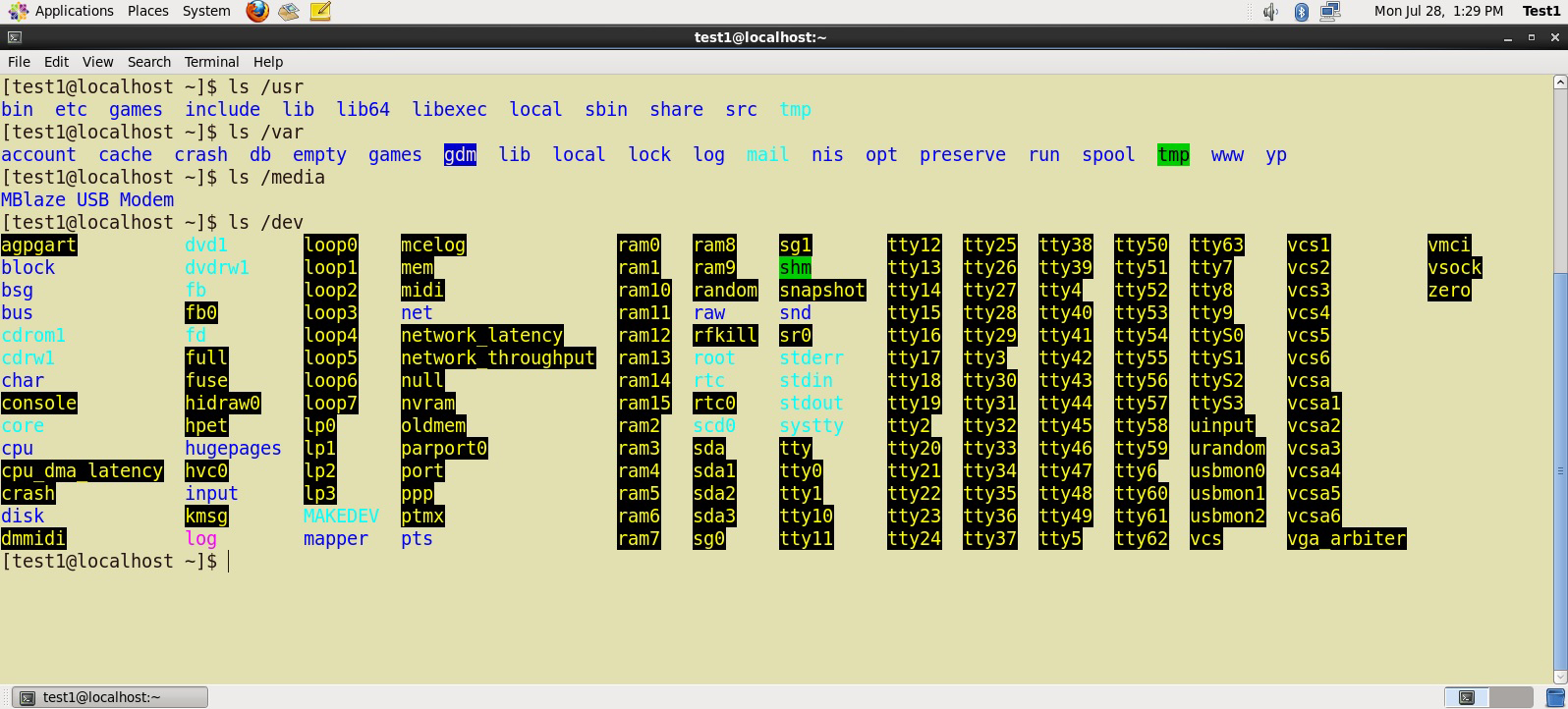
**Filesystem Hierarchy Standard**



The **Filesystem Hierarchy Standard** (**FHS**) grew out of historical standards from early versions of UNIX, such as the **Berkeley Software Distribution** (**BSD**) and others. The FHS provides Linux developers and system administrators with a standard directory structure for the filesystem, which provides consistency between systems and distributions.

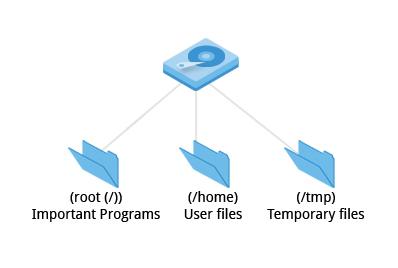
Visit<http://www.pathname.com/fhs/> for a list of the main directories and their contents in Linux systems.

Linux supports various filesystem types created for Linux, along with compatible filesystems from other operating systems such as **Windows** and **MacOS**. Many older, legacy filesystems, such as **FAT**, are supported.

Some examples of filesystem types that Linux supports are:

* 1. **ext3**, **ext4**, **btrfs**, **xfs** (native Linux filesystems)
  2. **vfat**, **ntfs, hfs** (filesystems from other operating systems)

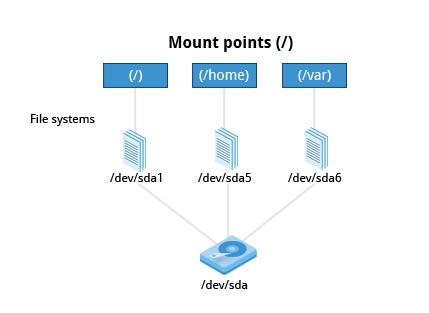
**Partitions in Linux**



Each filesystem resides on a hard disk **partition**. Partitions help to organize the contents of disks according to the kind of data contained and how it is used. For example, important programs required to run the system are often kept on a separate partition (known as root or /) than the one that contains files owned by regular users of that system (/home). In addition, temporary files created and destroyed during the normal operation of Linux are often located on a separate partition; in this way, using all available space on a particular partition may not fatally affect the normal operation of the system.

* [Previous](https://courses.edx.org/courses/LinuxFoundationX/LFS101x/2T2014/courseware/be8401e6c3bb415a92ca42d2132d8cea/550e21f962744f24aff5d856315fe9c1/1#)
* [Next](https://courses.edx.org/courses/LinuxFoundationX/LFS101x/2T2014/courseware/be8401e6c3bb415a92ca42d2132d8cea/550e21f962744f24aff5d856315fe9c1/1#)

**Mount Points**



Before you can start using a filesystem, you need to **mount** it to the filesystem tree at a **mount point.** This is simply a directory (which may or may not be empty) where the filesystem is to be attached (mounted). Sometimes you may need to create the directory if it doesn't already exist.

**Warning:** If you mount a filesystem on a non-empty directory, the former contents of that directory are covered-up and not accessible until the filesystem is unmounted. Thus mount points are usually empty directories.

**More About Mount Points**

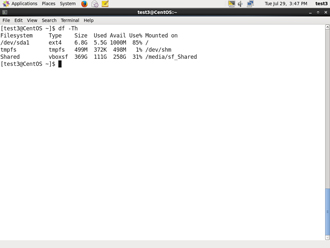
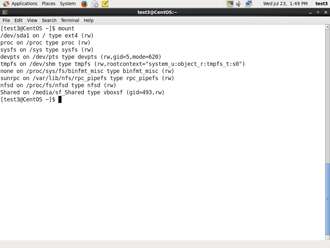
The **mount** command is used to attach a filesystem (which can be local to the computer or, as we shall discuss, on a network) somewhere within the filesystem tree. Arguments include the **device node** and **mount point**. For example,

$ mount /dev/sda5 /home

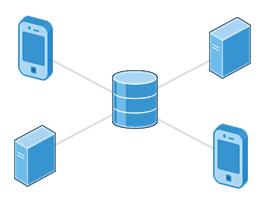
will attach the filesystem contained in the disk partition associated with the /dev/sda5 device node, into the filesystem tree at the /home mount point. (Note that unless the system is otherwise configured only the root user has permission to run **mount**.) If you want it to be automatically available every time the system starts up, you need to edit the file /etc/fstab accordingly (the name is short for **Filesystem Table**). Looking at this file will show you the configuration of all pre-configured filesystems. man fstab will display how this file is used and how to configure it.

Typing **mount** without any arguments will show all presently mounted filesystems.

The command df -Th (**disk-free**) will display information about mounted filesystems including usage statistics about currently used and available space.

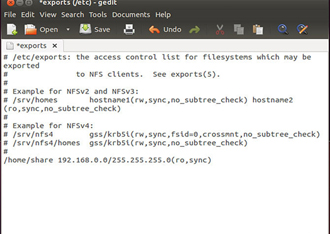


**The Network Filesystem**



Using **NFS** (the **Network Filesystem**) is one of the methods used for sharing data across physical systems. Many system administrators mount remote users' home directories on a **server** in order to give them access to the same files and configuration files across multiple **client** systems. This allows the users to log in to different computers yet still have access to the same files and resources.

**NFS on the Server**

We will now look in detail at how to use NFS on the server machine.

On the server machine, NFS daemons (built-in networking and service processes in Linux) and other system servers are typically started with the following command: sudo service nfs start

The text file /etc/exports contains the directories and permissions that a host is willing to share with other systems over NFS. An entry in this file may look like the following:

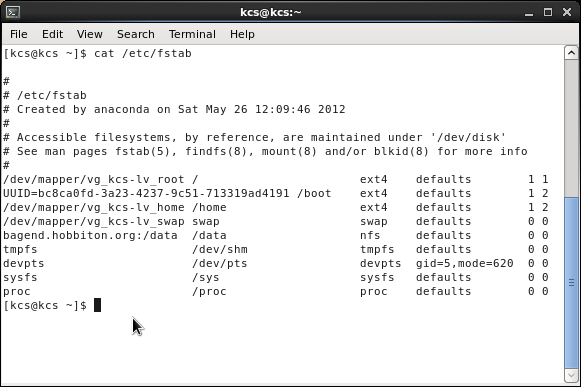
/projects \*.example.com(rw)

This entry allows the directory /projects to be mounted using NFS with read and write (rw) permissions and shared with other hosts in the example.com domain. As we will detail in the next chapter, every file in Linux has 3 possible permissions: **read** (r), **write** (w) and **execute** (x).

After modifying the /etc/exports file, you can use the exportfs -av command to notify Linux about the directories you are allowing to be remotely mounted using NFS (restarting NFS with sudo service nfs restart will also work, but is heavier as it halts NFS for a short while before starting it up again).

Click the image to view an enlarged version.

**NFS on the Client**

On the client machine, if it is desired to have the remote filesystem mounted automatically upon system boot, the /etc/fstab file is modified to accomplish this. For example, an entry in the client's /etc/fstab file might look like the following:

servername:/projects /mnt/nfs/projects nfs defaults 0 0

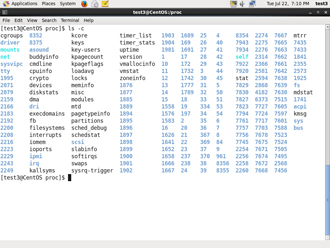
You can also mount the remote filesystem without a reboot or as a one-time mount by directly using the mount command:

$ mount servername:/projects /mnt/nfs/projects

Remember, if /etc/fstab is not modified, this remote mount will not be present the next time the system is restarted.

Click the image to view an enlarged version.

**proc Filesystem**

Certain filesystems like the one mounted at /proc are called **pseudo filesystems** because they have no permanent presence anywhere on disk.

The /proc filesystem contains virtual files (files that exist only in memory) that permit viewing constantly varying kernel data. This filesystem contains files and directories that mimic kernel structures and configuration information. It doesn't contain *real* files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc). Some important files in /proc are:

/proc/cpuinfo

/proc/interrupts

/proc/meminfo

/proc/mounts

/proc/partitions

/proc/version

/proc has subdirectories as well, including:

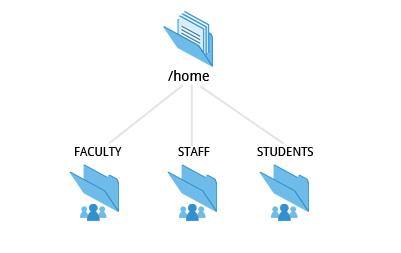
/proc/<Process-ID-#>

/proc/sys

The first example shows there is a directory for every **process** running on the system which contains vital information about it. The second example shows a virtual directory that contains a lot of information about the entire system, in particular its hardware and configuration. The /proc filesystem is very useful because the information it reports is gathered only as needed

and never needs storage on disk.

**Overview of Home Directories**



Now that you know about the basics of filesystems, let's learn about the filesystem architecture and directory structure in Linux.

Each user has a **home directory**, usually placed under /home. The /root (slash-root) directory on modern Linux systems is no more than the root user's home directory.

The /home directory is often mounted as a separate filesystem on its own partition, or even exported (shared) remotely on a network through NFS.

Sometimes you may group users based on their department or function. You can then create subdirectories under the /home directory for each of these groups. For example, a school may organize /home with something like the following:

/home/faculty/

/home/staff/

/home/students/

**The /bin and /sbin Directories**

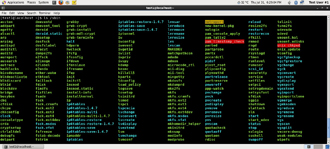
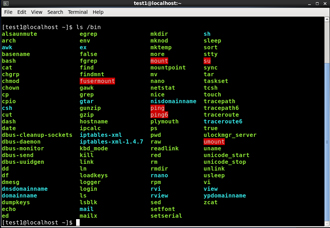
The /bin directory contains executable binaries, essential commands used in single-user mode, and essential commands required by all system users, such as:

|  |  |
| --- | --- |
| **Command** | **Usage** |
| ps | Produces a list of processes along with status information for the system. |
| ls | Produces a listing of the contents of a directory. |
| cp | Used to copy files. |

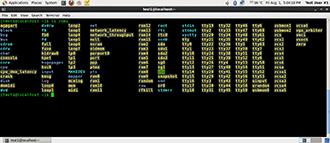
To view a list of programs in the /bin directory, type: ls /bin

Commands that are not essential for the system in single-user mode are placed in the /usr/bin directory, while the /sbin directory is used for essential binaries related to system administration, such as **ifconfig** and **shutdown.** There is also a /usr/sbin directory for less essential system administration programs.

Sometimes /usr is a separate filesystem that may not be available/mounted in single-user mode. This was why essential commands were separated from non-essential commands. However, in some of the most modern Linux systems this distinction is considered obsolete, and /usr/bin and /bin are actually just linked together as are /usr/sbin and /sbin



**The /dev Directory**

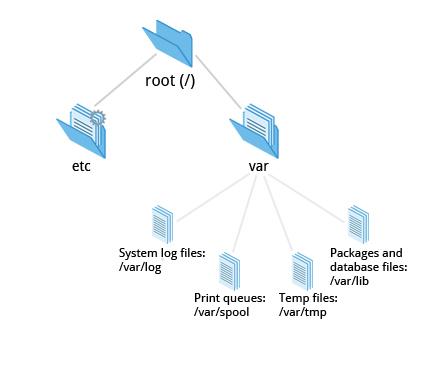


The /dev directory contains **device** **nodes**, a type of pseudo-file used by most hardware and software devices, except for network devices. This directory is:

* Empty on the disk partition when it is not mounted
* Contains entries which are created by the **udev** system, which creates and manages device nodes on Linux, creating them dynamically when devices are found. The /dev directory contains items such as:
  + - * /dev/sda1 (first partition on the first hard disk)
      * /dev/lp1 (second printer)
      * /dev/dvd1 (first DVD drive)

Click the image to view an enlarged version.

**The /var and /etc Directories**



The /var directory contains files that are expected to change in size and content as the system is running (**var** stands for **variable**) such as the entries in the following directories:

* System log files: /var/log
* Packages and database files: /var/lib
* Print queues: /var/spool
* Temp files: /var/tmp

The /var directory may be put in its own filesystem so that growth of the files can be accommodated and the file sizes do not fatally affect the system. Network services directories such as /var/ftp (the FTP service) and /var/www (the HTTP web service) are also found under /var.

The /etc directory is the home for system configuration files. It contains no binary programs, although there are some executable scripts. For example, the file resolv.conf tells the system where to go on the network to obtain host name to IP address mappings (DNS). Files like passwd,shadow and group for managing user accounts are found in the /etc directory. System run level scripts are found in subdirectories of /etc. For example, /etc/rc2.d contains links to scripts for entering and leaving run level 2. The rc directory historically stood for *Run Commands*. Some distros extend the contents of /etc. For example, **Red Hat** adds the sysconfig subdirectory that contains more configuration files.

**The /boot Directory**

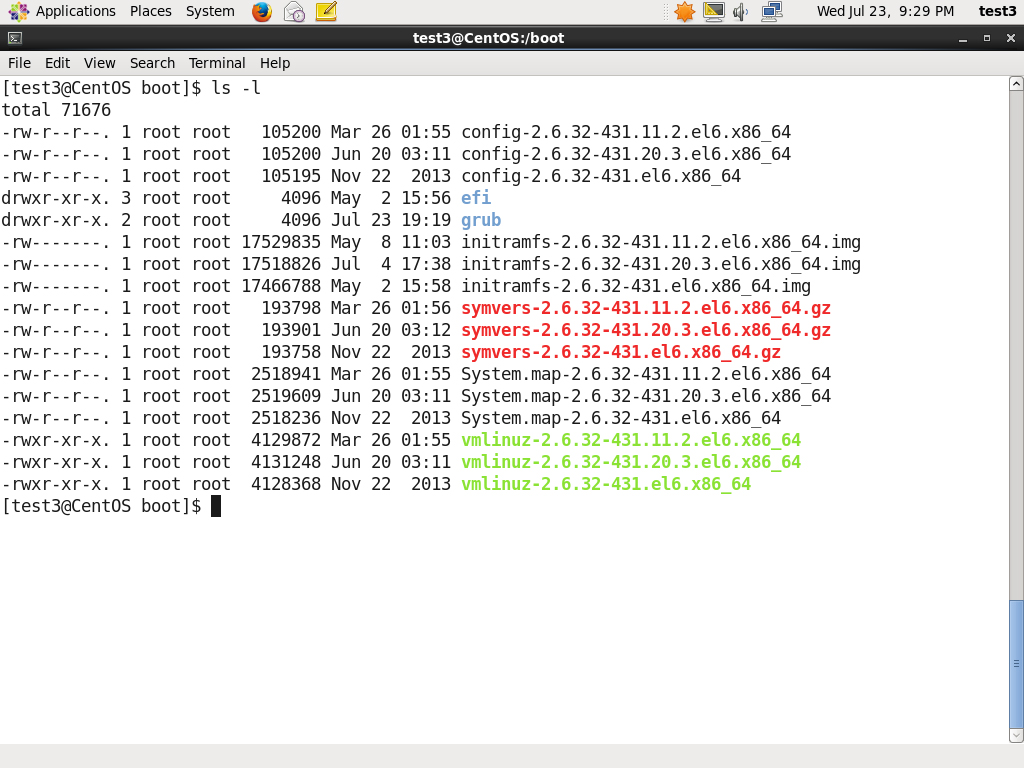
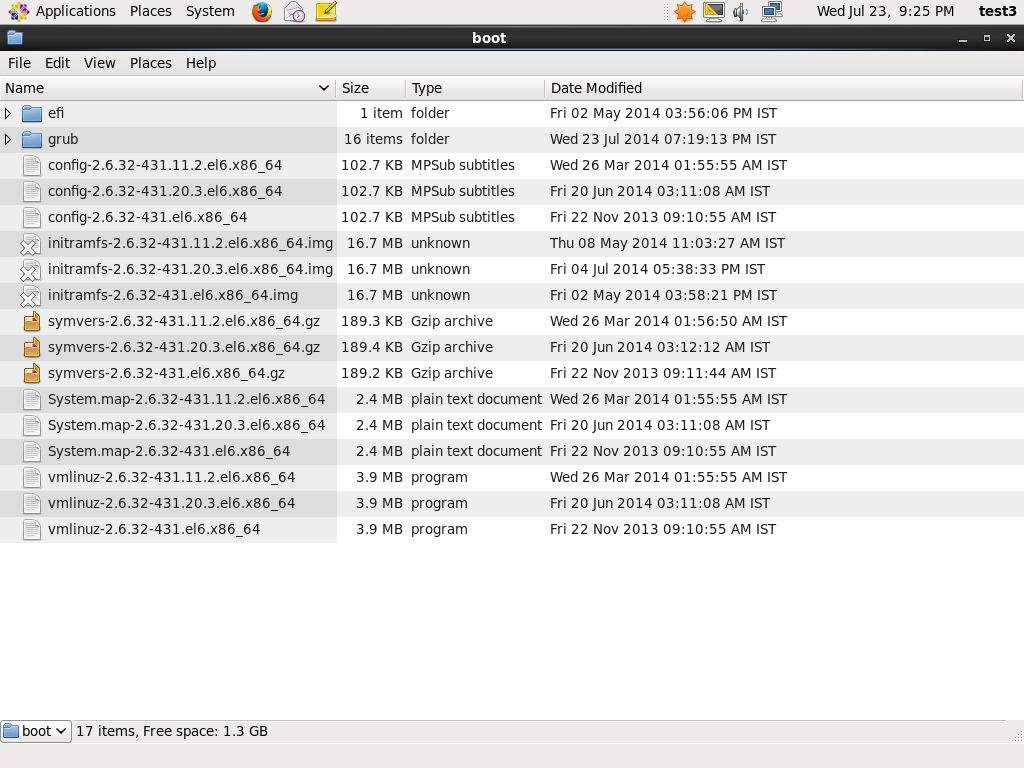
The /boot directory contains the few essential files needed to boot the system. For every alternative kernel installed on the system there are four files:

1. vmlinuz: the compressed Linux kernel, required for booting
2. initramfs: the initial ram filesystem, required for booting, sometimes called initrd, not initramfs
3. config: the kernel configuration file, only used for debugging and bookkeeping
4. System.map: kernel symbol table, only used for debugging

Each of these files has a kernel version appended to its name.

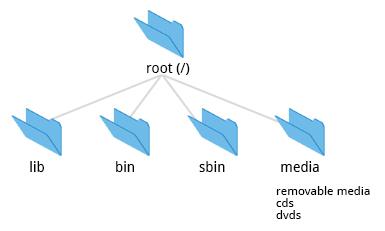
The **Grand Unified Bootloader** (**GRUB**) files (such as /boot/grub/grub.conf or /boot/grub2/grub2.cfg) are also found under the /boot directory.

The images show an example listing of the /boot directory, taken from a **CentOS** system that has three installed kernels. Names would vary and things would look somewhat different on a different distribution.



Click the image to view an enlarged version.

**The /lib and /media Directories**



/lib contains libraries (common code shared by applications and needed for them to run) for the essential programs in /bin and /sbin. These library filenames either start with ld or lib, for example, /lib/libncurses.so.5.7.

Most of these are what are known as **dynamically loaded libraries** (also known as **shared libraries** or **Shared Objects** **(SO**)). On some Linux distributions there exists a /lib64 directory containing 64-bit libraries, while /lib contains 32-bit versions.

Kernel **modules** (kernel code, often device drivers, that can be loaded and unloaded without re-starting the system) are located in /lib/modules/<kernel-version-number>.

The /media directory is typically located where removable media, such as CDs, DVDs and USB drives are mounted. Unless configuration prohibits it, Linux automatically mounts the removable media in the /media directory when they are detected.

* [Previous](https://courses.edx.org/courses/LinuxFoundationX/LFS101x/2T2014/courseware/be8401e6c3bb415a92ca42d2132d8cea/f36c31c43ddb4ec5addf193878191baa/1#)
* [Next](https://courses.edx.org/courses/LinuxFoundationX/LFS101x/2T2014/courseware/be8401e6c3bb415a92ca42d2132d8cea/f36c31c43ddb4ec5addf193878191baa/1#)

**Additional Directories Under /:**

The following is a list of additional directories under /and their use:

|  |  |
| --- | --- |
| **Directory name** | **Usage** |
| /opt | Optional application software packages. |
| /sys | Virtual pseudo-filesystem giving information about the system and the hardware. Can be used to alter system parameters and for debugging purposes. |
| /srv | Site-specific data served up by the system. Seldom used. |
| /tmp | Temporary files; on some distributions erased across a reboot and/or may actually be a ramdisk in memory. |
| /usr | Multi-user applications, utilities and data. |

* [Previous](https://courses.edx.org/courses/LinuxFoundationX/LFS101x/2T2014/courseware/be8401e6c3bb415a92ca42d2132d8cea/f36c31c43ddb4ec5addf193878191baa/1#)
* [Next](https://courses.edx.org/courses/LinuxFoundationX/LFS101x/2T2014/courseware/be8401e6c3bb415a92ca42d2132d8cea/f36c31c43ddb4ec5addf193878191baa/1#)

|  |  |
| --- | --- |
| **Directory name** | **Usage** |
| /usr/include | Header files used to compile applications. |
| /usr/lib | Libraries for programs in /usr/bin and /usr/sbin. |
| /usr/lib64 | 64-bit libraries for 64-bit programs in /usr/bin and /usr/sbin. |
| /usr/sbin | Non-essential system binaries, such as system daemons. |
| /usr/share | Shared data used by applications, generally architecture-independent. |
| /usr/src | Source code, usually for the Linux kernel. |
| /usr/X11R6 | **X Window** configuration files; generally obsolete. |
| /usr/local | Data and programs specific to the local machine. Subdirectories include bin, sbin, lib, share, include, etc. |
| /usr/bin | This is the primary directory of executable commands on the system. |

* [Previous](https://courses.edx.org/courses/LinuxFoundationX/LFS101x/2T2014/courseware/be8401e6c3bb415a92ca42d2132d8cea/f36c31c43ddb4ec5addf193878191baa/1#)
* [Next](https://courses.edx.org/courses/LinuxFoundationX/LFS101x/2T2014/courseware/be8401e6c3bb415a92ca42d2132d8cea/f36c31c43ddb4ec5addf193878191baa/1#)

**Comparing Files**

Now that you know about the filesystem and its structure, let’s learn how to manage files and directories.

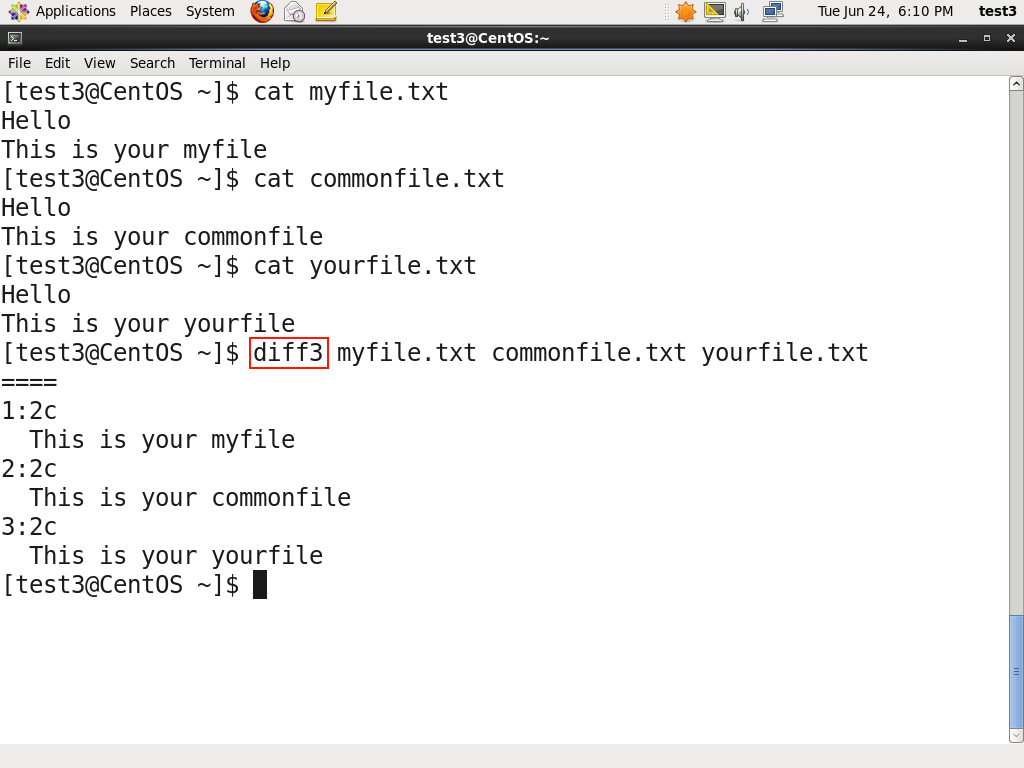
**diff** is used to compare files and directories. This often-used utiility program has many useful options (see man diff) including:

|  |  |
| --- | --- |
| **diff Option** | **Usage** |
| -c | Provides a listing of differences that include 3 lines of **context** before and after the lines differing in content |
| -r | Used to **recursively** compare subdirectories as well as the current directory |
| -i | **Ignore** the case of letters |
| -w | Ignore differences in spaces and tabs (**white space**) |

To compare two files, at the command prompt, type diff <filename1> <filename2>

In this section, you will learn additional methods for comparing files and how to apply **patches** to files.

**Using diff3 and patch**

You can compare three files at once using **diff3**, which uses one file as the reference basis for the other two. For example, suppose you and a co-worker both have made modifications to the same file working at the same time independently. **diff3** can show the differences based on the common file you both started with. The syntax for **diff3** is as follows:

$ diff3 MY-FILE COMMON-FILE YOUR-FILE

Many modifications to source code and configuration files are distributed utilizing **patches**, which are applied, not suprisingly, with the **patch** program. A patch file contains the **deltas** (changes) required to update an older version of a file to the new one. The patch files are actually produced by running **diff** with the correct options, as in:

$ diff -Nur originalfile newfile > patchfile

Distributing just the patch is more concise and efficient than distributing the entire file. For example, if only one line needs to change in a file that contains 1,000 lines, the **patch** file will be just a few lines long.

To apply a patch you can just do either of the two methods below:

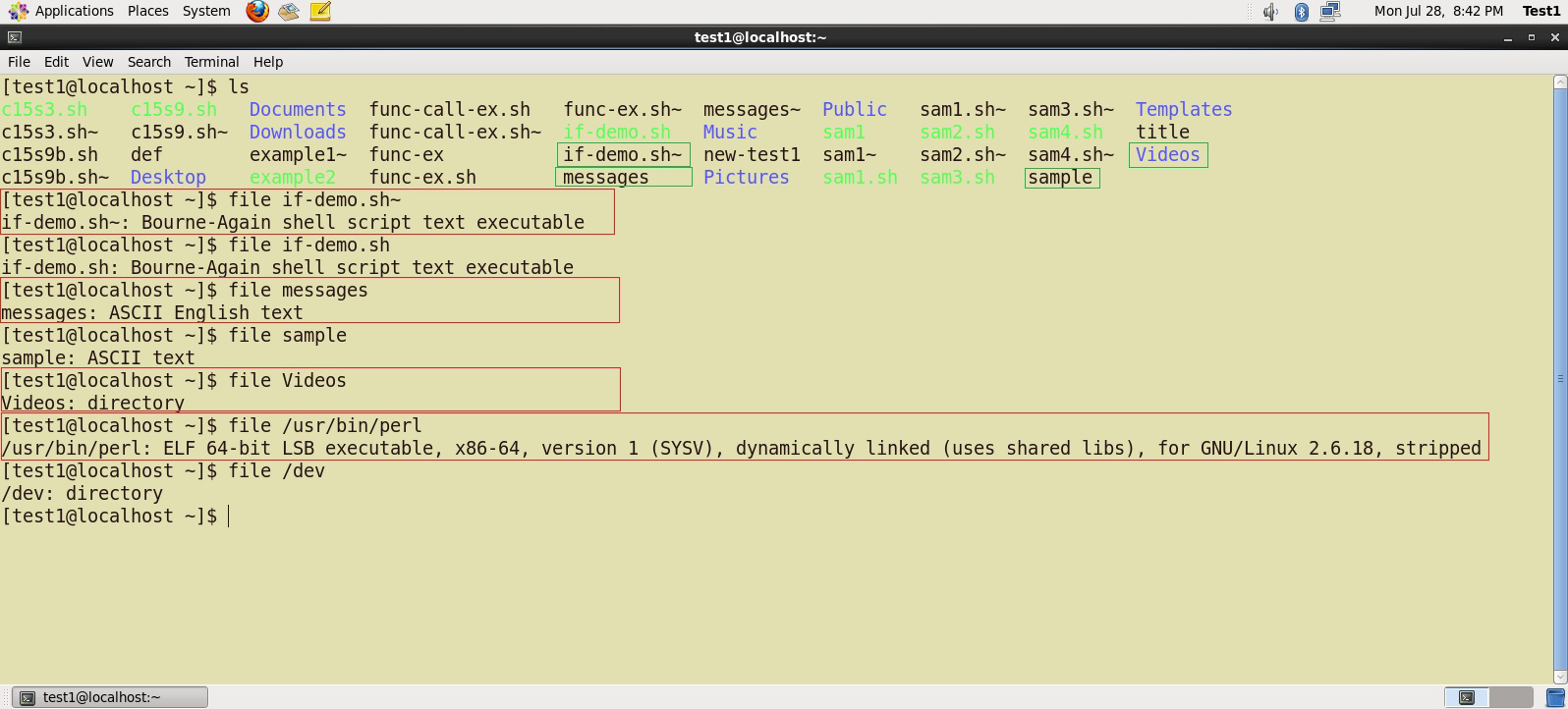
$ patch -p1 < patchfile

$ patch originalfile patchfile

The first usage is more common as it is often used to apply changes to an entire directory tree, rather than just one file as in the second example. To understand the use of the -p1 option and many others, see the **man** page for **patch**.

The graphic shows a patch file produced by **diff**.

**Using the 'file' utility**



In Linux, a file's extension often does not categorize it the way it might in other operating systems. One can not assume that a file named file.txt is a text file and not an executable program. In Linux a file name is generally more meaningful to the user of the system than the system itself; in fact most applications directly examine a file's contents to see what kind of object it is rather than relying on an extension. This is very different from the way **Windows** handles filenames, where a filename ending with .exe, for example, represents an executable binary file.

The real nature of a file can be ascertained by using the **file** utility. For the file names given as arguments, it examines the contents and certain characteristics to determine whether the files are plain text, shared libraries, executable programs, scripts, or something else.