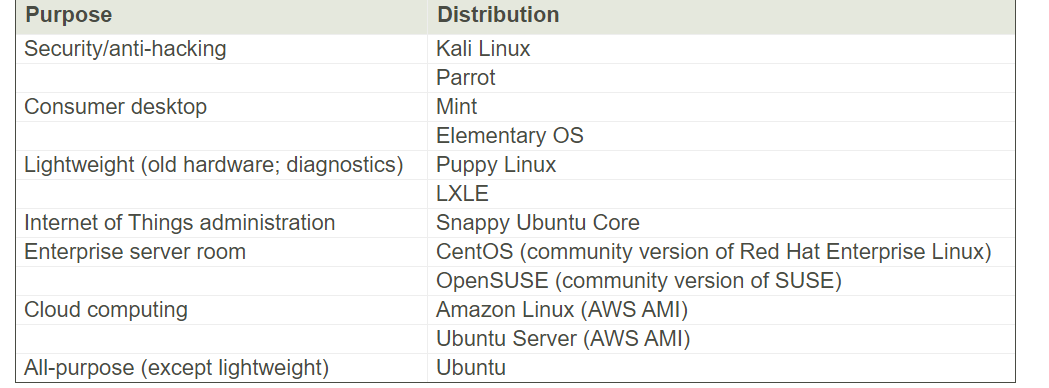
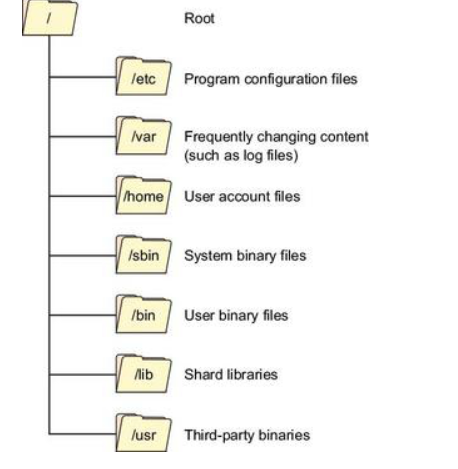
**Linux Distribution types:**

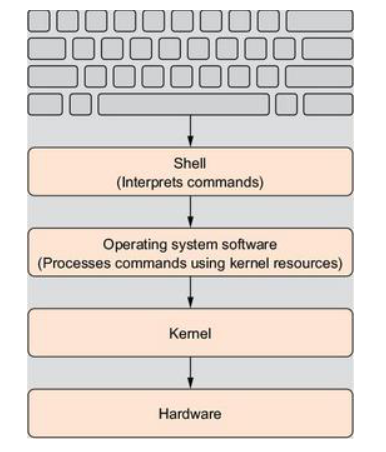


1. In Linux, all data stored as Text files. So it’s interest to know how text files are organized in Linux.
2. Most commonly used Linux file system is ext4
3. Common top level directories defined by Unix FHS [Filesystem Hierarchy Standard]



1. Ls [list]
2. Ls -l [List long]

Bash is probably the most popular UNIX shell. Great! But what's a shell? A *shell* is any user interface that interprets a user's commands, either through a command-line interface (CLI) or a graphical user interface (GUI). You can think of a shell (visualized in the figure) as a software layer meant to execute all appropriately formatted commands using the underlying kernel and hardware system resources. In other words, it's the way you talk to your computer.



**Security Best Practices**

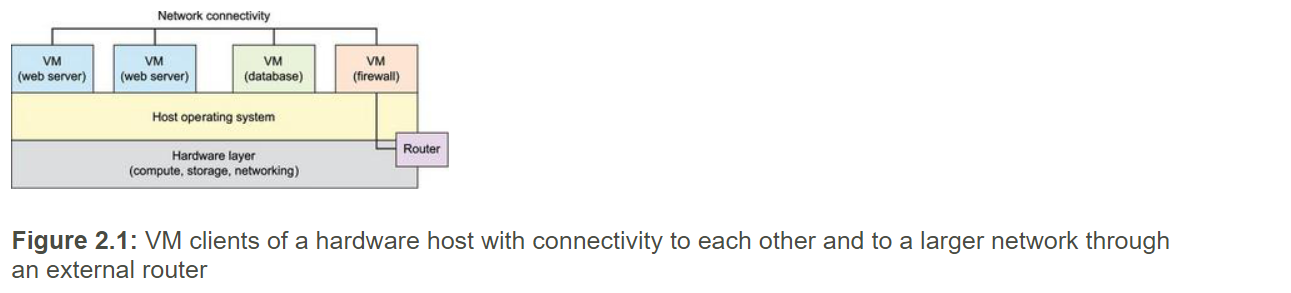
Avoid working on your Linux machine as the root user. Use a regular user account instead, and, when you need to perform administration tasks, use sudo.

Authorized users can invoke sudo to gain administration permissions for individual commands.

five basic tools: ls, pwd, cd, cat, and less.

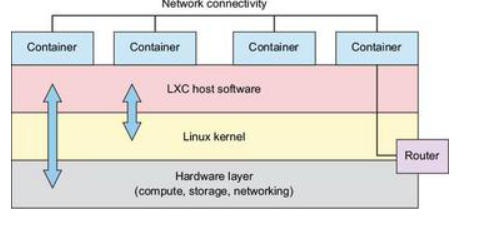
Virtualization is the single most important technology behind almost all recent improvements in the way services and products are delivered. It's made entire industries from cloud computing to self-driving cars not only possible, but compelling. Curious? Here are two virtualization facts you’ll need to know from the start:

* Linux absolutely dominates the virtual space.
* Virtualization makes it easier to learn any technology.



Virtualization achieved in two ways

1. Hypervisors
2. Containers



LXC architecture showing access between the LXC environment and both the Linux kernel and the hardware layer beneath it

**Virtual box:**

===================

VirtualBox provides an environment within which you can launch as many virtual computers as your physical system resources can handle. And it's a particularly useful tool for safely testing and learning new administration skills

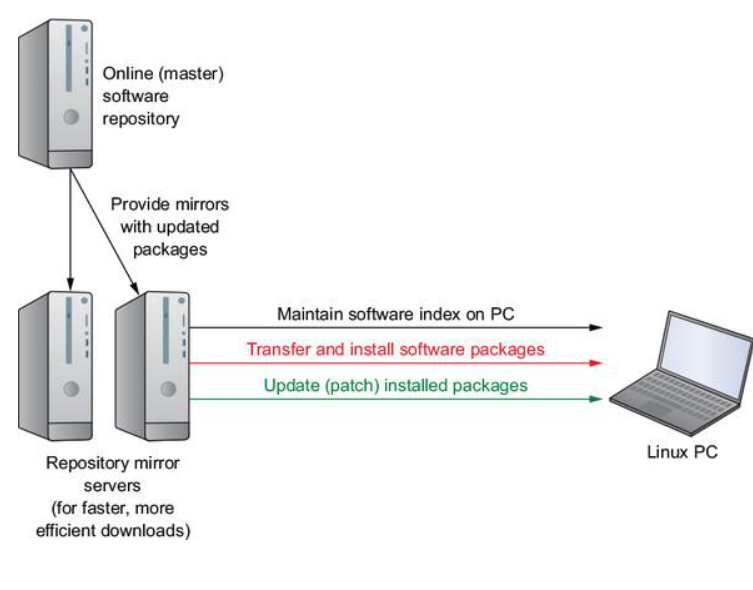
**Package manager:**

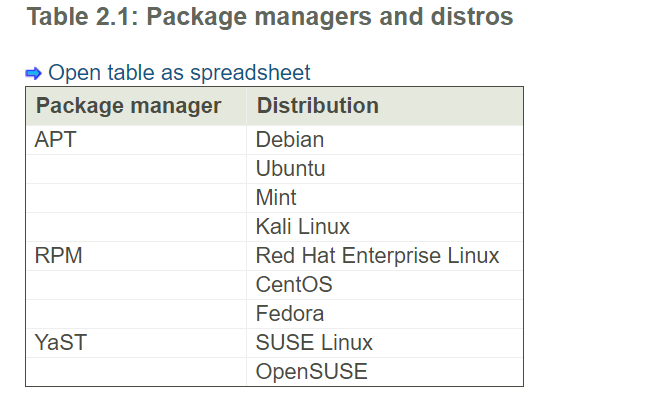
=======================

software package manager called Advanced Package Tool (APT, more commonly known as *apt*). In the Linux world, package managers connect computers to vast online repositories of thousands of software applications, most of them free and open source. The manager, which comes installed with Linux by default, has a number of jobs:

Below are the typical tasks performed by package manager

* Maintains a local index to track repositories and their contents
* Tracks the status of all the software that's installed on your local machine
* Ensures that all available updates are applied to installed software
* Ensures that software dependencies (other software packages or configuration parameters required by the package you're installing) are met for new applications before they're installed
* Handles installing and removing software packages





Almost all Linux distributions supply a shell program from the GNU Project called bash. The name is an acronym for *b*ourne-*a*gain *sh*ell, a reference to the fact that bash is an enhanced replacement for sh, the original Unix shell program written by Steve Bourne.

#### Command History

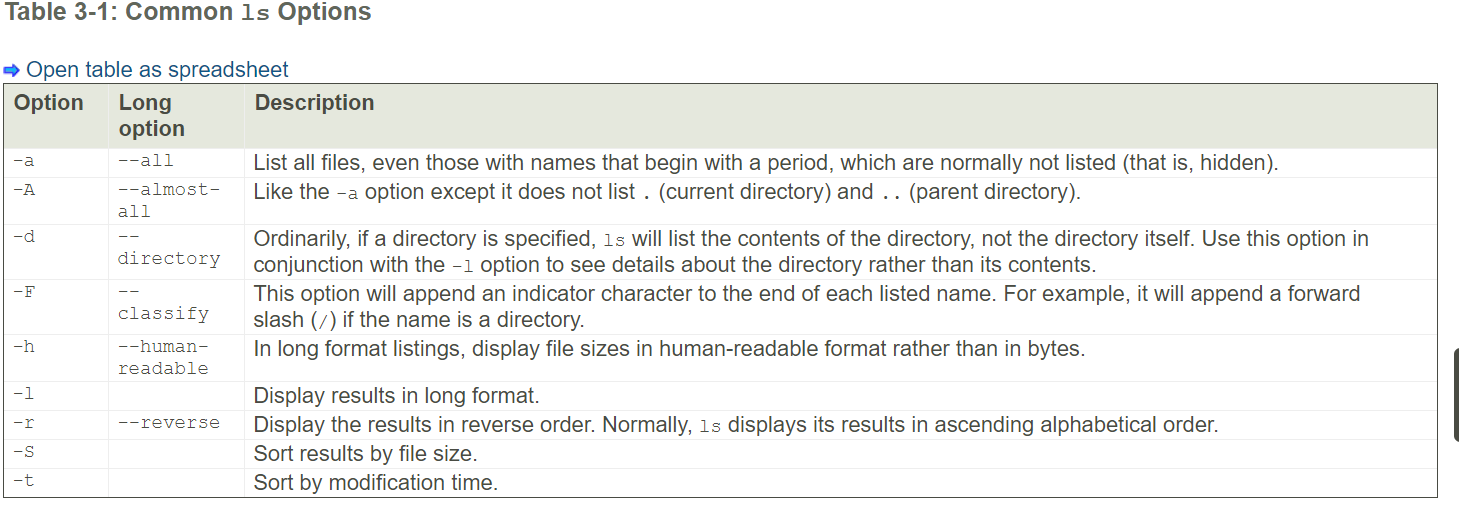
If we press the up arrow, we will see that the previous command entered, kaekfjaeifj, reappears after the prompt. This is called *command history*. Most Linux distributions remember the last 1,000 commands by default. Press the down arrow and the previous command disappears.

Ls

Ls -l

Ls -ltr

Ls ~ /usr



#### Options and Arguments

This brings us to a very important point about how most commands work. Commands are often followed by one or more *options* that modify their behavior and, further, by one or more *arguments*, the items upon which the command acts. So, most commands look kind of like this:

**command** *-options arguments*

### Determining a File's Type with file

**file** *filename*

When invoked, the file command will print a brief description of the file's contents. For example:

[me@linuxbox ~]$ **file picture.jpg**

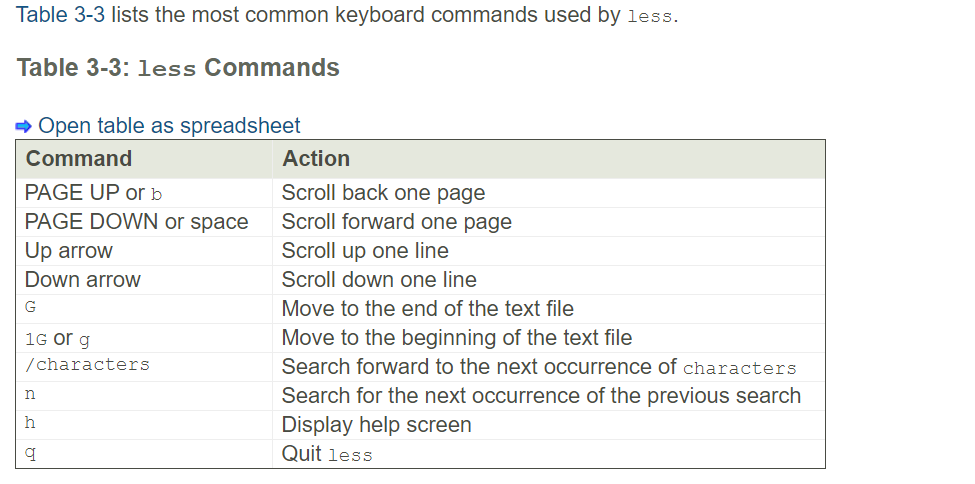
picture.jpg: JPEG image data, JFIF standard 1.01

### Viewing File Contents with less

The less command is a program to view text files. Throughout our Linux system, there are many files that contain human-readable text. The less program provides a convenient way to examine them.

The less command is used like this:

**less** *filename*



* cd into a given directory.
* List the directory contents with ls -l.
* If you see an interesting file, determine its contents with file.
* If it looks like it might be text, try viewing it with less.

**Man <command>**

**Example:** man pwd

Man [manual] command displays the usage information of a command

By default, the BASH shell has a dollar sign (**$**) prompt, but Linux has several other types of shells, each with its own prompt (**%** for the C shell, for example). The root user will have a different prompt, the **#**.

##### Tip:

*Some commands can be complex and take some time to execute. When you mistakenly execute the wrong command, you can interrupt and stop such commands with the interrupt key—CTRL-C*.

You can enter a command on several lines by typing a backslash just before you press ENTER. The backslash “escapes” the ENTER key, effectively continuing the same command line to the next line. In the next example, the **cp** command is entered on three lines:

$ **cp -i \**

**mydata \**

**/home/george/myproject/newdata**

You can also enter several commands on the same line by separating them with a semicolon (**;**). In effect the semicolon operates as an execute operation. Commands will be executed in the sequence they are entered. The following command executes an **ls** command followed by a **date** command.

$ ls ; date

You can also conditionally run several commands on the same line with the **&&** operator (see [Chapter 4](http://viewer.books24x7.com/assetviewer.aspx?bkid=23662&destid=181#181)). A command is executed only if the previous one is true. This feature is useful for running several dependent scripts on the same line. In the next example, the **ls** command runs only if the **date** command is successfully executed.

$ date && ls

Etc/inputrc 🡪 Bash key bindings configuration file

*This file is read automatically by your* /etc/profile *shell configuration file when you log in (see*[*Chapter 5*](http://viewer.books24x7.com/assetviewer.aspx?bkid=23662&destid=237#237)*). Users can customize their editing commands by creating an* .inputrc *file in their home directory (this is a dot file). It may be best to first copy the* /etc/inputrc *file as your* .inputrc *file and then edit it*. /etc/profile *will first check for a local* .inputrc *file before accessing the* /etc/inputrc *file. You can find out more about Readline in the BASH shell reference manual at* [gnu.org/manual/bash](http://gnu.org/manual/bash).

#### IMPORTANT FACTS ABOUT FILENAMES

On Linux systems, files are named in a manner similar to that of other systems such as Windows, but there are some important differences.

* Filenames that begin with a period character are hidden. This only means that ls will not list them unless you say ls -a. When your account was created, several hidden files were placed in your home directory to configure things for your account. In [Chapter 11](http://viewer.books24x7.com/assetviewer.aspx?bkid=146806&destid=337#337) we will take a closer look at some of these files to see how you can customize your environment. In addition, some applications place their configuration and settings files in your home directory as hidden files.
* Filenames and commands in Linux, like Unix, are case sensitive. The filenames *File1* and *file1* refer to different files.
* Though Linux supports long filenames that may contain embedded spaces and punctuation characters, limit the punctuation characters in the names of files you create to period, dash, and underscore. *Most important, do not embed spaces in filenames.* If you want to represent spaces between words in a filename, use underscore characters. You will thank yourself later.
* Linux has no concept of a "file extension" like some other operating systems. You may name files any way you like. The contents or purpose of a file is determined by other means. Although Unix-like operating systems don't use file extensions to determine the contents/purpose of files, many application programs do.
* **ls** List directory contents
* **file** Determine file type
* **less** View file contents

**ls -lt --reverse**

* bash The GNU Bourne Again Shell (bash) is based on the earlier Bourne shell for Unix but extends it in several ways. In Linux, bash is the most common default shell for user accounts, and it's the one emphasized in this book and on the Linux Professional Institute (LPI) exam.
* bsh The Bourne shell upon which bash is based also goes by the name bsh. It's not often used in Linux, although the bsh command is usually a symbolic link to bash.
* tcsh This shell is based on the earlier C shell (csh). It's a fairly popular shell in some circles, but no major Linux distributions make it the default shell. Although it's similar to bash in many respects, some operational details differ. For instance, you don't assign environment variables in the same way in tcsh as in bash.
* csh The original C shell isn't much used on Linux, but if a user is familiar with csh, tcsh makes a good substitute.
* ksh The Korn shell (ksh) was designed to take the best features of the Bourne shell and the C shell and extend them further. It has a small but dedicated following among Linux users.
* zsh The Z shell (zsh) takes shell evolution further than the Korn Shell, incorporating features from earlier shells and adding still more.