**Chapter 1: The Linux Foundation**

**About the Linux Foundation**

Since its inception in 1991, Linux has grown to become a major force in computing - powering everything from the New York Stock Exchange, to mobile phones, supercomputers, and consumer devices.

The **Linux Foundation** is a nonprofit organization that sponsors the work of Linux creator Linus Torvalds. It was founded in 2000 and its mission is to promote, protect, and advance Linux. The Linux Foundation is supported by leading technology companies and thousands of individual members from around the world and marshalls the resources of its members and the open source development community to ensure that Linux remains free and technically advanced.

The Linux Foundation is active on many fronts. In addition to its basic missions of protecting, promoting and advancing Linux, the Foundation:

* + Produces technical events throughout the world (more on this on the following screen).
  + Develops and delivers training programs.
  + Hosts major [collaborative projects](http://collabprojects.linuxfoundation.org/) and industry intiatives ([Click here](https://www.youtube.com/watch?v=ySxHxvaCVN8#t=79) to view a video).
  + Manages [kernel.org](http://www.kernel.org/) where the official versions of the Linux kernel are released.

Runs the popular website [linux.com](http://www.linux.com/).

**Linux Foundation Events**

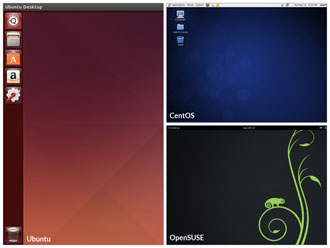
The Linux Foundation hosts conferences and other events throughout the world which bring community members together in person. These events:

* Provide an open forum for development of the next kernel (the actual operating system) release.
* Bring together developers and system administrators to solve problems in a real-time environment.
* Host workgroups and community groups for active discussions.
* Connect end users, system administrators, and kernel developers in order to grow Linux use in the enterprise.
* Encourage collaboration among the entire community.

Provide an atmosphere that is unmatched in its ability to further the platform.

<http://events.linuxfoundation.org>

**Course Software Requirements**

[](https://courses.edx.org/c4x/LinuxFoundationX/LFS101x/asset/LFS01_ch01_screen_14.jpg)

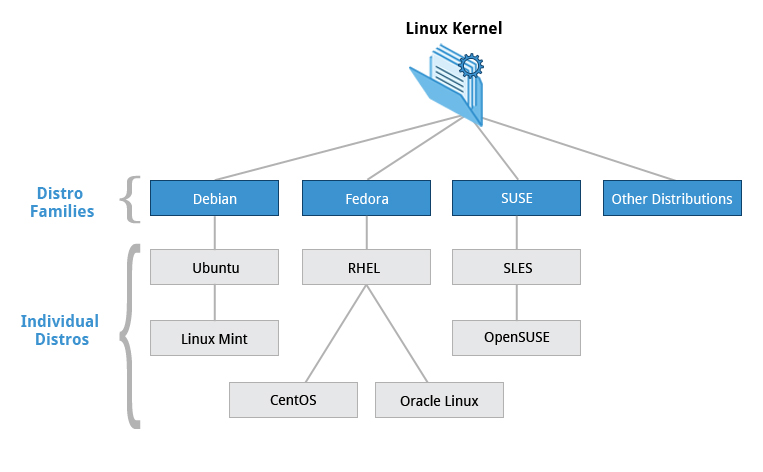
In order to fully benefit from this course, you'll need to have at least one **distribution** of Linux installed (if you're not already familiar with the term "distribution" as it relates to Linux you soon will be!) In the next screen, you will learn some more details about the many available Linux distributions and distribution families. Because there are literally hundreds of distributions, we couldn't possibly cover them all in this course. Instead we have decided to focus on the three major distribution families, and we've chosen one specific distribution from within each family to use for all illustrations, examples, and exercises. This is not meant to suggest that we endorse these specific distributions; they were simply chosen because they are fairly widely used and are broadly representative of their respective family.

The families and representative distributions we are using are:

* **Debian Family Systems (such as Ubuntu)**
* **SUSE Family Systems (such as OpenSUSE)**
* **Fedora Family Systems (such as CentOS)**

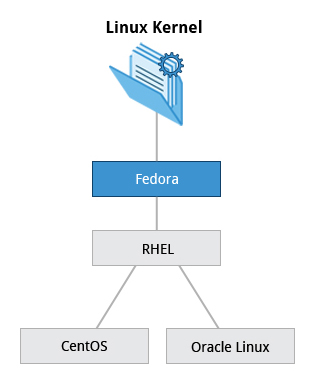
Click the image to view an enlarged version.

**Focus on Three Major Linux Distribution Families**



In the next chapter you will learn about the components that make up a Linux distribution. For now, what you need to know is that this course focuses on the three major Linux distribution families that currently exist. However, as long as there are talented contributors, the families of distributions and the distributions within these families will continue to change and grow. People see a need and develop special configurations and utilities to respond to that need. Sometimes that effort creates a whole new distribution of Linux. Sometimes that effort will leverage an existing distribution to expand the members of an existing family. There is an active discussion thread on Linux distributions on the [linux.com](http://www.linux.com/learn/answers/popular/13-distributions) website.

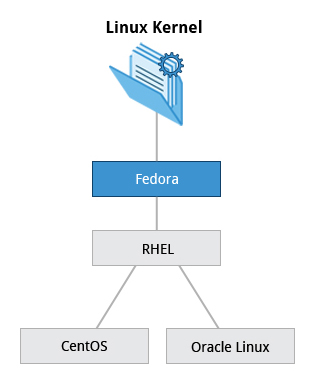
**Fedora Family**



**Fedora** is the community distribution that forms the basis of **Red Hat Enterprise Linux (RHEL)**, **CentOS**, **Scientific Linux**, and **Oracle Linux**. **Fedora** contains significantly more software than Red Hat’s enterprise version. One reason for this is that a diverse community is involved in building **Fedora**; it is not just a company. In this course, **CentOS** is used for activities, demos, and labs because it is available at no cost to the end user and has a much longer release cycle than **Fedora** (which typically releases a new version every six months or so).

For this reason, we have standardized the **Fedora** part of this course material on **CentOS** 6.5. Once installed, **CentOS** is also virtually identical to **Red Hat Enterprise Linux**, which is the most popular Linux distribution in enterprise environments.

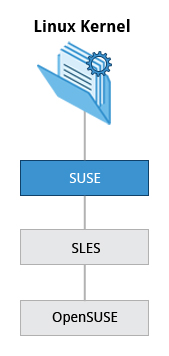
**Key Facts About the Fedora Family**



* + The **Fedora** family is upstream for **CentOS**, **RHEL**, and **Oracle Linux**.
  + The Linux kernel 2.6.32 is used in **RHEL/CentOS 6.x**
  + It supports hardware platforms such as x86, x86-64, Itanium, PowerPC, and IBM System z.
  + It uses the **RPM**-based **yum** package manager (we cover in more detail later) to install, update, and remove packages in the system.

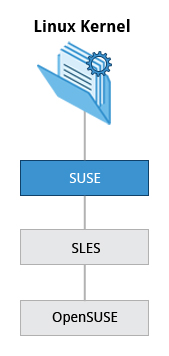
**RHEL** is widely used by enterprises which host their own systems.

**SUSE Family**



The relationship between **SUSE,** **SUSE Linux Enterprise Server (SLES),** and **OpenSUSE**is similar to the one described between **Fedora, Red Hat Enterprise Linux,** and**CentOS**. In this case, we decided to use **OpenSUSE** 12.3 as the reference distribution for the **SUSE** family as it is available to end users at no cost. The two products are extremely similar, and material that covers **OpenSUSE** can typically be applied to **SLES** with no problem.

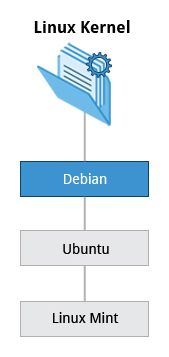
**Key Facts About the SUSE Family**



* + 1. **SUSE Linux Enterprise Server (SLES)** is upstream for **openSUSE**.
    2. The Linux kernel 3.11 is used in **OpenSUSE 12.3**.
    3. It uses the **RPM**-based **zypper** package manager (we cover in more detail later) to install, update, and remove packages in the system.
    4. It includes the **YaST** (Yet another System Tool) application for system administration purposes.

**SUSE** is widely used in the retail sector.

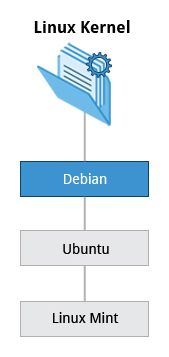
**Debian Family**



The **Debian** distribution is upstream for several other distributions including **Ubuntu**, and **Ubuntu** is upstream for **Linux Mint** and others. It is commonly used on both servers and desktop computers. **Debian** is a pure open source project and focuses on one key aspect, that is, stability. It also provides the largest and most complete software repository to its users.

**Ubuntu** aims at providing a good compromise between long term stability and ease of use. Since **Ubuntu** gets most of its packages from **Debian’s** stable branch, **Ubuntu** also has access to a very large software repository. For those reasons, we decided to use **Ubuntu** 14.04 LTS (Long Term Support) as the reference **Debian family** distribution for this course. **Ubuntu** is a registered trademark of Canonical Ltd. and is used throughout this course with their permission.

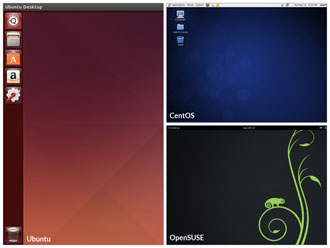
**Key Facts About the Debian Family**



* + The **Debian** family is upstream for **Ubuntu**, and **Ubuntu** is upstream for **Linux Mint** and others.
  + The Linux kernel 3.13 is used in **Ubuntu 14.04**.
  + It uses the **DPKG**-based **apt-get** package manager (we cover in more detail later) to install, update, and remove packages in the system.
  + **Ubuntu** has been widely used for cloud deployments.

While **Ubuntu** is built on top of **Debian**, it uses the **Unity** graphical interface, is **GNOME-**based and differs quite a bit visually from the interface on standard **Debian** as well as other distributions.

**More About the Software Environment**

[](https://courses.edx.org/c4x/LinuxFoundationX/LFS101x/asset/LFS01_ch01_screen_14.jpg)

The material produced by the Linux Foundation is distribution-flexible. This means that technical explanations, labs, and procedures should work on most modern distributions. While choosing a Linux distribution system, you will notice that the technical differences are mainly about package management systems, software versions, and file locations. Once you get a grasp of those differences it becomes relatively painless to switch from one Linux distribution to another.

The desktop environment used for this course is **GNOME**. As you'll see in Chapter 4, there are different environments, but we selected **GNOME** as it is the most popular.

Click the image to view an enlarged version.

**Summary**



You have completed this chapter. Let’s summarize the key concepts covered:

* The Linux Foundation is a nonprofit consortium dedicated to fostering the growth of Linux.
* The Linux Foundation training is for the community and by the community. Linux training is distribution-flexible, technically advanced, and created with the leaders of the Linux development community.

There are three major distribution families within Linux: **Fedora,** **SUSE** and **Debian**. In this course we will work with representative members of all of these families throughout.

**Chapter 2: Linux Philosophy and Concepts**

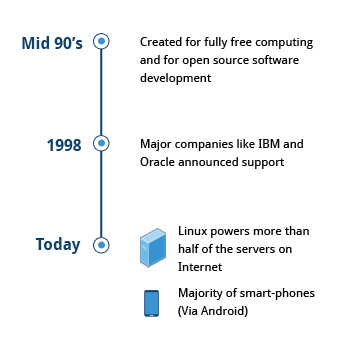
**Linux History**



Linus Torvalds was a student in Helsinki, Finland, in 1991 when he started a project: writing his own operating system**kernel**. He also collected together and/or developed the other essential ingredients required to construct an entire **operating system** with his kernel at the center. This soon became known as the **Linux** kernel.

In 1992, Linux was re-licensed using the **General Public License (GPL)** by **GNU** (a project of the Free Software Foundation (FSF) which promotes freely available software) which made it possible to build a worldwide community of developers. By combining the kernel with other system components from the GNU project, numerous other developers created complete systems called **Linux Distributions** in the mid-90’s.

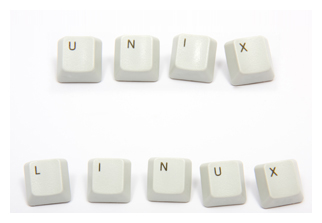
**More About Linux History**



The Linux distributions created in the mid-90’s provided the basis for fully free computing and became a driving force in the open source software movement. In 1998, major companies like **IBM**and **Oracle** announced support for the Linux platform and began major development efforts as well.

Today, Linux powers more than half of the servers on the Internet, the majority of smart-phones (via the **Android** system which is built on top of Linux), and nearly all of the world’s most powerful supercomputers.

**Linux Philosophy**



Linux borrows heavily from the **UNIX** operating system because it was written to be a free and open source version of **UNIX**. Files are stored in a hierarchical filesystem, with the top node of the system being root or simply "/". Whenever possible, Linux makes its components available via files or objects that look like files. Processes, devices, and network sockets are all represented by file-like objects, and can often be worked with using the same utilities used for regular files.

Linux is a fully **multitasking** (a method where multiple tasks are performed during the same period of time), **multiuser** operating system, with built-in networking and service processes known as **daemons** in the **UNIX** world.

That's correct!

Linux borrows heavily from the **UNIX** operating system because it was written to be a free and open source version of **UNIX**. Files are stored in a hierarchical filesystem, with the top node of the system being root directory or simply "/". Whenever possible, Linux makes its components available via files. Processes, devices, and network sockets are all files, and can often be worked with using the same utilities used for regular files.

**More About The Linux Community**

The Linux community is a far-reaching ecosystem consisting of developers, system administrators, users and vendors, who use many different forums to connect with one another. Among the most popular are:

* Linux User Groups (both local and online)
* **I**nternet **R**elay **C**hat (**IRC**) software (such as **Pidgin** and **XChat**)
* Online communities and discussion boards
* Newsgroups and mailing lists
* Community events (such as LinuxCon and ApacheCon)

One of the most powerful online user communities is [linux.com](http://www.linux.com/). This site is hosted by the **Linux Foundation** and serves over one million unique visitors every month. It has active sections on:

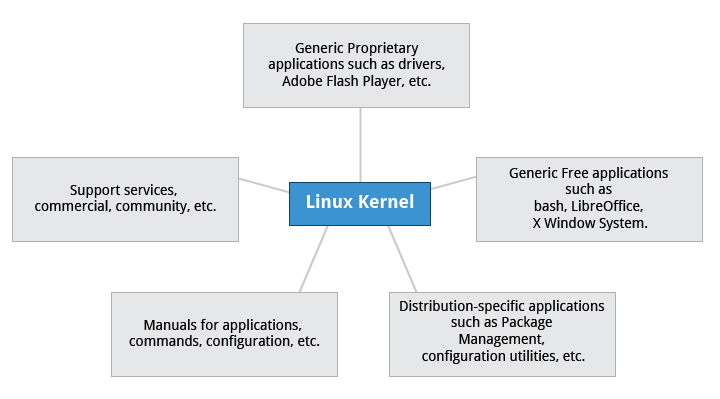
* News
* Community discussion threads
* Free tutorials and user tips

We will refer several times in this course to relevant articles or tutorials on this site.

**EXPLANATION**

The Linux community consists of vendors, developers, and active users. They communicate via Internet Relay Chat (IRC) channels, mailing lists, and newsgroups.

**Linux Distributions**



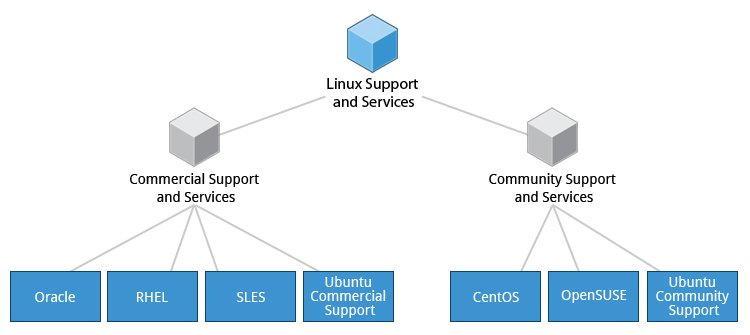
So, what is a Linux distribution and how does it relate to the Linux kernel?

As illustrated above, the Linux kernel is the core of a computer operating system. A full **Linux distribution** consists of the kernel plus a number of other software tools for file-related operations, user management, and software package management. Each of these tools provides a small part of the complete system. Each tool is often its own separate project, with its own developers working to perfect that piece of the system.

As mentioned earlier, the current Linux kernel, along with past Linux kernels (as well as earlier release versions) can be found at the [www.kernel.org](https://www.kernel.org/) web site. The various Linux distributions may be based on different kernel versions.  For example, the very popular **RHEL 6** distribution is based on the 2.6.32 version of the Linux kernel, which is rather old but extremely stable. Other distributions may move more quickly in adopting the latest kernel releases. It is important to note that the kernel is not an all or nothing proposition, for example, **RHEL 6** has incorporated many of the more recent kernel improvements into their version of 2.6.32.

Examples of other essential tools and ingredients provided by distributions include the **C/C++** compiler, the **gdb** debugger, the core system libraries applications need to link with in order to run, the low-level interface for drawing graphics on the screen as well as the higher-level desktop environment, and the system for installing and updating the various components including the kernel itself.

**Services Associated with Distributions**



A vast variety of Linux distributions cater to different audiences and organizations depending on their specific needs. Large commercial organizations tend to favor the commercially supported distributions from **Red Hat, SUSE** and Canonical **(Ubuntu)**.

**CentOS** is a popular free alternative to **Red Hat Enterprise Linux (RHEL)**. **Ubuntu** and **Fedora** are popular in the educational realm. **Scientific Linux** is favored by the scientific research community for its choice of scientific and mathematical software packages. Both **CentOS** and **Scientific Linux** are binary-compatible with **RHEL**; i.e., binary software packages in most cases will install properly across the distributions.

Many commercial distributors, including **Red Hat**, **Ubuntu**, **SUSE**, and **Oracle**, provide long term fee-based support for their distributions, as well as hardware and software certification. All major distributors provide update services for keeping your system primed with the latest security and bug fixes, and performance enhancements, as well as provide online support resources.

**Summary (1 of 2)**

You have completed this chapter. Let’s summarize the key concepts covered.

* + Linux borrows heavily from the **UNIX** operating system, with which its creators were well versed.
  + Linux accesses many features and services through files and file-like objects.
  + Linux is a fully multitasking, multiuser operating system, with built-in networking and service processes known as daemons.

Linux is developed by a loose confederation of developers from all over the world, collaborating over the Internet, with Linus Torvalds at the head. Technical skill and a desire to contribute are the only qualifications for participating.

**Summary (2 of 2)**

* The Linux community is a far reaching ecosystem of developers, vendors, and users that supports and advances the Linux operating system.
* Some of the common terms used in Linux are: Kernel, Distribution, Boot loader, Service, Filesystem, X Window system, desktop environment, and command line.

A full Linux distribution consists of the kernel plus a number of other software tools for file-related operations, user management, and software package management.

By the end of this chapter, you should be able to:

* Identify Linux filesystems.
* Identify the differences between partitions and filesystems.
* Describe the boot process.

Know how to install Linux on a computer.

**Chapter 3: Linux Structure and Installation**

**Linux Filesystems**



Think of a refrigerator that has multiple shelves that can be used for storing various items. These shelves help you organize the grocery items by shape, size, type, etc. The same concept applies to a **filesystem**, which is the embodiment of a method of storing and organizing arbitrary collections of data in a human-usable form.

**Different Types of Filesystems Supported by Linux:**

* Conventional disk filesystems: ext2, ext3, ext4, XFS, Btrfs, JFS, NTFS, etc.
* Flash storage filesystems: ubifs, JFFS2, YAFFS, etc.
* Database filesystems
* Special purpose filesystems: procfs, sysfs, tmpfs, debugfs, etc.

This section will describe the standard filesystem layout shared by most Linux distributions.

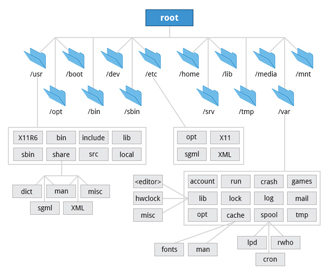
**Partitions and Filesystems**

A **partition** is a logical part of the disk, whereas a **filesystem** is a method of storing/finding files on a hard disk (usually in a partition). By way of analogy, you can think of filesystems as being like family trees that show descendants and their relationships, while the partitions are like different families (each of which has its own tree).

A comparison between filesystems in Windows and Linux is given in the following table:

|  |  |  |
| --- | --- | --- |
|  | **Windows** | **Linux** |
| Partition | Disk1 | /dev/sda1 |
| Filesystem type | NTFS/FAT32 | EXT3/EXT4/XFS... |
| Mounting Parameters | DriveLetter | MountPoint |
| Base Folder where OS is stored | C drive | / |

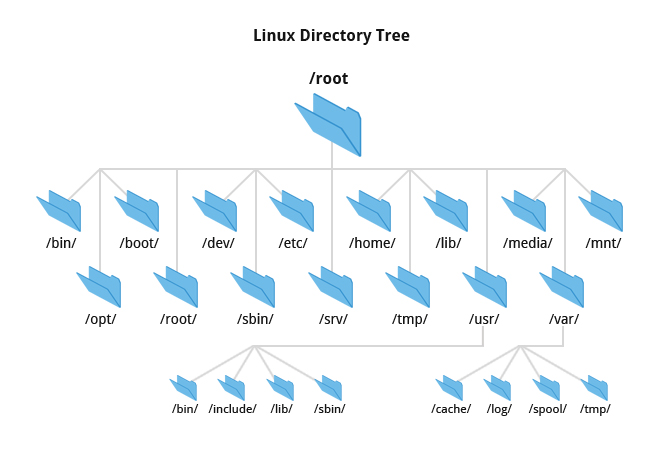
**The Filesystem Hierarchy Standard**

[](https://courses.edx.org/c4x/LinuxFoundationX/LFS101x/asset/chapter03_flowchart_scr05.jpg)

**Linux** systems store their important files according to a standard layout called the **Filesystem Hierarchy Standard**, or **FHS**. You can download a document that provides much greater detail [here](https://courses.edx.org/c4x/LinuxFoundationX/LFS101x/asset/LFS101_Ch3_Sec1_FSH.pdf), or look up the original source documents on the Linux Foundation [website](http://refspecs.linuxfoundation.org/fhs.shtml). This standard ensures that users can move between distributions without having to re-learn how the system is organized.

**Linux** uses the ‘/’ character to separate paths (unlike Windows, which uses ‘\’), and does not have drive letters. New drives are mounted as directories in the single filesystem, often under /media (so, for example, a CD-ROM disc labeled **FEDORA** might end up being found at /media/FEDORA, and a file **README.txt** on that disc would be at /media/FEDORA/README.txt).

**More About the Filesystem Hierarchy Standard**

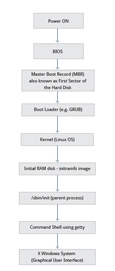


All Linux filesystem names are case-sensitive, so /boot, /Boot, and /BOOT represent three different directories (or folders). Many distributions distinguish between core utilities needed for proper system operation and other programs, and place the latter in directories under /usr (think "**user**"). To get a sense for how the other programs are organized, find the /usr directory in the diagram above and compare the subdirectories with those that exist directly under the system root directory (/).

Try it yourself:

* 1. Click the file manager icon on the left panel.
* By default it explores your home directory.
* 2. To see all directories under "root directory", click Computer on the left Places pane inside this window.
* 3. Double-click the etc directory to open it.
* 4. Double-click avahi and press Ctrl + l to see the current location. This will give you the path
* The current path appears at the top of the window. The path for this is "/etc/avahi".

**The Boot Process**

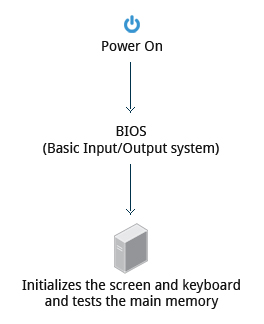
[](https://preview.edge.edx.org/c4x/Linux/LFS101/asset/chapter03_flowchart_scr15_1.jpg)

Have you ever wondered what happens in the background from the time you press the **Power** button until the Linux login prompt appears?

The Linux**boot process** is the procedure for initializing the system. It consists of everything that happens from when the computer power is first switched on until the user interface is fully operational.

Once you start using Linux, you will find that having a good understanding of the steps in the boot process will help you with troubleshooting problems as well as with tailoring the computer's performance to your needs.

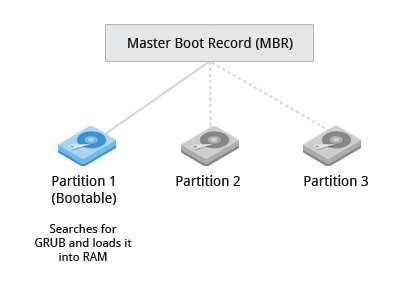
**BIOS - The First Step**



Starting an **x86-**based Linux system involves a number of steps. When the computer is powered on, the **Basic Input/Output System** (**BIOS)** initializes the hardware, including the screen and keyboard, and tests the main memory. This process is also called **POST** (**Power On Self Test**).

The BIOS software is stored on a ROM chip on the motherboard. After this, the remainder of the boot process is completely controlled by the operating system.

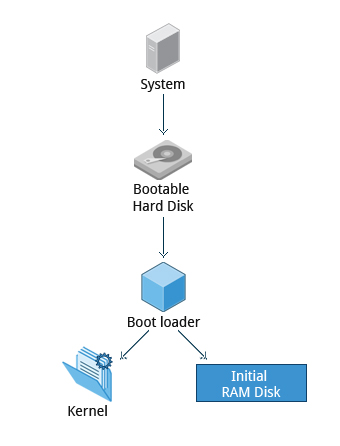
**Master Boot Records (MBR) and Boot Loader**



Once the **POST** is completed, the system control passes from the **BIOS** to the boot loader. The boot loader is usually stored on one of the hard disks in the system, either in the boot sector (for traditional **BIOS/MBR** systems) or the **EFI** partition (for more recent **(Unified)** **Extensible Firmware Interface** or **EFI/UEFI** systems). Up to this stage, the machine does not access any mass storage media. Thereafter, information on the date, time, and the most important peripherals are loaded from a small, battery-powered memory store called the **CMOS values** (after a technology used for the battery - which allows the system to keep track of the date and time even when it is powered off).

A number of boot loaders exist for Linux; the most common ones are **GRUB** (for **GRand Unified Boot loader**) and **ISOLINUX** (for booting from removable media). Most Linux boot loaders can present a user interface for choosing alternative options for booting Linux, and even other operating systems that might be installed. When booting Linux, the boot loader is responsible for loading the kernel image and the initial RAM disk (which contains some critical files and device drivers needed to start the system) into memory.

**Boot Loader in Action**



The boot loader has two distinct stages:

**First Stage**:

For systems using the BIOS/MBR method, the boot loader resides at the first sector of the hard disk also known as the **Master Boot Record** (**MBR**). The size of the MBR is just 512 bytes. In this stage, the boot loader examines the partition table and finds a bootable partition. Once it finds a bootable partition, it then searches for the second stage boot loader e.g, **GRUB**, and loads it into **RAM** (**Random Access Memory).**

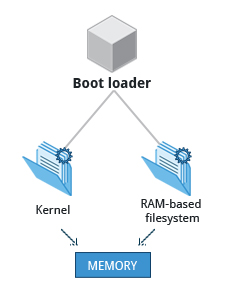
For systems using the **EFI**/**UEFI** method, **UEFI firmware** (software stored in RAM) reads its **Boot Manager** data to determine which **UEFI** application is to be launched and from where (i.e., from which disk and partition the **EFI** partition can be found). The firmware then launches the **UEFI** application, for example, **GRUB**, as defined in the boot entry in the firmware's boot manager. This procedure is more complicated but more versatile than the older MBR methods.

**Second Stage**:

The second stage boot loader resides under /boot. A **splash screen** is displayed which allows us to choose which Operating System (OS) to boot. After choosing the OS, the boot loader loads the kernel of the selected operating system into RAM and passes control to it.

The boot loader loads the selected kernel image (in the case of Linux) and passes control to it. Kernels are almost always compressed, so its first job is to uncompress itself. After this, it will check and analyze the system hardware and initialize any hardware device drivers built into the kernel.

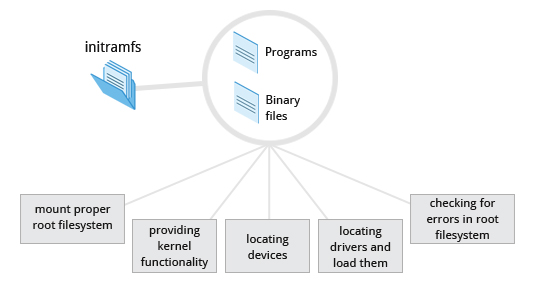
**The Linux Kernel**



The boot loader loads both the kernel and an initial RAM–based file system (**initramfs**) into memory so itcan be used directly by the kernel.

When the kernel is loaded in RAM, it immediately initializes and configures the computer’s memory and also configures all the hardware attached to the system. This includes all processors, I/O subsystems, storage devices, etc. The kernel also loads some necessary user space applications.

**The Initial RAM Disk**

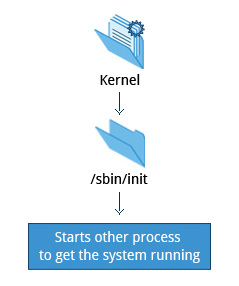


The **initramfs** filesystem image contains programs and binary files that perform all actions needed to mount the proper root filesystem, like providing kernel functionality for the needed filesystem and device drivers for mass storage controllers with a facility called **udev** (for **U**ser **Dev**ice) which is responsible for figuring out which devices are present, locating the **drivers** they need to operate properly, and loading them. After the root filesystem has been found, it is checked for errors and mounted.

The **mount** program instructs the operating system that a filesystem is ready for use, and associates it with a particular point in the overall hierarchy of the filesystem (the **mount point**). If this is successful, the **initramfs**is cleared from RAM and the **init** program on the root filesystem (/sbin/init) is executed.

**init**handles the mounting and pivoting over to the final real root filesystem. If special hardware drivers are needed before the mass storage can be accessed, they must be in the **initramfs** image.

**/sbin/init and Services**

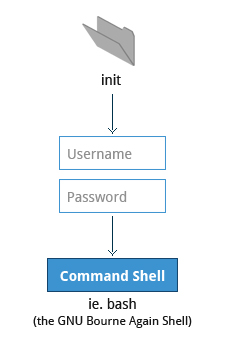


Once the kernel has set up all its hardware and mounted the root filesystem, the kernel runs the /sbin/init program. This then becomes the initial process, which then starts other processes to get the system running. Most other processes on the system trace their origin ultimately to **init**; the exceptions are kernel processes, started by the kernel directly for managing internal operating system details.

Traditionally, this process startup was done using conventions that date back to **System V UNIX**, with the system passing through a sequence of **runlevels**containing collections of scripts that start and stop services. Each runlevel supports a different mode of running the system. Within each runlevel, individual services can be set to run, or to be shut down if running. Newer distributions are moving away from the System V standard, but usually support the System V conventions for compatibility purposes.

Besides starting the system, **init**is responsible for keeping the system running and for shutting it down cleanly. It acts as the "manager of last resort" for all non-kernel processes, cleaning up after them when necessary, and restarts user login services as needed when users log in and out.

**Text-Mode Login**

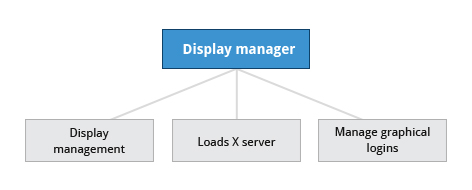


Near the end of the boot process,  **init** starts a number of text-mode login prompts (done by a program called **getty**). These enable you to type your username, followed by your password, and to eventually get a command shell.

Usually, the default command shell is **bash** (the GNU **Bourne Again Shell**), but there are a number of other advanced command shells available. The shell prints a text prompt, indicating it is ready to accept commands; after the user types the command and presses **Enter**, the command is executed, and another prompt is displayed after the command is done.

As you'll learn in the chapter 'Command Line Operations', the terminals which run the command shells can be accessed using the **ALT** key plus a **function** key. Most distributions start six text terminals and one graphics terminal starting with **F1** or **F2**. If the graphical environment is also started, switching to a text console requires pressing **CTRL-ALT +** the appropriate function key (with **F7** or **F1** being the GUI). As you'll see shortly, you may need to run the **startx** command in order to start or restart your graphical desktop after you have been in pure text mode.

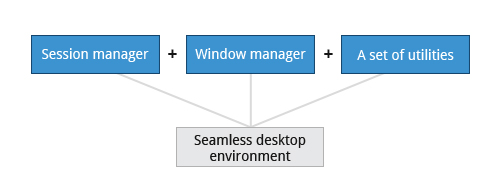
**X Window System**



Generally, in a Linux desktop system, the **X Window System** is loaded as the final step in the boot process.

A service called the **display manager** keeps track of the displays being provided, and loads the **X server** (so-called because it provides graphical services to applications, sometimes called **X clients**). The display manager also handles graphical logins, and starts the appropriate desktop environment after a user logs in.

**More About the X Window System**

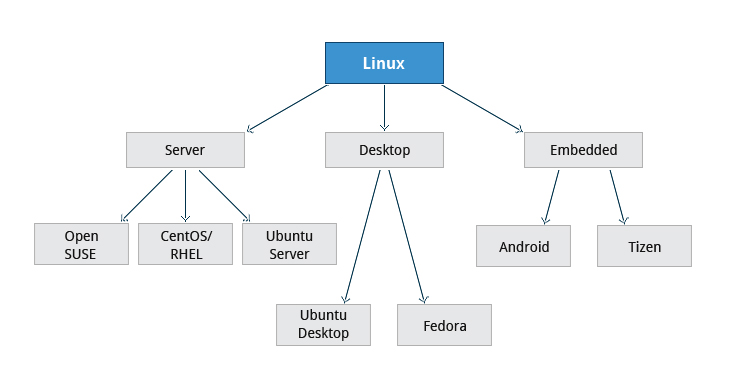


A desktop environment consists of a session manager, which starts and maintains the components of the graphical session, and the window manager, which controls the placement and movement of windows, window title-bars, and controls.

Although these can be mixed, generally a set of utilities, session manager, and window manager are used together as a unit, and together provide a seamless desktop environment.

If the display manager is not started by default in the default runlevel, you can start **X** a different way, after logging on to a text-mode console, by running **startx** from the command line.

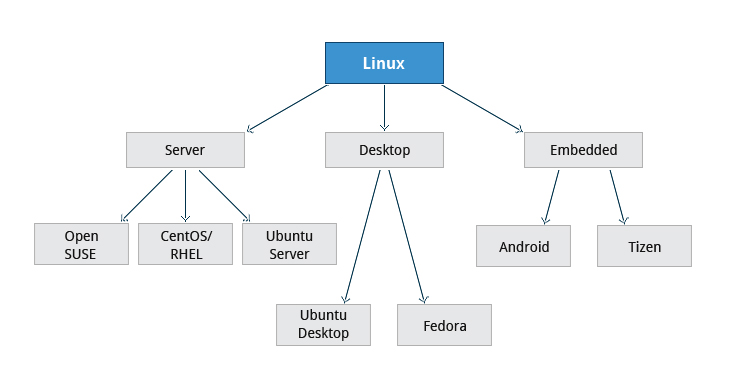
**Choosing a Linux Distribution**



Your family is planning to buy its first car. What are the factors you need to consider while purchasing a car? Your planning depends a lot on your requirements. For instance, your budget, available finances, size of the car, type of engine, after-sales services, etc.

Similarly, determining which distribution to deploy also requires some planning.  The figure shows some but not all choices, as there are other choices for distributions and standard embedded Linux systems are mostly neither **Android** or **Tizen**, but are slimmed down standard distributions.

**Questions to Ask When Choosing a Linux Distribution**

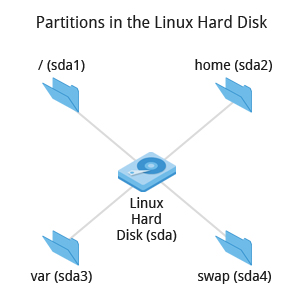


Some questions worth thinking about before deciding on a distribution include:

* What is the main function of the system? (server or desktop)
* What types of packages are important to the organization? For example, web server, word processing, etc.
* How much hard disk space is available? For example, when installing Linux on an embedded device, there will be space limitations.
* How often are packages updated?
* How long is the support cycle for each release? For example, LTS releases have long term support.
* Do you need kernel customization from the vendor?
* What hardware are you running the Linux distribution on? For example, **X86, ARM, PPC**, etc.

Do you need long-term stability or short-term experimental software?

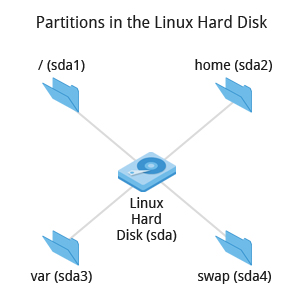
**Linux Installation: Planning**



A **partition** layout needs to be decided at the time of installation because Linux systems handle partitions by mounting them at specific points in the filesystem.  You can always modify the design later, but it always easier to try and get it right to begin with.

Nearly all installers provide a reasonable filesystem layout by default, with either all space dedicated to normal files on one big partition and a smaller **swap** partitition, or with separate partitions for some space-sensitive areas like /home and /var. You may need to override the defaults and do something different if you have special needs, or if you want to use more than one disk.

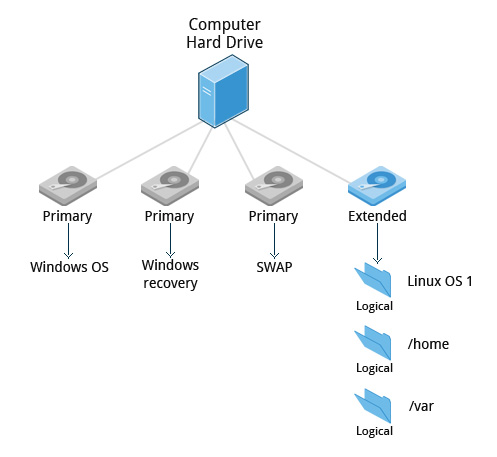
**Linux Installation: Planning**



A **partition** layout needs to be decided at the time of installation because Linux systems handle partitions by mounting them at specific points in the filesystem.  You can always modify the design later, but it always easier to try and get it right to begin with.

Nearly all installers provide a reasonable filesystem layout by default, with either all space dedicated to normal files on one big partition and a smaller **swap** partitition, or with separate partitions for some space-sensitive areas like /home and /var. You may need to override the defaults and do something different if you have special needs, or if you want to use more than one disk.

**More About Planning in Linux Installation**

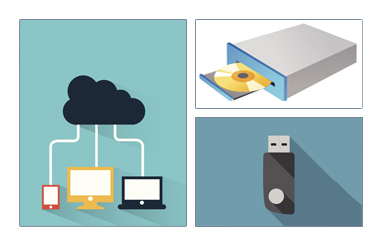


All installations include the bare minimum software for running a Linux distribution.

Most installers also provide options for adding categories of software. Common applications (such as the **Firefox** web browser and **LibreOffice** office suite), developer tools (like the **vi** and **emacs** text editors which we will explore later in this course), and other popular services, (such as the **Apache** web server tools or **MySQL** database) are usually included. In addition, a desktop environment is installed by default.

All installers secure the system being installed as part of the installation. Usually, this consists of setting the password for the superuser (**root**) and setting up an initial user. In some cases (such as **Ubuntu**), only an initial user is set up; direct root login is disabled and root access requires logging in first as a normal user and then using **sudo** as we will describe later.  Some distributions will also install more advanced security frameworks, such as **SELinux** or **AppArmor.**

**Linux Installation: Install Source**



Like other operating systems, Linux distributions are provided on optical media such as CDs or DVDs. USB media is also a popular option. Most Linux distributions support booting a small image and downloading the rest of the system over the network; these small images are usable on media or as network boot images, making it possible to install without any local media at all.

Many installers can do an installation completely automatically, using a configuration file to specify installation options. This file is called a **Kickstart** file for **Fedora**-based systems, an **AutoYAST** profile for **SUSE**-based systems, and a **preseed file** for the **Debian**-based systems.

Each distribution provides its own documentation and tools for creating and managing these files.

**Linux Installation: The Process**



The actual installation process is pretty similar for all distributions.

After booting from the installation media, the installer starts and asks questions about how the system should be set up. (These questions are skipped if an automatic installation file is provided.) Then, the installation is performed.

Finally, the computer reboots into the newly-installed system. On some distributions, additional questions are asked after the system reboots.

Most installers have the option of downloading and installing updates as part of the installation process; this requires Internet access. Otherwise, the system uses its normal update mechanism to retrieve those updates after the installation is done.

**Summary**

You have completed this chapter. Let’s summarize the key concepts covered:

* A **partition** is a logical part of the disk.
* A **filesystem** is a method of storing/finding files on a hard disk.
* By dividing the hard disk into partitions, data can be grouped and separated as needed. When a failure or mistake occurs, only the data in the affected partition will be damaged, while the data on the other partitions will likely survive.
* The boot process has multiple steps, starting with **BIOS**, which triggers the **boot loader** to start up the Linux kernel. From there the **initramfs**filesystem is invoked, which triggers the **init** program to complete the startup process.

Determining the appropriate distribution to deploy requires that you match your specific system needs to the capabilities of the different distributions.