**Learning Report**

Bootloader:

A boot loader is a program responsible for loading the Linux kernel with optional kernel parameters and the Linux initial RAM disk, known as initrd. Linux kernel is the core of the Linux operating system, and it starts the init (short for initialization) process, or an init replacement such as [systemd](https://www.freedesktop.org/wiki/Software/systemd/), immediately after being loaded. The Linux initial RAM disk provides a temporary storage space for loading critical files into memory before the real root file system can be mounted.

On older computers with BIOS (Basic Input/Output System), a boot loader resides in the MBR (Master Boot Record), which occupies the first 512 bytes on a disk, but newer computers with UEFI (Unified Extensible Firmware Interface) store it in a special partition called EFI System Partition.

Kernel:

The Linux® kernel is the main component of a Linux operating system (OS) and is the core interface between a computer’s hardware and its processes. It communicates between the 2, managing resources as efficiently as possible.

The kernel is so named because—like a seed inside a hard shell—it exists within the OS and controls all the major functions of the hardware, whether it’s a phone, laptop, server, or any other kind of computer.

User and SuperUser

Linux puts a lot of power at your fingertips. That’s the best reason to switch to Linux; it’s also the most dangerous thing about the system. Linux controls how much power you can use on the computer based on your Login ID. It keeps a database of all users, and it keeps track of which user owns which files, and which users have permission to view, edit, and execute each file, folder or program. An ordinary user will not be able to do really dangerous things, like editing the user database, or deleting every file on the system.

But right now you are logged in as root. You are not just an ordinary user, you are SuperUser. (SuperUser is a real Unix term, synonymous with root.) There are no restrictions on your power. You have the ability to crash the system and make it otherwise unusable in more ways than you can imagine. As a novice it is very easy to make your system completely unusable with a single erroneous command. Believe me. I speak from experience. The first weekend after I installed Linux, I had to reinstall it FOUR times before I finally got smart and quit destroying it. And I’m a pretty savvy guy around computers, so don’t think you’re immune just because you know your way around a PC.

Because it is so dangerous to be logged in as root, you should never use this account unless you have to. The root account is meant to be used by the System Administrator to perform certain duties which can be destructive and therefore should only be performed by an expert. Some examples are emptying log files, mounting and unmounting file systems (more on this later under [**Getting to CD’s and Floppies**](https://www.control-escape.com/linux/lx-mounting.html) ), installing or removing programs, and creating or deleting user accounts.

If you are using Mandrake Linux, you will have a tool available to perform the most common administration tasks, even when logged in as a regular user. This is called the Mandrake Control Center, which you may find on your desktop or in the Configuration menu. It will ask you for the root password when you start it for security reasons. As a result of this handy tool, you may never need to actually log in as root.

Becoming SuperUser

No phone booth needed. The obvious way is simply to login as root. That may be the best way to do it if you plan on doing a bunch of system maintenance type stuff, but operating as root regularly is a bad idea, as you lose all the security protections that Linux provides. Logging in as the root user is generally discouraged and is in fact prohibited on some Linux systems by default. Fear not, there is a better way.

Try this:

[user]$ sudo ls

Password:\*\*\*\*\*

At the password prompt, type *your password*, not the root password. If it works, you will have just listed the current directory *as superuser*. Congratulations!

If you got an error about not being in the sudoers file, see the section on configuring sudo.

Type this:

[user]$ su

Password:\*\*\*\*\*

[root]#

Bang! Just like that, you are SuperUser! A few cautions: Although you are now SuperUser, this is not a “login” shell, so your environment hasn’t changed. The biggest way this will effect you is that some programs you normally run as root may appear to be missing. That’s because your PATH environment variable, the list of places Linux looks for executables, does not contain /sbin or /usr/sbin. If you try to run a command like shutdown (see below) and it complains, try typing /sbin/shutdown instead. That should do it.

When you are finished with your maintenance tasks you should immediately change back to normal user mode:

[root]# exit

[user]$

Notice that while you are SuperUser, your command prompt looks different. An ordinary user is prompted with the dollar sign ($) while SuperUser gets a pound sign or hashmark (#). This makes it easy to tell which mode you are in. (This is true on most Linux distributions, but your prompts may be different, and they can be customized.)

Pseudo Files:

10

'Pseudo-' means false, pretend. So "pseudo-filesystem" means a filesystem that doesn't have *actual* files – rather, it has virtual entries that the filesystem itself makes up on the spot.

For example, /proc on many OSes is a procfs which dynamically generates directories for every process. Similarly, /sys on Linux generates files and directories to represent hardware layouts. There are FUSE-based pseudo-filesystems for a *lot* of things.

/dev may be a real filesystem (just a subdirectory of /), or a virtual pseudo-filesystem (e.g. devfs), or a middle point such as Linux devtmpfs (which is a full in-memory filesystem but still creates device nodes out of nowhere).

- In the Linux kernel there are 2 different modules:

1. **Static Modules 2. Dynamic Modules.**

**Static modules**are those which are compiled as part of the base kernel and it is available at any time.

**Dynamic Modules** are compiled as modules separately and loaded based on user demand. These are also called as**Loadable Kernel Modules(LKM)**.

Which module(Static or Dynamic) to choose while kernel programming..?

* **Static Modules:** Let’s say I have written a static module(example: ”skgn”) for a device driver. If I compile “skgn” as a static module along with the kernel, I will be adding an extra size to the kernel image permanently. Suppose if I modify “skgn“ device driver then need to re-compile the entire kernel to build it. Every time needs to rebooted for the changes to take effect.
* **LKM:** Let’s say If I want to build my “skgn” module as dynamically(LKM). In case of dynamic no need to rebuild the kernel. It can be compiled separately. It can be loaded into the kernel at run time without having the machine to reboot.

In LKM there are two different ways to load the kernel modules.

1. Insmod : (To load a module into the kernel).
2. modprobe : (To detect any dependent module required) and load it.

* rmmod : To remove a module from kernel.
* dmesg: Command which is used to display the kernel-related m

Primary BootLoader: & Secondary:

Boot loaders are usually configured in one of two ways: either as a primary boot loader or as a secondary boot loader.

Primary boot loaders are where the first stage of the boot loader is installed on the MBR (per the previous description).

Secondary boot loaders are where the **first** stage of the boot loader is installed onto a bootable partition.

**A separate boot loader** must then be installed into the MBR and configured to pass control to the secondary boot loader.

Should "first" in "Secondary boot loaders are where the **first** stage of the boot loader is installed onto a bootable partition" be "second" instead?

Where is the secondary boot loader stored? Not in MBR where the primary boot loader is stored?

What is "a separate boot loader" in the last sentence? Is it the primary or secondary boot loader?

is the primary boot loader the same concept as the initial program loader (IPL, mentioned in the same link)?

Due to the very small amount of data the BIOS can access, most boot loaders load in two stages. In the first stage of the boot, the BIOS loads a part of the boot loader known as the initial program loader, or IPL. The IPL interrogates the partition table and subsequently is able to load data wherever it may exist on the various media. This action is used initially to locate the second stage boot loader, which holds the remainder of the loader. The second stage boot loader is the real meat of the boot loader; many consider it the only real part of the boot loader. This contains the more disk-intensive parts of the loader, such as user interfaces and kernel loaders. These user interfaces can range from a simple command line to the all-singing, all-dancing GUIs.