Basics of Linux Kernel

The Linux Kernel is often misunderstood or not even known about. To help everyone out we can go over some basics of the Kernel and help you to understand things better.  
  
There are many aspects of the Kernel which will help you to understand the concept of the Kernel.  
  
**General**  
  
When a system is booted up there are several things occurring before the Operating System (OS) loads. The boot process is performed in the following six stages:

1. Basic Input/Output System (BIOS)
2. Master Boot Record (MBR)
3. GRand Unified Bootloader (GRUB)
4. Linux Kernel
5. Init
6. Runlevel

**NOTE:** I am using the BIOS as Step 1, but UEFI does exist as well. Of course, I am assuming the use of GRUB, but there are others which can be placed in this slot. No matter the boot loader the function is similar.  
  
Let’s look over these six items in simple terms so we can look further into the Kernel.  
  
**Basic Input/Output System BIOS**  
  
The BIOS is firmware that performs hardware initialization when the system is powered-on. The BIOS will initialize and test system hardware. The BIOS contains enough information to load a driver for the keyboard and sometimes a mouse. When entering the BIOS settings you can use the keyboard and maybe the mouse to save specific settings in the CMOS.  
  
Within the BIOS settings (CMOS) is the boot device order. Control will be passed to each boot device in order of priority. Each device will be checked for a Master Boot Record (MBR). When one is found the BIOS passes control to the devices MBR.  
  
**Master Boot Record (MBR)**  
  
The Master Boot Record (MBR) is technically not required in some cases, but the MBR is a boot sector placed at the beginning of a fixed or removable disk. The MBR contains the data to let the system know about the partitions on a disk. There is also executable code which is a loader (boot loader). Not all partitions contain a boot loader and the if no boot loader is found then an OS will not be loaded. The MBR will exist without the executable code or boot loader. Do not confuse the MBR and the boot loader.  
  
**NOTE:** An MBR is not present on non-partitioned media such as floppies and similar storage devices. You may have noticed that all new systems do not contain a floppy drive.  
  
Imagine partitioning and formatting a hard disk but not installing an OS. When an OS is installed the execution code for a boot loader will then be placed in the MBR. With no OS the system will not boot properly. The MBR exists without the boot loader.  
  
**GRand Unified Bootloader (GRUB)**  
  
The first stage of the boot loader is placed in the MBR which calls the main boot loader which is installed on a boot partition (/boot). GRUB, if configured to do so, will show a menu and allow you to pass control to a specific Kernel. This feature allows a system to have multiple Operating Systems installed on it. The GRUB Configuration file is stored at ‘/boot/grub/grub.conf’.  
  
Keep in mind that there may be multiple menu items for the same OS. Each entry can pass options to the Kernel which can either load the OS or perform diagnostics. Once a menu item is selected or the default is accepted then control is passed to the specified Kernel.  
  
**Kernel**  
  
When the Kernel has control the Initial RAM Disk (‘initrd’) is used as a temporary root file system until the Kernel is booted. Once booted the Kernel then mounts the real root file system. The location of the ROOT file system is passed from GRUB has an option ‘root=’. The Kernel as drivers included to access hard drive partitions as well as some other hardware. The drivers compiled into the Kernel allows it to mount other file systems.  
  
The Kernel will then execute ‘/sbin/init’.  
  
**Init**  
  
Since ‘init’ is the first program loaded by the Kernel it will have the Process ID (PID) of 1.  
  
The ‘init’ program will check the ‘/etc/inittab’ file to determine the Linux runlevel. Once checked, programs are loaded at the specified runlevel.  
  
**Runlevel**  
  
The runlevels are as follows:

* 0 – halt
* 1 – Single user mode
* 2 – Multiuser, without NFS
* 3 – Full multiuser mode
* 4 – unused
* 5 – X11
* 6 – reboot

Programs set to auto-start at boot time can be found in one of the following:  
  
Run level 0 – /etc/rc.d/rc0.d/

* Run level 1 – /etc/rc.d/rc1.d/
* Run level 2 – /etc/rc.d/rc2.d/
* Run level 3 – /etc/rc.d/rc3.d/
* Run level 4 – /etc/rc.d/rc4.d/
* Run level 5 – /etc/rc.d/rc5.d/
* Run level 6 – /etc/rc.d/rc6.d/

To see the default runlevel you can execute the following in a Terminal:  
  
*runlevel*  
  
**Kernel**   
  
From the previous information on the Kernel you should have a basic understanding of it to get the system running.  
  
Once the system is running, the Kernel has control of the software and hardware. No access to hardware is done directly by an application or conflicts would occur. If two applications were accessing the hard drive and one locked it to exclusive rights then no other application could access the hard drive. Applications call functions from the OS Library. The Library is a set of instructions, usually written in C, and manage the necessary abilities of the programs. For instance, a function could request a file from the hard drive.  
  
The Library function will then make System Calls. System Calls are made to the Kernel which in turn provide the information requested. For example, if the function needs a file then the Library will make the System Call to open a file, read the file and close the file when done. The Kernel will perform the file operations and return the data which was read from the file.  
  
The Kernel also tracks all the hardware in the system or that which is connected to it. Even if there isn’t a driver for a device, the device is still listed by the Kernel.  
  
To see a listing of hardware, use the command ‘lshw’ which is ‘list hardware’.  
  
To see specific hardware listings check the following list:

* lspci – PCI device
* lsusb – USB devices
* lsblk – Block devices (devices from which you read blocks of data)
* lscpu – CPU information
* lsdev – other system devices
* lsipc – information on process communication (ability of processes to communicate with one another)
* lslogins – specify a user name and get information about the user’s login
* lsinitramfs – specify the initram file and the contents will be displayed
* lslocks – shows the system locks incurred by the Kernel
* lsof – list of open files (opened by system calls, of course)

If you open a Terminal and execute ‘lshw’ you can see a listing of all the hardware known by the Kernel. To get a better visual you can run the command ‘lshw -html > hardware.html’ and then open the new file with a browser since the output is in an HTML format.  
  
Not all the commands can output to an HTML format. Most of the listed programs will only output to standard text. With some commands you can use the option ‘-v’ to provide more details, or verbose, information. Such as the command ‘lsusb’. You can use the command ‘lsusb -v’ to see more detailed information on the USB hardware. Some of these commands will produce more information if you run it with ROOT privileges.  
  
Play around with the commands and see the details on your system.  
  
**The Initial RamDisk (initrd)**  
  
The ‘initrd’ file is found in the folder ‘/boot’ and has a name starting with ‘initrd.img-’. The remainig portion of the name is the version number. If you perform the command ‘ls /boot/initrd.img-\*’ you will see a listing of all the ‘initrd’ files on the system. Of course, only one is used. Other files listed may be previous versions which have been updated to a newer version. If there is a listing of more than one file then you can determine which one is being used with the command ‘uname -r’. The version listed is that of the Kernel. The results of the ‘uname -r’ command will be the rest of the file name.  
  
For example, if I ran the command ‘uname -r’ and received the response of ‘4.4.0-79-generic’ then the ‘initrd’ file name is ‘initrd.img-4.4.0-79-generic’.  
  
If you wanted to see the contents of the ‘initrd’ file you can run the command ‘lsinitramfs’ with the file name. An example is ‘lsinitramfs /boot/initrd.img-4.4.0-79-generic’. As you may guess, the ‘initrd’ file is compressed so it can be read faster.  
  
Do not modify or delete the ‘initrd’ file or the system may not boot.  
  
The Kernel file is in the same folder as the ‘initrd’ and starts with ‘vmlinuz-’. The end of the file name is the Kernel version number given by the command ‘uname -r’.  
  
For example, from above the result of the command ‘uname-r’ was ‘ 4.4.0-79-generic’. The Kernel file name is ‘vmlinuz- 4.4.0-79-generic’ To see the information about the Kernel file you can perform the command ‘ls -l vmlinuz-\*’. Find the file for your current running version and you can see the file size.  
  
It is imperative that the Kernel is not deleted or modified without knowledge of what you are doing or the system may not boot.  
  
I hope this helps with your understanding of the workings of the Kernel as well as the physical files used for booting your system.