System Boot Sequence

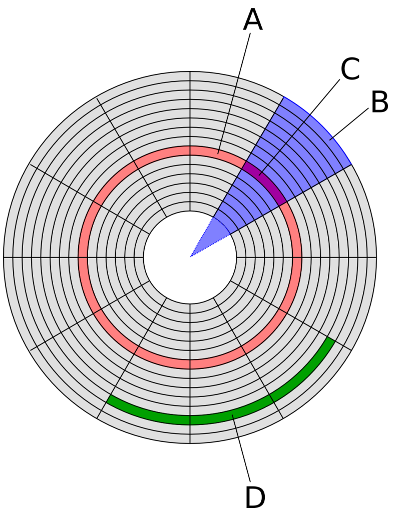
Zilogic Systems

1. Hard Disk

1.1. Terminology

Platter	individual disks that make up a hard disk.
Head	the device that reads and writes information on a platter’s surface.
Tracks	concentric rings on the surface of the platters on which data is stored.
Sectors	small equal sized arcs on tracks, that forms the smallest unit of data that can be read or stored.
Partition	defined storage area on a hard disk that acts as if it were a separate hard disk.

Figure 1. Disk Structure



a. Track

b. Geometrical Sector

c. Sector

d. Cluster of Sectors

1.2. Need for Partitions

1.2.1. Multi Boot Systems

- Multi-boot systems have more than one operating system.
- The operating system to be used can be selected during boot up.
- Partitions are required to prevent one operating system from stepping over another operating system’s data.

1.2.2. Isolating Files

- Operating system and user files can be stored in separate partitions.
- A re-install of the operating system will not affect the user’s files.

1.2.3. Swap Partitions

- Operating systems are capable of moving out data from memory to disk in the event of non-availability of memory while executing a program.
- The data can be moved back from disk backup to memory when required.
- This process is called swapping.
- Separate swap partitions are used to store data that are swapped out by the operating system.

1.3. Partitions

- PCs were originally designed to have a maximum of 4 partitions per disk.
- This was sufficient for older hard disks, that were just few hundred MBs in size.
- This became a serious limitation when the hard disks got larger.
- Logical partitions were invented to overcome this problem.
- One of the primary partitions is converted to an extended partition.
- The extended partition can in turn contain any no. of logical partitions.

1.4. Partition Naming

- GNU/Linux represents disks and partitions using device files.

Table 1. Device files for Disks

Device File	Disk Type
`/dev/hd[a-d]`	Hard Disk Drives
`/dev/sd[a-z]`	SCSI, SATA and USB Drives
`/dev/fd[a-b]`	Floppy drives

- The device files of partitions are named as `disk device file name + partition number`. Ex: `/dev/hda1`, `/dev/sdb2`, …
- The primary partitions are numbered from 1 to 4. The logical partition numbering starts from 5, even if there are less than 4 primary partitions.
- Other tools like the GRUB boot loader, have a different disk and partition naming scheme.

Table 2. Names for Disks in GRUB

Name	Disk Type
`(hd[0-9])`	Hard Disks
`(fd[0-9])`	Floppy Disks

- Examples of partition names are shown below.

Table 3. Names of Partitions

Name	Partition
`(hd0,2)`	Second partition in the first disk.
`(hd1,3)`	Third partition in the second disk.

2. System Boot Sequence

The following components are involved in the system boot sequence

BIOS	firmware, that initalizes the system.
Boot Loader	program that loads the operating system kernel.
Kernel	resource manager of the operating system.
init	parent of all processes in the system.
getty	configures the communication link to the terminal, and attaches itself to the terminal, and spawns login.
login	authenticates the user and spawns the login shell.

3. BIOS

- The BIOS is a program that initializes the system and performs basic tests on the system, called POST (Power On Self Test).
- Any faults detected by the system is reported to the user.
- The BIOS is located in a non-volatile memory chip.
- When the system is powered ON or reset, the processor starts executing the BIOS code.
- After the BIOS performs system initialization, and POST, it gets the first boot device, also stored in non-volatile memory.
- The boot device can be changed by the user during boot up. The boot order can be changed through the BIOS setup program.
- If the first boot device is the hard disk, it loads the first sector of the hard disk to memory, and executes.
- The first sector of the hard disk is called the Master Boot Record (MBR).

4. MBR

- The MBR is the first sector of the hard disk.
- It is not part of any partition.
- In the PC, the MBR has two parts a boot code and the partition table.
- The partition table specifies what primary partitions are present in the system and what are their physical extents.
- The instructions in the boot code, depends on whether the MBR was installed by Windows or GNU/Linux.
- If the MBR was installed by Windows, it is called a DOS MBR.
- If the MBR was installed by GNU/Linux, it is called a GRUB MBR. (Named after GNU/Linux' boot loader)

4.1. DOS MBR

- The partition table contains a flag called the active flag.
- The active flag, specifies which of these partitions contains the operating system.
- The DOS MBR, loads the first sector of the active partition and executes it.

4.2. GRUB MBR

- The GRUB MBR, loads the rest of the GRUB boot loader from disk to memory, and executes it.

5. Boot Loader

- The boot loader is responsible for loading the operating system kernel from disk to memory and executing it.

- The boot loader also provides a menu from which the operating system to be booted can be choosen.
- Just as a command can accept arguments and options, the kernel also accepts arguments.
- The boot loader also permits the user to pass additional arguments to the kernel, that modifies its behaviour.
- The default boot loader of most GNU/Linux systems is GRUB.
- LILO is yet another popular boot loader.
- LILO is file system unaware, meaning that you will have to specify that which **sectors** contains the kernel.
- GRUB is much more powerful, and is filesystem aware, meaning that you can specify location of the kernel as a pathname, rather than sectors. GRUB can traverse the file system, and fetch the kernel.
- GRUB provides three interfaces - menu interface, command interface, menu editing interface

5.1. Command Interface

- Press `c` in the menu interface to enter the command interface.
- The `root` variable is used to specify the partititon that is to be considered as root.
- All subsequent references to files are considered to be located within the root partition.
- To set the first partition in the first hard disk as root, the following command can be used.

```
set root=(hd0,1)
```

- The command to load the kernel to be booted is `linux`.
- The command takes the location of the kernel and kernel boot arguments as parameters.
- The following example specifies the location of the kernel, and does not pass any boot arguments.

```
linux /boot/vmlinuz
```

- One information that is required by the kernel during boot up is the partition in which the root filesystem is located.
- The location of the root filesystem is passed as an argument to the kernel.
- The following example, specifies the location of the kernel, and provides a boot argument to the kernel.

```
linux /boot/vmlinuz root=/dev/sda1
```

- The kernel also requires a bunch of drivers during boot up, like the hard disk drivers and the file system drivers.
- These drivers are stored in an archive called the `initrd`. The boot loader loads the `initrd` into memory as well.
- The initrd is loaded using the `initrd` command.
- The following command loads the `initrd` located at `/boot/initrd.img`

```
initrd /boot/initrd.img
```

- The loaded kernel is executed using the `boot` command. The `boot` command does not take any arguments.

5.2. Menu Interface

- Instead of typing the command each time, the commands can be stored in GRUB’s configuration file, which GRUB uses to display a menu.
- When a menu item is chosen, the commands corresponding to the menu item will be executed.
- GRUB’s configuration file is located in `/boot/grub/grub.cfg`
- The configuration file contains general commands and menu item specific commands.
- The general commands get executed before GRUB displays the menu interface.
- The menu item specific commands get executed when the user selects a menu item.
- Each menu item is started with the `menuentry` command, and is followed by command block to be executed when the menu item is selected.
- The commands mentioned before can be turned into menu item entry by adding the following to the configuration file.

```
menuentry 'Debian GNU/Linux, with Linux 2.6.32-5-amd64' {
        insmod part_msdos
        insmod ext2
        set root='(hd0,1)'
        echo    'Loading Linux 2.6.32-5-amd64 ...'
        linux   /boot/vmlinuz-2.6.32-5-amd64 root=/dev/sda1 ro quiet
        echo    'Loading initial ramdisk ...'
        initrd  /boot/initrd.img-2.6.32-5-amd64
}
```

- The `boot` command is implied and is not required here.
- In the general commands, a couple of GRUB variables are set using the `set` command. Some commonly used variables are `default` and `timeout`.
- During boot up, GRUB displays a menu. If the user does not select a menu item within a timeout period, GRUB boots with the default menu item.
- The `default` variable specifies which menu item is to be booted by default when the timeout expires. The command takes menu entry no. as argument. The numbering starts from zero.
- The following command specifies that the second menu entry is the default.

```
set default=1
```

- The `timeout` variable specifies the no. of seconds to wait before which the default menu item will be selected. The command takes the no. of seconds as argument.
- The following command specifies the timeout period as 10 seconds.

```
set timeout=10
```

- An example of a complete configuration file is shown below.

```
set default=0
set timeout=5

menuentry 'Debian GNU/Linux, with Linux 2.6.32-5-amd64' {
        insmod part_msdos
        insmod ext2
        set root='(hd0,1)'
        echo    'Loading Linux 2.6.32-5-amd64 ...'
        linux   /boot/vmlinuz-2.6.32-5-amd64 root=/dev/sda1 ro quiet
        echo    'Loading initial ramdisk ...'
        initrd  /boot/initrd.img-2.6.32-5-amd64
}
```

5.3. Menu Editing Interface

- The commands corresponding to a menu entry can be edited, by selecting the menu entry and pressing `e`.
- Within the menu editing interface the following keystrokes apply.

Key	Function
`F10`	Boot with modified commands.
`ESC`	Discard edits and go back to menu interface.

6. Kernel

- The kernel initializes itself and the system hardware.
- The kernel mounts the root filesystem at `/`.
- The kernel spawns `init`.
- The kernel accepts various boot arguments, that can be used to modify its behaviour.
- The `root` argument specifies the partition that contains the root filesystem. After the kernel boots up, it mounts the specified root filesystem at `/`.
- The `quiet` boot argument can be used to disable kernel boot messages.
- The `time` boot argument causes the kernel to prefix the kernel messages with a time stamp.
- The `ro` boot argument causes the kernel to mount to the root file system read only.
- The `init` boot argument specifies the location of the `init` program. The shell can be spawned by the kernel instead of `init` by specifing `init=/bin/sh` as boot argument.

7. `init`

- `init` is the first user level process spawned by the kernel, and is responsible for spawing other processes required for the proper operation of the system.
- `init`'s configuration file is `/etc/inittab`.
- What processes are spawned by `init` depends upon the runlevel.

> A runlevel is a software configuration of the system which allows only a selected group of processes to exist.
>
> — man init

- The available runlevels are 0 - 6 and S.
- To initialize the system on boot, the system enters runlevel S.
- To work in single user mode, the system enters runlevel 1.
- To halt, the system enters runlevel 0.
- To reboot, the system enters runlevel 6.
- The processes to be started or terminated when a run level is entered, is specified by the directory `/etc/rcX.d`, where `X` is the runlevel.

- `/etc/init.d` contains shell scripts that can be used to start or stop processes during boot up.
- Example: `/etc/init.d/ssh start` starts the SSH server. `/etc/init.d/ssh stop` stops the SSH server.
- `/etc/rcX.d` contains links to the shell scripts present in `/etc/init.d`
- The following listing shows a clipped output of the `ls` command on `/etc/rc2.d`.

```
$ ls -l /etc/rc2.d/
total 1
lrwxrwxrwx 1 root root  20 2007-09-27 20:21 K77ntp-server -> ../init.d/ntp-server
lrwxrwxrwx 1 root root  18 2007-09-17 23:57 S10sysklogd -> ../init.d/sysklogd
lrwxrwxrwx 1 root root  15 2007-09-17 23:57 S11klogd -> ../init.d/klogd
lrwxrwxrwx 1 root root  17 2007-09-18 00:06 S18portmap -> ../init.d/portmap
lrwxrwxrwx 1 root root  15 2007-09-18 00:09 S19hplip -> ../init.d/hplip
...
lrwxrwxrwx 1 root root  14 2007-09-17 23:57 S89cron -> ../init.d/cron
lrwxrwxrwx 1 root root  18 2007-09-17 23:57 S99rc.local -> ../init.d/rc.local
lrwxrwxrwx 1 root root  19 2007-09-17 23:57 S99rmnologin -> ../init.d/rmnologin
lrwxrwxrwx 1 root root  23 2007-09-17 23:57 S99stop-bootlogd -> ../init.d/stop-bootlogd
```

- The links have the following general syntax.

```
[KS][0-9][0-9]name
```

- The first letter is either `K` or `S`. `K` specifies that a particular process should be started. `S` specifies that a particular process should be stoped.
- The next two digits specifies the order in which the processes should be started/stopped.
- The default runlevel is specified in `/etc/inittab`, by the line that looks like

```
id:2:initdefault:
```

- The `2` in the above line specifies that the default run level is 2.
- `init` finally spawns `getty` , once for each terminal.

8. `getty`

- `getty` after attaching itself to the terminal, `getty` provides a login prompt.
- The following listing shows `getty`'s attached to each virtual terminal.

```
$ ps -A | grep getty
 3114 tty1     00:00:00 getty
 3115 tty2     00:00:00 getty
 3116 tty3     00:00:00 getty
 3117 tty4     00:00:00 getty
 3118 tty5     00:00:00 getty
 3119 tty6     00:00:00 getty
```

- When the user enters the username and hits enters, `getty` executes `login`, which provides the password prompt.
- The user enters the password, `login` verifies the password, spawns the login shell.

9. Further Reading

- Debian Reference: The system initialization - https://www.debian.org/doc/manuals/debian-reference/ch03.en.html
- Kernel Boot Command-Line Parameter Reference - http://files.kroah.com/lkn/lkn_pdf/ch09.pdf