

How a computer boots

04/08/07 – First Draft
Copyright 2007 Jesse Caulfield
<jmc at kbg dot ch>

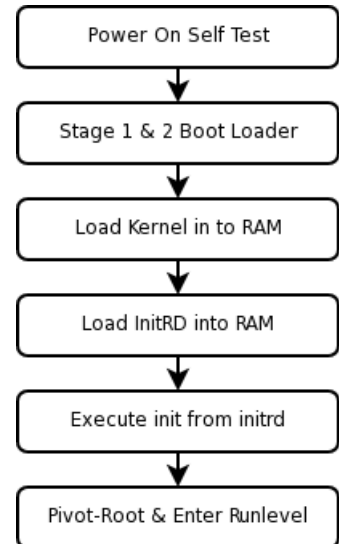
Abstract

All x86-based personal computers (Intel plus AMD) follow the same process from power-on to window presentation.

1. POST: Power On Self Test
2. Boot Loader
3. Load Kernel
4. Copy Initial RAM Disk (initrd) into memory
5. Execute init program
6. Execute pivot-root and enter default run level

POST

When you power on the device it runs through a Power On Self Test. This is essentially a series of procedures etched in the system's firmware meant to discover and account for all the hardware resources present and available, like RAM, PCI devices, network devices and mass storage. The POST behavior presented the user is determined by the system's firmware and BIOS settings and can often be modified to show or hide the gory details. Large PC manufacturers often commission a custom BIOS for their products which present a splash screen with their logo instead of displaying tabular hardware information. There are very few unique PC BIOS versions available, as Award was bought by Phoenix in 1998. Redboot is one example of an open source BIOS, targeted for embedded systems. Newer, Intel-based Apple computers and Intel Itanium-based platforms use a different, newer boot process called EFI (Extensible Firmware Interface), where the BIOS is replaced by a more intelligent and flexible hardware manager. EFI support in linux is presently at a very early stage, with some development work underway in the LILO and GRUB communities.



First & Second Stage Boot Loader

Examples of well known linux first and second stage boot loaders include GRUB, LILO, SYSLINUX and ISOLINUX.

Upon completion of the POST, the system BIOS sequentially looks for any available first stage boot loader at specific byte address location on all available bootable devices identified in the CMOS settings. Examples of bootable devices include hard drives, CDROMs, removable (USB Flash drives), and the network interface. The first stage boot loader is very small, and it's only function is to redirect system booting to the second stage boot loader.

The second stage boot loader is read into memory and actually begins to boot the operating system. The second stage is where options are presented to the user for choosing which operating system or kernel

version to boot and allowing the user to modify any available boot parameters. The GRUB boot menu is an example of a user interface built into the stage 2 boot loader menu.

Linux Kernel

Once selected by the stage 2 boot loader, a linux kernel is loaded into memory and begins initializing system devices and data structures. The kernel then looks for and loads its initial ram disk (initrd) into memory and begins bringing up the system with programs and modules in the ram disk.

The location of the ram disk was either specified when the kernel was compiled or passed to the kernel from the stage 2 boot loader as a boot argument. For example, on a typical linux system, the following parameters are passed to the kernel at boot time:

```
kernel    /vmlinuz-2.6.17-11-generic root=/dev/hda3 ro quiet splash
initrd    /initrd.img-2.6.17-10-generic
```

Copy initial ram disk into memory

The initial ram disk contains essential kernel modules, programs and scripts necessary to properly boot the system.

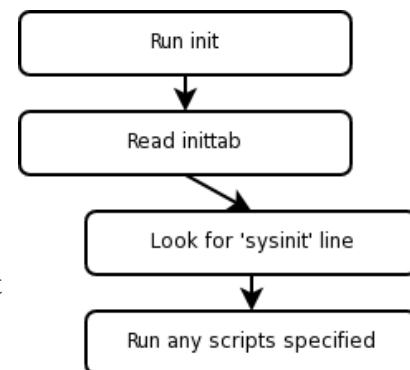
Once all necessary modules are loaded and the ram disk is copied into memory the kernel executes the init program (also located within the ram disk) and begins initializing the system as specified by scripts that it calls.

Init Program

The init program begins operation by reading the inittab file and looking for the line 'sysinit'. Init then executes whatever script is specified there. On a typical linux system sysinit usually refers to a system initialization script like /etc/init.d/rc or /etc/rc.d/rc.sysinit. These scripts then call a series of start up scripts which bring up the system. Examples include assigning the system hostname, set the system clock, bring up networking, check and mount all hard drives, and load any additional necessary kernel modules not within the initrd.

For a Live-CD, we must create a special set of initialization scripts or modify the init program itself. The more manageable method is to add a new set of scripts to the system instead of modify the basic booting procedure. The initialization steps for a Live-CD that differ from a typical hard-disk boot are few but important. The most important two differences are:

- The system must create and use a RAM disk as its operating root directory instead of a hard disk. This is done with a kernel boot parameter.
- The operating root disk must contain contents from the RAM disk and the CDROM in order to access the entire operating system. This is done via the UnionFS kernel module and implemented with a system initialization script.



Execute pivot-root and enter default run level

Once the system initialization scripts are complete the init program executes a neat trick called pivot-root, whereby the system's root directory, which was mounted to the initial ram disk by the kernel during start up, is 'pivoted' to the root directory of the hard-disk file system the sysinit script checked and mounted. Following a successful pivot-root, the init program enters the default runlevel, specified also in the inittab

file by the `initdefault` keyword, and executes any boot scripts called for by that runlevel in the `inittab` file. Once completed, the system is presented to the user with either a 'login:' prompt or a graphical desktop manager like `gdm`.

For a Live-CD, the pivot-root procedure is the same. However, instead of pivoting to a hard disk file system we tell the `init` program to pivot to our RAM disk file system.

A Quick Tour of the Root and Boot File Systems

The root file system contains everything needed to run Linux, and must contain at a minimum the following

- A basic file system structure and minimum set of directories `/dev /proc /bin /etc /lib /usr /tmp`
- Basic system utilities `sh, sl, cp, mv` etc.
- A minimum set of configuration files `rc, inittab, fstab`
- Necessary devices `/dev/hd*, /dev/tty, /dev/ram, etc.`
- Necessary run time libraries `libc.so`

A boot file system contains the boot loader, kernel and initial ram disk (`initrd`). The boot file system is typically contained within the `/boot` directory.

A root file system is everything located under the root directory `"/` except the contents of the boot file system.

Knowing these simple basics, we can easily analyze the core components of any existing Linux distribution to understand how it works.