Linux System Administration-I
CSE-4043

Chapter 3 : Booting and Shutting Down

NIBEDITA JAGADEV

Department of CSE

Asst. Professor

SOA Deemed to be University, Bhubaneswar, Odisha , India

nibeditajagadev@soa.ac.in

Contents

- Bootstrapping
- Steps in Booting Process
- Initialization of the Kernel
- Device detection and configuration
- Creation of spontaneous system processes
- Operator intervention
- Execution of system startup scripts
- Multi-user operation

BOOTSTRAPPING

- **Bootstrapping is the standard term for "starting up a computer."**
- **The operating system's normal facilities are not available during the startup process, so the computer must "pull itself up by its own bootstraps."**
- **During bootstrapping, the kernel is loaded into memory and begins to execute. A variety of initialization tasks are performed, and the system is then made available to users.**
- **The system can be booted two ways:**
  - **Automatically**
    - **Ordinarily, systems are booted this way.**
  - **Manually**
    - **Used by system administrators for setup or troubleshooting.**
    - **Computer is in "Single-User Mode".**

- **Boot time is a period of special vulnerability. Errors in configuration, missing or unreliable equipment, and damaged file systems can all prevent a computer from coming up.**
- **Boot configuration is often one of the first tasks an administrator must perform on a new system, especially when adding new hardware.**
- **When a computer is turned on, it first executes boot code that is stored in ROM. That code in turn attempts to figure out how to load and start the kernel.**
- **The kernel probes the system's hardware and then spawns the system's init process, which is always process number 1.**

- **Before the system is fully booted, file systems must be checked and mounted, and system daemons started .**
- **These procedures are managed by a series of shell scripts (sometimes called “init scripts”) that are run in sequence by init. The exact layout of the startup scripts and the manner in which they are executed vary among systems.**

Six Steps in Booting Process

- Loading and initialization of the kernel
- Device detection and configuration
- Creation of spontaneous system processes
- Operator intervention (single user boot only)
- Execution of system startup scripts
- Multi-user operation

Step 1: Loading and Initialization of the Kernal

- Kernel (e.g., program) is loaded into memory to be executed
- Pathname of kernel is vendor dependent

  Example: **/unix** or **/vmunix**
- 2-stage loading process
  - ❖ 1st: small boot program read into memory to enable kernel loading (outside domain of Unix)
  - ❖ 2nd: kernel runs tests to determine memory availability
    - kernels run in a fixed amount of memory and know what to reserve for internal storage and I/O buffers.

Step 2: Device Detection and Configuration

- Kernel performs hardware check
  - General hardware device info is incorporated in kernel configuration
  - Locate and initialize each device
  - Acquire more info via drivers
    - If not found, will disable hardware
    - If hardware added, must reboot, to access

Step 3: Creation of Spontaneous System Processes

- After basic initialization, kernel creates spontaneous processes in users space
  - Not created via normal UNIX **fork** mechanism
    - **fork** creates copy of the original process, with new ID, that is identical to the parent
  - BSD has 3 processes
    - **Swapper** - process 0; **init** - process1; **pagedaemon** - process 2
  - ATT: varies
    - **sched** - process 0; **init** - process 1; various memory handlers
  - **Note- ONLY INIT USER PROCESS**
- Kernel role complete; **init** handles processes for basic operations and UNIX daemons

Step 4: Operator Intervention (Single-User Boot Only)

- **init** notified via command-line flag from kernel
- **init** creates shell and waits for it to terminate (<control-d> or exit) before continuing on with rest of startup procedure
  - Always in bourne shell (e.g., sh) and runs as root with root partition mounted
  - Available programs located in **/bin**, **/sbin**, **/etc**, and **/usr**; ~ other file systems must be mounted by operator
  - Daemons not available
  - **fsck** (checks and repairs file systems) must be run by hand

Step 5: Execution of System Startup Scripts

- The location, content, and organization of **shell** (e.g., **sh**) scripts vary from system to system
- **BSD**: kept in **/etc** and names begin with **rc**
- **ATT**: kept in **/etc/init.d** with links made to other directories such as **/etc/rc0.d**, **/etc/rc1.d**…
- Examples of tasks performed in initialization scripts
  - Set computer name**;** Set time zone**;** **fsck** disk check**;**
  - Mount system's disks**;** Remove files from **/tmp**
  - Configure network interfaces**;**
  - Start up daemons and network services**;**
  - Turn on accounting and quotas

Step 6: Multi-User Operation

- To complete boot process and allow user access, **init** produces **getty** process on each workstation
- BSD: **init** has only two states ~ single-user and multi-user
- ATT: **init** has one single-user and several multi-user “run levels” to determine which system resources are enabled

Linux System Administration-I
CSE-4043

Chapter 3 Booting and Shutting Down

NIBEDITA JAGADEV

Department of CSE

Asst. Professor

SOA Deemed to be University, Bhubaneswar, Odisha , India

nibeditajagadev@soa.ac.in

Contents

- BIOS
- Master Boot Record
- Booting pcs
- GRUB
- Grub command line option
- **Kernel options**

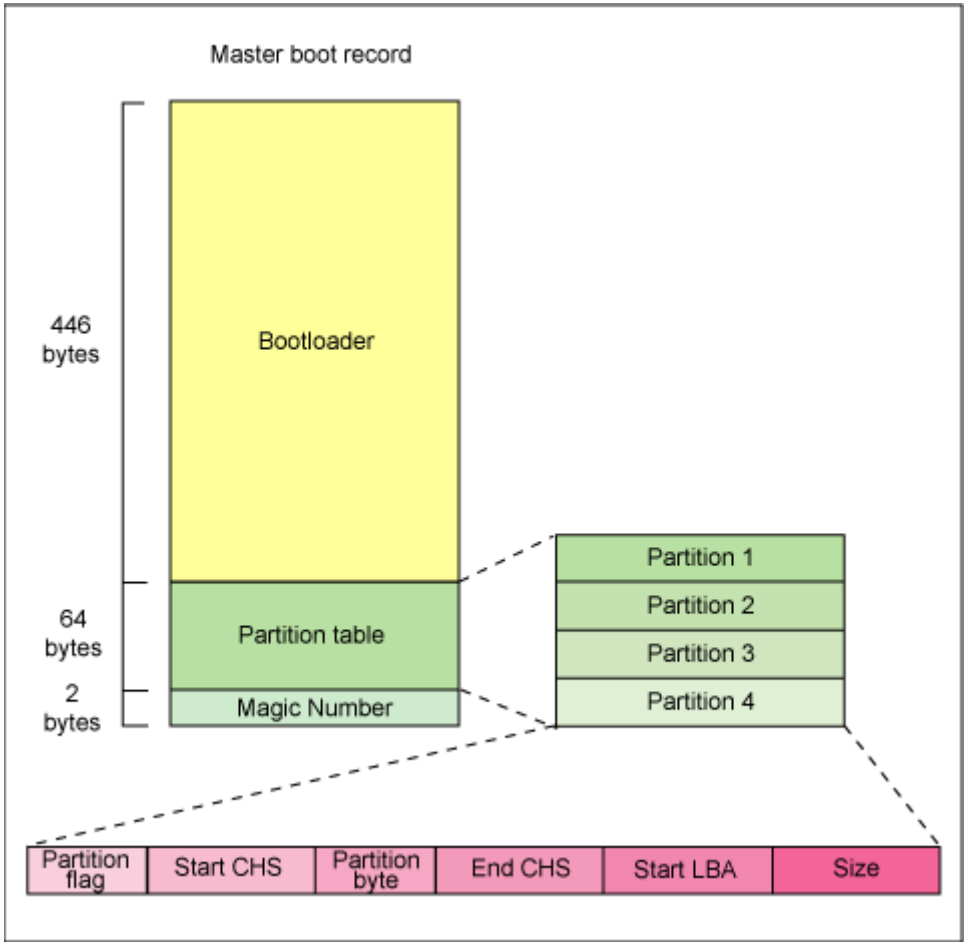
BIOS (Basic Input/Output System)

- BIOS refers to the software code run by a computer when first powered on
- The primary function of BIOS is code program embedded on a chip that recognizes and controls various devices that make up the computer.

MBR (Master Boot Record)

- OS is booted from a hard disk, where the Master Boot Record (MBR) contains the primary boot loader
- The MBR is a 512-byte sector, located in the first sector on the disk (sector 1 of cylinder 0, head 0)
- After the MBR is loaded into RAM, the BIOS yields control to it.

MBR (Master Boot Record)



MBR (Master Boot Record)

- The first 446 bytes are the primary boot loader, which contains both executable code and error message text
- The next sixty-four bytes are the partition table, which contains a record for each of four partitions
- The MBR ends with two bytes that are defined as the magic number (0xAA55). The magic number serves as a validation check of the MBR

Booting the pc

- After the initialization scripts have run, the system is fully operational. System daemons, such as DNS and SMTP servers, are accepting and servicing connections
- When a machine boots, it begins by executing code stored in ROMs.
- On a machine designed explicitly for UNIX or another proprietary operating system, the code is typically firmware that knows how to use the devices connected to the machine, how to talk to the network on a basic level, and how to understand disk-based file systems. Such intelligent firmware is convenient for system administrators. For example, you can just type in the filename of a new kernel, and the firmware will know how to locate and read that file.

Booting the pc

- On a machine designed explicitly for UNIX or another proprietary operating system, the code is typically firmware that knows how to use the devices connected to the machine, how to talk to the network on a basic level, and how to understand disk-based file systems. Such intelligent firmware is convenient for system administrators. For example, you can just type in the filename of a new kernel, and the firmware will know how to locate and read that file. On PCs, the initial boot code is generally called a BIOS (Basic Input/Output System), and it is extremely simplistic compared to the firmware of a proprietary workstation.
- PC has several levels of BIOS: one for the machine itself, one for the video card, one for the SCSI card if the system has one, and sometimes components for other peripherals such as network cards.

Booting the pc

- The BIOS normally lets you to specify an ordered list of preferences such as “Try to boot from a DVD, then a USB drive, then the hard disk.”
- Once the BIOS has figured out what device to boot from, it tries to read the first block of the device. This 512-byte segment is known as the master boot record or MBR. The MBR contains a program that tells the computer from which partition to load a secondary boot program, the “boot loader.”

Booting the pc

- The default MBR contains a simple program that tells the computer to get its boot loader from the first partition on the disk. Once the MBR has chosen a partition to boot from, it tries to load the boot loader specific to that partition. This loader is then responsible for loading the kernel

GRUB: Grand Unified Bootloader

- GRUB is an operating system independent boot loader
- A multiboot software packet from GNU
- Flexible command line interface
- File system access
- Support multiple executable format
- Support diskless system
- Download OS from network
- GRUB reads its default boot configuration from /boot/grub/menu.lst or /boot/grub/grub.conf.

Grub command line option

COMMAND	MEANING
reboot	Reboots the system
find	Finds files on all mountable partitions
root	Specifies the root device (a partition)
kernel	Loads a kernel from the root device
help	Gets interactive help for a command
boot	Boots the system from the specified kernel image

Kernel options

- GRUB lets you pass command-line options to the kernel. These options typically modify the values of kernel parameters, instruct the kernel to probe for particular devices, specify the path to init, or designate a specific root device.

acpi=off	Disables Advanced Configuration and Power Interface components
init=/bin/bash	Starts only the Bash shell; useful for emergency recovery
root=/dev/foo	Tells the kernel to use /dev/foo as the root device
Single^a	Boots to single-user mode

Linux System Administration-I
CSE-4043

Chapter 3 : Booting and Shutting Down

NIBEDITA JAGADEV

Department of CSE

Asst. Professor

SOA Deemed to be University, Bhubaneswar, Odisha , India

nibeditajagadev@soa.ac.in

Contents

- Multibooting
- Booting to Single-user Mode
- Working with Startup Scripts
- Init and its run levels
- Overview of Startup Scripts
- Rebooting & Shutting down
- *Halt and Reboot*

MultiBooting

- Since many operating systems run on PCs, it's fairly common practice to set up a machine to boot several different operating systems. To make this work, you need to configure a boot loader to recognize all the different operating systems on your disks.
- Each disk partition can have its own second-stage boot loader.

  However, the boot disk has only one MBR.

GRUB is really the only option for Intel-based UNIX and Linux systems.

Booting To Single-user Mode

1-Single-user mode with GRUB-

- At the GRUB splash screen, high-light the desired kernel and press ‘a’ to append to the boot options.
- To boot into single-user mode, add the single(or –s on Solaris) flag to the end of the existing kernel options.
- Ex- grub append> ro root=LABEL=/ rhgb quiet single

2-Single-user mode on SPARC

- To interrupt the boot procedure and enter the Open Boot PROM on Sun hardware, press the L1 and ‘a’ keys simultaneously. L1 is sometimes labeled STOP on modern Sun keyboards. From the boot PROM, you can type boot –s to boot to single-user mode.

Working with Startup Scripts

- init executes the system startup scripts. These scripts are really just garden-variety shell scripts that are interpreted by sh or bash.
- Some tasks that are often performed in the startup scripts are
  - Setting the name of the computer
  - Setting the time zone
  - Checking the disks with fsck
  - Mounting the system's disks
  - Removing old files from the /tmp directory
  - Configuring network interfaces
  - Starting up daemons and network services

Working with Startup Scripts

- Removing old files from the **/tmp directory**
- Configuring network interfaces
- Starting up daemons and network services

- Startup scripts displays status messages that can be a tremendous help if the system hangs midway through booting or if we are trying to locate error in one of these scripts.

init and its run levels:

- **init defines at least seven run levels, each of which represents a particular complement** of services that the system should be running.
- **Following points are all generally true:**
- At level 0, the system is completely shut down.
- Levels 1 and S represent single-user mode.
- Levels 2 through 5 include support for networking.
- Level 6 is a “reboot” level.

Overview of startup scripts

- The master copies of the startup scripts live in the **/etc/init.d directory. Each** script is responsible for one daemon or one particular aspect of the system. The scripts understand the arguments **start and stop to mean that the service they** deal with should be initialized or halted.
- As a system administrator we can manually start or stop services by running their associated init.d scripts by hand.

Rebooting & Shutting Down

1-Shutdown:the genteel way to halt the system

Used for to initiate a halt or reboot or to return the system to single-user mode.

2-halt and reboot: simpler ways to shut down

- The halt command performs the essential duties required to shut the system
- down. It is called by **shutdown –h.**
- Halt logs the shutdown, kills nonessential processes, executes the sync system call (called by and equivalent to the sync command), waits for file system writes to complete, and then halts the kernel.
- **Halt -n** prevents the sync call.
- It's used by fsck after it repairs the root partition. If fsck did not **use -n**, the kernel might overwrite fsck's repairs with old versions of the superblock that were cached in memory.
- Reboot is almost identical to halt, but it causes the machine to reboot instead of halting.
- Reboot is called by **shutdown -r**

Thank you