UNIX Startup and Shutdown

Chapter 2

Performance Objectives

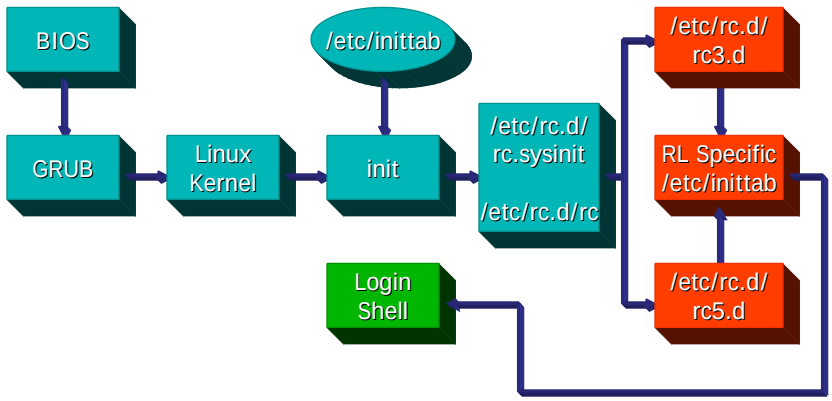
You will learn:

- Bootstrapping.
- Startup scripts and run levels.
- Boot configuration and troubleshooting.
- System shutdown.
- LILO.
- Sync.
- Run levels.

Bootstrapping

- Starting the system.
- Process of loading kernel into memory.
- Boot Modes:
  - Normal
  - Single User
  - Rescue from the CD

Boot Process



Boot Process

- Load and initialize the kernel.
- Detect and configure devices.
- Fork system processes.
  - Stops if Single User mode.
- Run startup scripts.
- Start multiuser operations.

Boot Loader

- ROM loads the boot program from disk.
- The Boot program finds/loads kernel:
  - Checks available memory.
  - Initializes kernel internal data structures.
- GRUB: GRand Unified Bootloader:
  - Can boot multiple operating systems.
  - Boot options can be edited at boot.
- LILO: Linux Boot Loader:
  - Can boot multiple operating systems.
  - Single User: linux single
  - Rescue: linux rescue

/etc/grub.conf

```
default=1
timeout=10
splashimage=(hd0,5)/boot/grub/splash.xpm.gz
title Fedora Core - N321 (2.6.11-1.1369_FC4)
       root (hd0,5)
       kernel /boot/vmlinuz-2.6.11-1.1369_FC4 ro
       root=LABEL=/1 rhgb quiet initrd
              /boot/initrd-2.6.11-1.1369_FC4.img
title Windows XP
       rootnoverify (hd0,0)
       chainloader +1
title Red Hat Enterprise WS (2.6.9-11.EL)
       rootnoverify (hd0,4)
       kernel /boot/vmlinuz-2.6.9-11.EL ro
       root=LABEL=/ rhgb quiet initrd
              /boot/initrd-2.6.9-11.EL.img
```


System Processes

- BSD Systems:
  - swapper – PID 0
  - init – PID 1
  - pagedaemon – PID 2
- AT&T SVR4:
  - sched – PID 0 (invisible under RedHat)
  - init – PID 1
  - /etc/inittab

Startup Scripts

- Hostname
- Timezone
- Check the hard drives.
- Mount the hard drives.
- Remove files from /tmp.
- Configure network interfaces.
- Start daemons and network services.

BSD /etc/rc* Scripts

- /etc/rc
  - Master script
  - Executes supplemental scripts
- Example:

  Supplemental scripts - freeBSD
  - **`/etc/defaults/rc.conf`**
  - **`/etc/rc.conf`**
  - **`/etc/rc.conf.local`**

/etc/inittab

- Initializes system for use.
- Format: id:rl:action:process
  - **`id`**: Uniquely identifies entry.
  - **`rl`**: Run level entry applies to.
  - **`action`**: How to execute process.
  - **`process`**: Process command line.

inittab actions

Action	Purpose
once	Start process once only.
wait	Start process once and wait until finished before going further.
respawn	Restart process if it finishes.
initdefault	Set default run level.

Virtual Consoles

- Started in /etc/inittab:

```
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```


Startup Run Levels

Solaris	RedHat	Mode
1 (S)	1 (S)	Single user.
2	2	Multiuser (no networking).
3	3	Full Multiuser.
4	4	Unused.
5		Power-off shutdown.
	5	X11.
6	6	Reboot.
0	0	Halt.

Run Levels - Setting

- Started in **`/etc/inittab`**:

```
id:3:initdefault:

  l0:0:wait:/etc/rc.d/rc 0
  l1:1:wait:/etc/rc.d/rc 1
  l2:2:wait:/etc/rc.d/rc 2
  l3:3:wait:/etc/rc.d/rc 3
  l4:4:wait:/etc/rc.d/rc 4
  l5:5:wait:/etc/rc.d/rc 5
  l6:6:wait:/etc/rc.d/rc 6
```


Run Level Scripts

- /Actual scripts - /etc/rc.d/init.d/
- /etc/rc.d/rc#.d/
  - Symbolic links to /etc/init.d scripts.
  - S## - Start scripts.
  - K## - Stop scripts.
  - /etc/sysconfig/ - Script configuration files.

Startup Messages

- Displayed to console.
- RedHat – daemon [ok]
- /bin/dmesg
  - History of boot messages.
  - Convenient to search.
- RedHat allows for interactive setup (“i”).

System Shutdown

- Turning off the power can detrimentally effect the operating system.
- Reboot.
  - reboot
  - shutdown –r
- Halting the system.
  - halt
  - shutdown –h
- Changing the Run Level.
  - telinit <mode>
  - shutdown –i<mode>

Shutdown – When to

- Failures.
- Maintenance and Upgrades.
- Regularly Scheduled:
  - Housecleaning
  - Window for Maintenance/Upgrades
- User Notification:
  - /etc/motd
  - Email
  - Support web pages

Bootstrapping and Shutdowns - Overview

- During bootstrapping, the computer first loads a small piece of code called the bootstrap loader, which in turn loads and starts the operating system.
- The bootstrap loader is usually stored in a fixed location on a hard disk.
- The reason for this two step process is that the operating system is large and complicated, however, the first piece of code that the computer loads must be very small (a few hundred bytes), in order to avoid making the firmware unnecessarily complicated.

PC Bootstrapping

- Different computers perform bootstrapping differently.
- For PCs, the computer (its BIOS) reads in the first sector (called the boot sector) of a floppy or hard disk.
- The bootstrap loader is contained within this sector.
- It loads the operating system from elsewhere on the disk (or from some other place).

Hardware and Device Drivers

- After Linux has been loaded, it initializes the hardware and device drivers, and then runs init.
- init starts other processes which allows users to log in, and do things.

Shutdown Procedure

- In order to shut down a Linux system, all the processes are instructed to terminate.
  - This closes any files and performs other housekeeping tasks in order to keep things tidy.
  - Next the filesystems and swap areas are unmounted.
  - A message is sent to the console stating that that the power can be turned off.
- If the proper procedure is not followed, there is the likelihood that problems will result and most importantly, the filesystem buffer cache may not flush properly.
- In this situation, all the data in the buffer will be lost and the filesystem on disk is inconsistent, and therefore possibly unusable.

Booting Process

- When a PC workstation is booted, the BIOS performs various diagnostic tests and then commences the actual booting.
- The boot process initially starts with the boot sector on the hard disk.
- The boot sector to be booted from is known as the master boot record.
  - This is because a hard disk can contain several partitions, each with their own boot sectors.

Boot Sector

- The boot sector contains a small program, which is small enough to fit into one sector, whose responsibility is to read the actual operating system from the disk and start it.
- When booting from the hard disk, the code in the master boot record will:
  - examine the partition table, which is also in the master boot record.
  - identify the active partition, which is the partition that has been marked as bootable.
  - read the boot sector from that partition.
  - start the code in that boot sector.
- The code in the partition's boot sector will read the kernel from the partition and then perform initialization.

LILO

- When booting with LILO, the process will normally go directly ahead and read in and boot the default kernel.
  - It is also possible to configure LILO to be able to boot one of several kernels, or other operating systems than Linux.
  - The user can have the capability to select which kernel or operating system is to be booted at boot time.
- LILO can be configured to have the alt, shift, or ctrl key at boot time ask what is to be booted and not boot the default right away.
- Alternatively, LILO can be configured to present the boot options, inclusive of an optional timeout, which will result in the default kernel being booted.

Linux Kernel

- After the Linux kernel has been read into memory and has been started, the following events occur:
  - The Linux kernel is installed compressed, it will need to uncompress itself.
    - The first part of the kernel image contains a small program that does this.
  - If a super−VGA card has been installed which Linux recognizes, there will be special text modes, and Linux will ask which mode is to be used.
  - The kernel then checks what other hardware has been installed on the system (hard disks, floppies, network adapters, etc.), and configures its device drivers appropriately.
    - While doing this, it outputs messages about its findings.

Linux Kernel

  - The kernel will then attempt to mount the root filesystem.
    - The root filesystem is configurable at compilation time, or any time with rdev or LILO.
  - The kernel then starts the program init (located in /sbin/init) in the background .
    - This will always become process number 1.
    - init performs various startup chores.
  - init then switches to multi−user mode, and starts a getty for virtual consoles and serial lines.
    - getty is the program which provides the capability for allowing users to log in via virtual consoles and serial terminals.
    - Depending on how it is configured; init may also start some other programs.
  - The boot is then complete, and the system is up and running normally.

Shutdowns

- It is important to follow the correct procedures when shutting down a Linux system.
- If this is not done, filesystems will become unstable and files damaged.
  - This is because Linux has a disk cache that will not immediately perform a write to disk; it will only be performed at intervals.
- Another reason to not turn off the power switch is that on a multitasking system there in all likelihood will be a variety of processing going on in the background and shutting the power can result in a number of problems.
- Using the proper shutdown sequence, will ensure that all the background processes can save their data.

Single User Shutdown

- The command for properly shutting down a Linux system is shutdown.
- It is typically used in one of two ways.
  - If a system is being run when there is only a single user, the standard procedure is to:
    - Quit all running programs.
    - Log out on all virtual consoles.
    - Log in as root on a virtual console or remain logged in as root.
      - Change to the root's home directory or the root directory, in order to avoid problems with unmounting.
      - Give the command shutdown:

        −h now
      - In order to invoke a delay, substitute now with a plus sign and a number in minutes.

Multi User Shutdown

- Alternatively, if a system has many users, the command shutdown −h +time message can be used, where:
  - time is the time in minutes until the system is halted.
  - message is a short explanation of why the system is shutting down.

    # shutdown −h +10 'We will install a new disk. System should
    > be back on−line in three hours.'
- This will send out a warning message that the system will shut down in ten minutes, and that users should log off or risk losing data.
- The warning is automatically repeated several times before the boot, with shorter and shorter intervals as the time runs out.

Power Off the Machine

- When the a “true” shut down commences after the delays:
  - all filesystems, except the root, are unmounted.
  - user processes, if anybody is still logged in, are killed.
  - daemons are shut down.
  - filesystem are unmounted.
  - When that is done, init prints out a message that the machine can be powered down.
  - The power switch can then be turned to the off position.

Shut Down Problem

- Sometimes it will not be possible to shut down properly.
  - For instance, if the kernel crashes new commands will not be able to be entered.
  - Accordingly, it may well be difficult or impossible to perform a normal shutdown.
  - In this situation turning off the power and hoping that minimal or no damage is being caused to the system may well be all that can be done.
- If the troubles are somewhat less severe and the kernel and the update program still run normally, the recommendation is to wait a couple of minutes in order to allow the system a chance to flush the buffer cache, and then turn off the power.

sync

- Some individuals will prefer to shut down using the command sync three times, waiting for the disk I/O to stop, then turn off the power.
  - If there are no running programs, this is essentially equivalent to using shutdown.
  - However, it does not unmount any filesystems and this can lead to problems with the ext2fs "clean filesystem" flag.
- The triple−sync method is not a recommended practice.

Rebooting

- This can be accomplished by first shutting it down completely, turning power off, and then turning it back on.
- A simpler way is to ask shutdown to reboot the system, instead of halting it.
- This is accomplished by using the −r option to shutdown.
  - For example:
    - the command shutdown −r now.

Rebooting

- Most Linux systems run shutdown −r now when ctrl−alt−del is pressed on the keyboard.
- This reboots the system.
- The action on ctrl−alt−del is configurable, however, and it might be better to allow for some delay before rebooting on a multiuser machine.

init

- init is a program that is essential to the operation of a Linux system.
- A quality Linux distribution will come with a configuration for init that will work for most systems, and on these systems there is nothing that needs to be done to init.
- Typically, it will not be necessary to be concerned with init when serial terminals have dial−in (not dial−out) modems, or if it is necessary to change the default run level.

init Location

- The kernel looks for init in a few locations that have been historically used for storage; however, the proper location for on a Linux system is /sbin/init.
- If the kernel can't find init, it tries to run /bin/sh, and if that also fails, the startup of the system fails.
- When init starts, it finishes the boot process by performing a number of administrative tasks, such as:
  - checking filesystems.
  - cleaning up /tmp.
  - starting various services.
  - starting a getty for each terminal and virtual console where users should be able to log in.

Adopts Orphan Processes

- After the system is up and running properly, init restarts getty for each terminal after a user has logged out in order that the next user can log in.
- init also adopts orphan processes:
  - When a process starts a child process and dies before its child, the child immediately becomes a child of init.
  - This is important information; which will make it easier to understand process lists and process tree graphs.

/etc/inittab

- When it starts up, init reads the /etc/inittab configuration file.
  - While the system is running, the file will be re−read, when the HUP signal is sent.
    - This feature makes it unnecessary to boot the system to make changes made to the init configuration take effect.

/etc/inittab

- Lines in /etc/inittab consist of four colon-delimited fields:
  id:runlevels:action:process

The fields are:

- **Id**
  - This identifies the line in the file.
    - For getty lines, it specifies the terminal it runs on.
      - The characters after /dev/tty in the device file name.
    - For other lines, except for length restrictions, it doesn't matter.
    - However, it should be unique.
- **runlevels**
  - The runlevels are given as single digits, without delimiters.
- **action**
  - The action that should be taken by the line.
    - e.g., respawn to run the command in the next field again, when it exits, or once to run it just once.
- **process**
  - The command to run.

/etc/inittab

- In order to start a getty on the first virtual terminal (/dev/tty1) in all the normal multi−user run levels (2−5), it will be necessary to run the following line:

  1:2345:respawn:/sbin/getty 9600 tty1

  - The first field indicates that this is the line for /dev/tty1.
  - The second field indicated that it applies to run levels 2, 3, 4, and 5.
  - The third field indicates that the command should be run again, after it exits in order that it a log in, log out, and then relog in can be performed.
- The last field is the command that runs getty on the first virtual terminal.

Run Levels

- A run level is a state of init and the entire system defines what system services are operating.
- Run levels are identified by numbers.
- There is no consensus of how to use the user defined run levels (2 through 5).
- Some system administrators use run levels to define which subsystems are working.
  - e.g., whether X is running, whether the network is operational, and so on.
- Another approach is to have all subsystems always running or starting and stopping them individually.
  - In general, run levels will be too coarse for controlling systems.

Run Level List

- 0
  - Halt the system
- 1
  - Single−user mode
    - for special administration
- 2−5
  - Normal operation
    - user defined.
- 6
  - Reboot

Run Level Configuration

- Run levels are configured in /etc/inittab by lines such as:
- l2:2:wait:/etc/init.d/rc 2
  - The first field is an arbitrary label.
  - The second field indicates that this applies to run level 2.
  - The third field indicated that init should run the command in the fourth field once, when the run level is entered, and that init should wait for it to be completed.
  - The /etc/init.d/rc command runs whatever commands are necessary to start and stop services to enter run level 2.
  - The command in the fourth field performs all the work associated with setting up a run level.
    - It starts services that aren't already running, and stops services that shouldn't be running in the new run level any more.
- The distribution of Linux will determine exactly what the command is, and how run levels are configured.

Special Configuration in /etc/inittab

- The /etc/inittab has special features that allow init to react to special circumstances.
- These special features are marked by special keywords in the third field.
- Examples:
  - powerwait
    - Allows init to shut the system down, when the power fails. This assumes the use of a UPS, and software that watches the UPS and informs init that the power is off.
  - ctrlaltdel
    - Allows init to reboot the system, when the user presses ctrl−alt−del on the console keyboard.
    - The system administrator can configure the reaction to ctrl−alt−del to be something else instead.
  - sysinit
    - Command to be run when the system is booted.
      - This command usually cleans up /tmp.