Chapter 8
System Initialization and X Windows
Objectives

- Summarize the major steps necessary to boot a Linux system
- Detail the configuration of common Linux boot loaders
- Explain the UNIX SysV and Systemd system initialization processes
- Start, stop, and restart daemons
- Configure the system to start and stop daemons upon entering certain runlevels and targets
Objectives

- Explain the purpose of the major Linux GUI components: X Windows, window manager, and desktop environment
- List common window managers and desktop environments used in Linux
- Configure X Windows settings
The Boot Process

- **POST** (Power On Self Test): series of tests run when computer initializes
  - Ensures functionality of hardware
- Boot loader: program used to load an OS
  - Locates and executes the kernel of the OS
- MBR might contain pointer to a partition containing a boot loader on the first sector
- Active partition: partition pointed to by MBR
  - One per HDD
The Boot Process

- `/boot`: directory containing kernel and boot-related files
- `Vmlinuz-<kernel version>`: Linux kernel file
- Daemon: system process that performs useful tasks
  - e.g., printing, scheduling, OS maintenance
- Init (initialize) daemon: first process started by Linux kernel
  - Loads all other daemons
  - Brings system to usable state
The Boot Process

Figure 8-1: The boot process
Boot Loaders

- Primary function of boot loaders: load the Linux kernel into memory

- Other functions:
  - Passing information to the kernel during startup
  - Booting another OS: known as dual booting

- Two most common boot loaders:
  - Linux Loader (LILO)
  - GRand Unified Boot loader (GRUB)/GRUB2
LILO

- Stands for Linux Loader
- Traditional Linux boot loader
  - No longer used by most modern Linux distributions
  - Often found on other distributions that require a smaller boot loader than GRUB
- Typically located on MBR/GPT
- Lilo boot: prompt appears following BIOS POST
  - Allows choice of OS to load at startup
- To configure, edit /etc/lilo.conf file
LILO Example

prompt
timeout=50
default=linux
boot=/dev/had
map=/boot/map
install=/boot/boot.b
lba32
append="mem=4096M"
append="hd=2100,16,63"

image=/boot/vmlinux=3.11.10-301.fc20.x86_64
label=linux
initd=/boot/vmlinux=3.11.10-301.fc20.x86_64.img
read-only
append="rhgb quiet root=/dev/hda1"
### LILO

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>image=</td>
<td>Specifies the absolute pathname to the Linux kernel.</td>
</tr>
<tr>
<td>root=</td>
<td>Specifies the device and the partition that contains the Linux root filesystem.</td>
</tr>
<tr>
<td>prompt</td>
<td>Displays a LILO boot prompt provided there is no message= keyword specified.</td>
</tr>
<tr>
<td>message=</td>
<td>Specifies the absolute pathname to the file that contains a graphical LILO screen that can be used instead of a prompt. You can press Ctrl+x at this graphical screen to switch to the LILO boot prompt.</td>
</tr>
<tr>
<td>timeout=</td>
<td>Specifies the number of 1/10th seconds to wait for user input before loading the default operating system kernel.</td>
</tr>
<tr>
<td>default=</td>
<td>Specifies the label for the default operating system kernel.</td>
</tr>
<tr>
<td>label=</td>
<td>Specifies the friendly name used to identify an operating system kernel within boot loader screens.</td>
</tr>
<tr>
<td>boot=</td>
<td>Specifies where LILO should be installed. If the device specified is a partition on a hard disk, LILO is installed at the beginning of the partition. If the device specified is a disk, LILO is installed to the MBR on that device.</td>
</tr>
<tr>
<td>linear</td>
<td>Specifies that LILO uses linear sector addressing.</td>
</tr>
<tr>
<td>read-only</td>
<td>Initially mounts the Linux root filesystem as read-only to reduce any errors with running fsck during system startup.</td>
</tr>
<tr>
<td>initrd=</td>
<td>Specifies the pathname to a ramdisk image used to load modules into memory needed for the Linux kernel at boot time.</td>
</tr>
<tr>
<td>password=</td>
<td>Specifies a password required to boot the Linux kernel.</td>
</tr>
<tr>
<td>append=</td>
<td>Specifies parameters that are passed to the Linux kernel when loaded.</td>
</tr>
<tr>
<td>map=</td>
<td>Specifies the file that contains the exact location of the Linux kernel on the hard disk.</td>
</tr>
<tr>
<td>install=</td>
<td>Specifies the file that contains the physical layout of the disk drive.</td>
</tr>
<tr>
<td>lba32</td>
<td>Specifies large block addressing (32-bit) for hard drives that have more than 1024 cylinders.</td>
</tr>
</tbody>
</table>

**Table 8-1: Common /etc/lilo.conf keywords**
LILO

- `append=` keyword (in `/etc/lilo.conf`): Useful for manually passing information to Linux kernel
  - Can pass almost any hardware information
- Must reinstall LILO if `/etc/lilo.conf` file altered
- `lilo` command: Reinstalls LILO
  - `-u` option: Uninstall LILO
## LILO

<table>
<thead>
<tr>
<th>Error Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>The first part of the LILO boot loader failed to load, usually as a result of incorrect hard disk parameters. Simply rebooting the machine sometimes fixes this problem. However, you might also need to add the word “linear” to /etc/lilo.conf.</td>
</tr>
<tr>
<td>LI</td>
<td>The second part of the LILO boot loader failed to load or the /boot/boot.b file is missing. Adding the word “linear” to /etc/lilo.conf might fix the problem.</td>
</tr>
<tr>
<td>Lil</td>
<td>LILO has loaded properly but cannot find certain files required to operate, such as the /boot/map and /boot/boot.b files. Adding the word “linear” to /etc/lilo.conf might fix the problem.</td>
</tr>
</tbody>
</table>

Table 8-2: LILO error messages
GRUB

- Originally created to replace LILO
- Stage1: first major part of GRUB
  - Typically resides on MBR/GPT
  - Remaining parts of the boot loader (Stage1.5 and Stage2) reside in the /boot/grub directory
- Stage1.5: loads filesystem support and Stage2
- Stage2: performs the actual boot loader functions
  - Displays graphical boot loader screen
GRUB

Figure 8-2: GRUB boot loader screen
GRUB

- To configure, edit the /boot/grub/grub.conf file
  - Read directly by Stage2 boot loader
  - HDDs and partitions are identified by numbers
    - Format: (hd<drive#>,<cl:partition#>)
- GRUB root partition: partition containing Stage2 boot loader and grub.conf file
- GRUB normally allows manipulation of boot loader during system startup
  - To prevent this, password protect GRUB modifications during boot time
GRUB

- If you press any key during first five seconds after the BIOS POST, you will get a graphical GRUB boot menu screen
  - Allows you to manipulate the boot process
  - Get a grub> prompt to enter commands
    - Help screen provides list of all available commands
- `grub-install` command: installs GRUB boot loader
  - Typically for reinstallation when GRUB becomes damaged
GRUB

Figure 8-3: The GRUB boot menu for a Fedora system
GRUB2

- Grand Unified Bootloader version 2 (GRUB2) - most common boot loader used on modern Linux systems
- Similar structure to GRUB
  - Stage2 loads a terminal-friendly boot loader screen
- Main configuration file for GRUB2 is /boot/grub/grub.cfg (or /boot/grub2/grub.cfg)
- When a new device driver needs to be loaded by the boot loader
  - The package often adds a file to /etc/default/grub.d
GRUB2

Figure 8-4: The GRUB2 boot screen on a Fedora 20 system
GRUB2

To configure or edit:

- Add or modify existing lines within the /etc/default/grub file

After modifying the file or adding scripts to the /etc/grub.d directory

- Run the `grub2-mkconfig` command to rebuild the /boot/grub/grub.cfg file

Use the `grub2-install` command

- If the GRUB2 boot loader becomes damaged
Linux Initialization

- The kernel resumes control after Linux is loaded
  - Executes first daemon process (init daemon) which then performs a system initialization process
    - Brings the system into a usable state
- Recent Linux distributions have adopted the Systemd system initialization process
  - Older Linux systems used a UNIX standard called SysV
- Systemd is completely compatible with SysV
  - Implements new features for management
Working with the UNIX SysV System Initialization Process

- Two system initialization processes
  - Traditional SysV
  - Upstart (Systemd) vi

- In both systems
  - The init daemon runs a series of scripts to start other daemons (service units) to provide system services

- The init daemon is responsible for starting and stopping daemons after system initialization
Runlevels

- Runlevel (target units): defines the number and type of daemons loaded into memory and executed by the kernel
- The init daemon responsible for changing runlevels
  - Often called initstates
- Seven standard runlevels (see Table 8-3)
- runlevel command: displays current and most recent runlevel
- init command: changes the runlevel on a system
  - telinit command: Alias to init command
# Runlevels

<table>
<thead>
<tr>
<th>Runlevel</th>
<th>Common Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Halt</td>
<td>A system that has no daemons active in memory and is ready to be powered off.</td>
</tr>
<tr>
<td>1, 5, S, single</td>
<td>Single User Mode</td>
<td>A system that has only enough daemons to allow one user (the root user) to log in and perform system maintenance tasks.</td>
</tr>
<tr>
<td>2</td>
<td>Multiuser Mode</td>
<td>A system that has most daemons running and allows multiple users the ability to log in and use system services. Most common network services other than specialized network services are available in this runlevel as well.</td>
</tr>
<tr>
<td>3</td>
<td>Extended Multiuser Mode</td>
<td>A system that has the same abilities as Multiuser Mode, yet with all extra networking services started (e.g., SNMP, NFS).</td>
</tr>
<tr>
<td>4</td>
<td>Not used</td>
<td>Not normally used, but can be customized to suit your needs.</td>
</tr>
<tr>
<td>5</td>
<td>Graphical Mode</td>
<td>A system that has the same abilities as Extended Multiuser Mode, yet with a graphical login program. On systems that use the GNOME desktop, this program is called the GNOME Display Manager (gdm) and is started on tty1 or tty7 to allow for graphical logins.</td>
</tr>
<tr>
<td>6</td>
<td>Reboot</td>
<td>A special runlevel used to reboot the system.</td>
</tr>
</tbody>
</table>

Table 8-3: Linux runlevels
The /etc/inittab File (SysV)

- Unless otherwise specified, the init daemon enters the default runlevel indicated in the /etc/inittab file
  - Syntax: `id:5:initdefault`
- Contains a single uncommented line that configures the default runlevel
Runtime Configuration Scripts (SysV)

- Runtime configuration (rc) scripts: scripts that prepare the system, start daemons and bring the system to a usable state

- init daemon executes script for default runlevel (5)

  `/etc/rc.d/rc5.d` script

  - Executes all files that start with S or K in the `/etc/rc.d/rc5.d` directory
Runtime Configuration Scripts (SysV)

- When user specifies runlevel 1, init daemon runs default script but executes files in the /etc/rc.d/rc1.d directory
- Each file in an /etc/rc.d directory is a symbolic link to a script that can be used to start or stop a daemon
  - Depending on whether the symbolic link filename started with an S (start) or K (kill/stop)
Runtime Configuration Scripts (SysV)

Figure 8-5: A traditional UNIX SysV system initialization process
Runtime Configuration Scripts (SysV)

- On Linux systems that use the upstart init system
  - The /etc/rc.d directories are not used
- The init daemon identifies the default runlevel in the /etc/inittab file
  - Directly executes the rc scripts within the /etc/init.d directory
- Each daemon has a separate configuration file within the /etc/init directory
  - Uses standard wildcard notation to identify runlevels it should be started or stopped in
Runtime Configuration Scripts (SysV)

Figure 8-6: An upstart system initialization process
Starting and Stopping Daemons Manually (SysV)

- Most daemons accept arguments `start`, `stop`, `restart`
  - Can be used to manipulate daemons after system startup
- For example, to restart the cron daemon, type
  - `/etc/init.d/cron restart`
- `service` command: start, stop, or restart daemons within `/etc/rc.d/init.d` directory
- The upstart init system also provides the `stop`, `start`, and `restart` commands
Configuring Daemons to Start in a Runlevel (SysV)

To configure a daemon to start or stop in a particular runlevel:

- Create or modify symbolic links within `/etc/rc[runlevel].d` directories
- `chkconfig` command: view and modify daemons that are started in each runlevel
- The `chkconfig` command is not available in Ubuntu Server 14.04
  - Use the `update-rc.d` command to configure files within `/etc/rc[runlevel].d` directories
Working with the Systemd System Initialization Process

- Systemd is similar to SysV
  - Can also be used to start, stop, and configure many other OS components
- Each OS component is called a unit
- Daemons are called service units
- Runlevels are called target units (or targets)
Working with the Systemd

System Initialization Process

- Each target maps to a UNIX SysV runlevel:
  - `poweroff.target` = Runlevel 0
  - `rescue.target` = Runlevel 1 (Single User Mode)
  - `multi-user.target` = Runlevel 2, 3, and 4
  - `graphical.target` = Runlevel 5
  - `reboot.target` = Runlevel 6

- Default target on a system with a GUI installed is the `graphical.target`

- To configure a different target, update the `/etc/systemd/system/default.target` symbolic link
Figure 8-7: A Systemd system initialization process
Working with the Systemd

System Initialization Process

- **systemctl command**: used to start and stop daemons, as well as configure them to automatically start during system initialization

  - **Syntax**:
    - `systemctl restart crond.service`

- **systemctl arguments**
  - **status**: see detailed information about a daemon
  - **enable**: configure a daemon to start in the default target
  - **isolate**: to change between targets
    - `systemctl isolate runlevel5.target`
The X Windows System: Linux GUI Components

Figure 8-8: Components of the Linux GUI
X Windows

- X Windows: core component of Linux GUI
  - Provides the ability to draw graphical images in windows that are displayed on terminal screen
  - Sometimes referred to as X server
- X client: programs that tell X Windows how to draw the graphics and display the results
  - Need not run on same computer as X Windows
- XFree86: OSS version of X Windows
  - Originally intended for Intel x86 platform
Windows Managers and Desktop Environments

- **Window manager**: modifies look and feel of X Windows
  - `yum install wmctrl`
  - `wmctrl -m` # Displays information about window manager

- **Desktop environment**: standard set of GUI tools
  - Works with a window manager to provide standard GUI environment
  - Provides toolkits that speed up process of creating new software
  - KDE and GNOME are the most common
## Windows Managers and Desktop Environments

<table>
<thead>
<tr>
<th>Window Manager</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiz Fusion</td>
<td>A highly configurable and expandable window manager based on the Compiz and Beryl window managers that utilizes 3D acceleration on modern video cards to produce 3D graphical effects, including 3D window behavior and desktop cube workspaces.</td>
</tr>
<tr>
<td>enlightenment</td>
<td>A highly configurable window manager that allows for multiple desktops with different settings. It is commonly used by the GNOME desktop.</td>
</tr>
<tr>
<td>fvwm</td>
<td>A window manager based on twm that uses less computer memory and gives the desktop a 3-D look; its full name is “Feeble Virtual Window Manager.”</td>
</tr>
<tr>
<td>kwin</td>
<td>The window manager used for the KDE desktop.</td>
</tr>
<tr>
<td>lxde</td>
<td>A window manager specifically designed for use on underpowered systems such as netbooks, mobile devices, and legacy computers; its full name is “Lightweight X Desktop Environment.”</td>
</tr>
<tr>
<td>metacity</td>
<td>The default window manager used by the GNOME version 1 and 2 desktop environments.</td>
</tr>
<tr>
<td>mutter</td>
<td>The default window manager used by the GNOME version 3 desktop environment.</td>
</tr>
<tr>
<td>mwm</td>
<td>A basic window manager that allows the user to configure settings using standard X utilities; its full name is “Motif Window Manager.”</td>
</tr>
<tr>
<td>sawfish</td>
<td>A window manager commonly used for the GNOME desktop. It allows the user to configure most of its settings via tools or scripts.</td>
</tr>
<tr>
<td>twm</td>
<td>One of the oldest and most basic window managers; its full name is “Tab Window Manager.”</td>
</tr>
<tr>
<td>wmaker</td>
<td>A window manager that provides drag-and-drop mouse movement and imitates the NeXTSTEP operating system interface made by Apple Computer, Inc.; its full name is “Window Maker.”</td>
</tr>
</tbody>
</table>

### Table 8-4: Common window managers
Windows Managers and Desktop Environments

- K Windows Manager (kwm): window manager that works under KDE
- Qt toolkit: software toolkit included in KDE
- GNOME desktop environment: default desktop environment in most Linux distributions
  - Uses mutter window manager and the GTK+ toolkit
- Can configure KDE or GNOME to use different window manager
  - e.g., Compiz Fusion
Starting and Stopping X Windows

- When the init daemon boots to runlevel 5 or graphical.target, GNOME Display Manager (GDM) starts
  - Displays graphical login screen
  - Allows user to choose the desktop environment
- If you use runlevel1 (or rescue.target) or runlevel 2-4, the GDM is not started by default
  - Type `startx` at a character terminal to start X Windows and the default window manager
Configuring X Windows

• X Windows interfaces with video hardware
  • Requires information regarding keyboard, mouse, monitor, and video adapter card
• Attempts to automatically detect required information
  • If automatic detection fails, user needs to specify correct hardware information manually
Configuring X Windows

- User-configured settings are stored in files under the /etc/X11/xorg.conf.d directory.
- Common settings such as the display resolution can be modified using the Displays utility within the GNOME desktop environment.
- You can manually run the `system-config-keyboard` command to configure keyboard.
- Use `xvidtune` utility to fine-tune the vertical refresh rate (vsync) and horizontal refresh rate (hsync).
Configuring X Windows

Figure 8-11: Selecting a keyboard layout
Configuring X Windows

Figure 8-12: The xvidtune utility
Summary

- Boot loaders are typically loaded by the system BIOS from the MBR/GPT or the first sector of the active partition of a hard disk.
- After the boot loader loads the Linux kernel, a system initialization process proceeds to load daemons that bring the system to a usable state.
- There are two common system initialization processes: UNIX SysV and Systemd.
Summary

- UNIX SysV uses seven runlevels to categorize a Linux system based on the number and type of daemons loaded in memory.
- Systemd uses five standard targets that correspond to the seven UNIX SysV runlevels.
- The init daemon is responsible for loading and unloading daemons when switching between runlevels and targets.
- Daemons are typically executed during system initialization via rc scripts.
Summary

- The `service` command is commonly used to start, stop, and restart UNIX SysV daemons, and the `systemctl` command is commonly used to start, stop, and restart Systemd daemons.
- Use the `chkconfig` or `update-rc.d` commands to configure UNIX SysV daemon startup at boot time.
- The Linux GUI has several interchangeable components: X server, X clients, window manager, and optional desktop environment.
Summary

- X Windows is the core component of the Linux GUI that draws graphics to the terminal screen.
- The hardware information required by X windows is automatically detected, but can be modified using several utilities.