

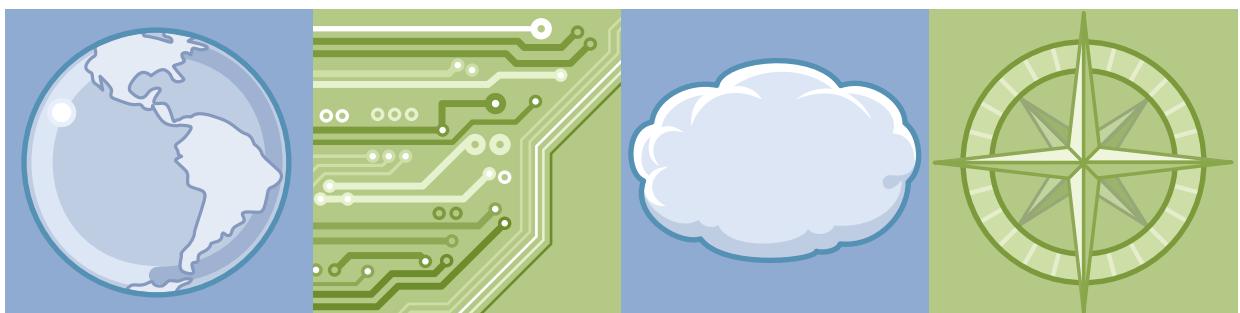


# IBM Training

Instructor Guide

## **Linux Jumpstart for UNIX System Administrators**

Course code LX15 ERC 8.0



## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

AIX®	GPFS™	Notes®
Power®	PowerPC®	System i®
System p®	System x®	System z®
400®		

PostScript is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

## September 2013 edition

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

# Contents

<b>Trademarks</b> .....	<b>xiii</b>
<b>Instructor course overview</b> .....	<b>xv</b>
<b>Course description</b> .....	<b>xvii</b>
<b>Agenda</b> .....	<b>xix</b>
<b>Unit 1. Introduction to Linux</b> .....	<b>1-1</b>
Unit objectives .....	1-2
A short history of Linux (1 of 2) .....	1-4
A short history of Linux (2 of 2) .....	1-7
What's so special about Linux (GNU/Linux)? .....	1-10
Effects of the license model .....	1-13
Linux has become a way of life .....	1-16
Linux today .....	1-19
Linux hardware support .....	1-22
Looking at Linux .....	1-25
Unit review .....	1-27
Checkpoint .....	1-29
Unit summary .....	1-31
<b>Unit 2. Installing Linux</b> .....	<b>2-1</b>
Unit objectives .....	2-2
2.1. Manual installation .....	2-5
Preparing a system for installation .....	2-6
Installing Linux .....	2-8
Network installations .....	2-10
Installation steps .....	2-13
Select language, keyboard, and mouse .....	2-15
Install class .....	2-17
Disk partitioning .....	2-19
Configure a boot loader .....	2-23
Configure network .....	2-26
Configure root and user accounts .....	2-28
Select package groups .....	2-30
Configure X .....	2-32
Other (optional) installation screens .....	2-34
Installing packages .....	2-36
Post-install configuration .....	2-38
2.2. Automated installations .....	2-39
RHEL Kickstart installations .....	2-41
RHEL Kickstart example (abbreviated) .....	2-44
SLES AutoYaST installations .....	2-47

AutoYaST example (abbreviated) . . . . .	2-49
Checkpoint (1 of 2) . . . . .	2-51
Checkpoint (2 of 2) . . . . .	2-53
Exercise: Installing Linux . . . . .	2-55
Unit summary . . . . .	2-57
<b>Unit 3. Linux documentation.....</b>	<b>3-1</b>
Unit objectives . . . . .	3-2
The man command . . . . .	3-4
man example (1 of 2) . . . . .	3-6
man example (2 of 2) . . . . .	3-8
man sections . . . . .	3-10
The info command . . . . .	3-12
info example . . . . .	3-15
The --help option . . . . .	3-17
HOWTO documents . . . . .	3-19
HOWTO example . . . . .	3-21
Other documentation . . . . .	3-23
Internet . . . . .	3-25
Unit review . . . . .	3-27
Checkpoint . . . . .	3-29
Exercise: Linux documentation . . . . .	3-31
Unit summary . . . . .	3-33
<b>Unit 4. Startup and shutdown.....</b>	<b>4-1</b>
Unit objectives . . . . .	4-2
Linux startup flow . . . . .	4-4
Basic Input/Output System . . . . .	4-6
BIOS Boot Loader location . . . . .	4-9
Unified Extensible Firmware Interface . . . . .	4-11
GUID Partition Table . . . . .	4-14
GRand Unified Bootloader . . . . .	4-17
GRUB configuration example . . . . .	4-20
Starting the kernel . . . . .	4-24
Initial RAM disk . . . . .	4-27
init . . . . .	4-30
/etc/inittab example . . . . .	4-33
Starting services: System V init style . . . . .	4-36
Starting services: Upstart . . . . .	4-39
Starting services (systemd) . . . . .	4-42
Configuring services per runlevel . . . . .	4-45
Starting and stopping services manually . . . . .	4-48
Booting Linux in single-user mode . . . . .	4-51
Shutting down a Linux system . . . . .	4-54
Checkpoint . . . . .	4-56
Exercise: Startup and shutdown . . . . .	4-58
Unit summary . . . . .	4-60

<b>Unit 5. System administration tools . . . . .</b>	<b>5-1</b>
Unit objectives . . . . .	5-2
System administration tools . . . . .	5-4
RHEL setup . . . . .	5-7
RHEL system-config-* . . . . .	5-10
SUSE YaST . . . . .	5-13
Webmin . . . . .	5-15
Webmin screenshot . . . . .	5-17
Other system administration tools . . . . .	5-19
Checkpoint . . . . .	5-21
Exercise: System administration tools . . . . .	5-23
Unit summary . . . . .	5-25
<b>Unit 6. Package management . . . . .</b>	<b>6-1</b>
Unit objectives . . . . .	6-2
Software management . . . . .	6-4
RPM Package Manager . . . . .	6-6
RPM philosophy . . . . .	6-9
RPM Files . . . . .	6-12
RPM installing, freshening, and upgrading . . . . .	6-15
RPM uninstalling . . . . .	6-20
RPM querying . . . . .	6-22
RPM verification . . . . .	6-25
RPM signatures . . . . .	6-28
RPM dependencies . . . . .	6-31
Solutions for RPM dependencies/updates . . . . .	6-34
Yum . . . . .	6-37
PackageKit . . . . .	6-39
Yum repositories . . . . .	6-41
Red Hat Network . . . . .	6-44
YaST software module . . . . .	6-47
Zypper . . . . .	6-49
Checkpoint . . . . .	6-51
Exercise: Package management . . . . .	6-53
Unit summary . . . . .	6-55
<b>Unit 7. X Window System and VNC . . . . .</b>	<b>7-1</b>
Unit objectives . . . . .	7-2
7.1. X Window System . . . . .	7-5
X Window System . . . . .	7-6
X client/server architecture . . . . .	7-9
Examples of X stations . . . . .	7-12
X servers in Linux . . . . .	7-15
X.org configuration . . . . .	7-17
Starting X . . . . .	7-20
Stopping X . . . . .	7-23
Session managers . . . . .	7-25
X applications networked . . . . .	7-27

Applications over TCP/IP .....	7-29
SSH tunneling of X .....	7-33
<b>7.2. Virtual Network Computing .....</b>	<b>7-37</b>
Virtual Network Computing .....	7-38
Configure KDE/GNOME desktop for VNC .....	7-41
Configure VNCserver .....	7-43
VNC client .....	7-47
Checkpoint .....	7-49
Exercise: X Window System and VNC .....	7-51
Unit summary .....	7-53
<b>Unit 8. User administration and security .....</b>	<b>8-1</b>
Unit objectives .....	8-2
<b>8.1. User administration .....</b>	<b>8-5</b>
Security concepts .....	8-6
User hierarchy .....	8-8
Groups .....	8-11
User Private Groups .....	8-14
Shadow password suite .....	8-16
Command line user tools .....	8-19
/etc/skel .....	8-21
Command line group tools (1 of 2) .....	8-23
Command line group tools (2 of 2) .....	8-25
Passwords .....	8-27
/etc/passwd .....	8-29
/etc/shadow .....	8-32
/etc/group and /etc/gshadow .....	8-35
/etc/issue and /etc/issue.net .....	8-37
Message of the day .....	8-39
<b>8.2. User-level security .....</b>	<b>8-41</b>
User-level security overview .....	8-42
Pluggable Authentication Modules .....	8-45
Authentication before PAM .....	8-47
Authentication with PAM .....	8-49
PAM configuration file example .....	8-52
Common PAM modules .....	8-57
Advanced authentication options .....	8-59
Principles of authorization .....	8-62
File permissions .....	8-65
Changing permissions .....	8-68
Access Control Lists (1 of 2) .....	8-70
Access Control Lists (2 of 2) .....	8-72
umask .....	8-74
Example: Creating a team directory .....	8-76
Root access .....	8-78
su command .....	8-80
sudo command .....	8-82
Security logs .....	8-85

Useful commands .....	8-87
Additional commands .....	8-89
Checkpoint (1 of 2) .....	8-91
Checkpoint (2 of 2) .....	8-93
Exercise: User administration and security .....	8-95
Unit summary .....	8-97
<b>Unit 9. Disk management, LVM, and RAID.....</b>	<b>9-1</b>
Unit objectives .....	9-2
9.1. Disk management .....	9-5
Internal hard disk type overview .....	9-6
Monitoring hard disk health .....	9-9
Disk partitioning principles .....	9-12
Partitioning tools .....	9-15
9.2. Logical volume management (LVM) .....	9-17
Logical volume management (1 of 2) .....	9-18
Logical volume management (2 of 2) .....	9-21
LVM implementation overview .....	9-23
Physical volume commands .....	9-25
Volume group commands .....	9-27
Logical volume commands .....	9-29
Striping logical volumes .....	9-31
Logical volume snapshots .....	9-33
Extending or reducing a volume group .....	9-36
Extending or reducing a logical volume .....	9-38
LVM backup and recovery .....	9-40
Additional LVM considerations .....	9-42
9.3. RAID .....	9-45
RAID .....	9-46
RAID levels (1 of 2) .....	9-49
RAID levels (2 of 2) .....	9-52
Linux RAID support .....	9-54
Linux software RAID implementation: mdadm .....	9-56
mdadm modes .....	9-59
mdadm implementation .....	9-62
Watching a running RAID .....	9-65
Spare disks .....	9-68
Additional RAID considerations .....	9-71
Checkpoint .....	9-74
Exercise: Disk management, LVM, and RAID .....	9-76
Unit summary .....	9-78
<b>Unit 10. File systems and file system quota .....</b>	<b>10-1</b>
Unit objectives .....	10-2
10.1. File systems.....	10-5
What is a file? .....	10-6
What is a file system? .....	10-9
The virtual file system .....	10-11

---

File systems supported . . . . .	10-14
Basic file system example: ext2 . . . . .	10-16
Superblock . . . . .	10-18
i-nodes . . . . .	10-20
Data blocks . . . . .	10-23
Ext2fs summary . . . . .	10-25
Other file system features . . . . .	10-28
Creating a file system . . . . .	10-31
Mounting a file system . . . . .	10-33
Mounting file systems at system startup . . . . .	10-35
Mount options . . . . .	10-38
Unmounting file systems . . . . .	10-41
Checking a file system . . . . .	10-43
ext2/3/4-specific information . . . . .	10-46
ReiserFS-specific information . . . . .	10-49
Comparing file systems . . . . .	10-52
SHMFS-specific information . . . . .	10-54
<b>10.2. File system quota . . . . .</b>	<b>10-57</b>
Quota concepts . . . . .	10-58
Quota implementation on Linux . . . . .	10-60
Enabling quota . . . . .	10-62
Configuring quota . . . . .	10-64
Quota information . . . . .	10-67
Checkpoint . . . . .	10-69
Exercise: File systems and file system quota . . . . .	10-71
Unit summary . . . . .	10-73
<b>Unit 11. Kernel services, kernel configuration and device management . . . . .</b>	<b>11-1</b>
Unit objectives . . . . .	11-2
<b>11.1. Kernel, kernel services and kernel configuration . . . . .</b>	<b>11-5</b>
Linux kernel . . . . .	11-6
Kernel components . . . . .	11-9
Loading modules . . . . .	11-12
Kernel services . . . . .	11-15
udev/devtmpfs . . . . .	11-17
procfs filesystem . . . . .	11-20
sysfs filesystem . . . . .	11-22
D-Bus . . . . .	11-24
Configuring the kernel . . . . .	11-27
Configuring the kernel at boot time . . . . .	11-29
Configuring modules at load time . . . . .	11-32
Configuring the running kernel . . . . .	11-35
Upgrading a kernel . . . . .	11-37
<b>11.2. Device management . . . . .</b>	<b>11-39</b>
UNIX device management . . . . .	11-40
Character devices . . . . .	11-43
Character device naming . . . . .	11-45
Virtual character devices . . . . .	11-47

Block devices . . . . .	11-50
Block device naming . . . . .	11-52
The loop device . . . . .	11-54
lspci command output . . . . .	11-57
lsusb command output . . . . .	11-59
udevadm info command output . . . . .	11-61
Checkpoint (1 of 2) . . . . .	11-63
Checkpoint (2 of 2) . . . . .	11-65
Exercise: Kernel services, kernel configuration and device management . . . . .	11-67
Unit summary . . . . .	11-69
<b>Unit 12. Memory management. . . . .</b>	<b>12-1</b>
Unit objectives . . . . .	12-2
Memory usage: Process view . . . . .	12-4
Virtual memory . . . . .	12-7
Memory metrics . . . . .	12-10
free command, /proc/meminfo . . . . .	12-13
ps command . . . . .	12-17
top command . . . . .	12-21
Procinfo-ng command . . . . .	12-24
vmstat command . . . . .	12-27
vmstat -s command . . . . .	12-31
Process memory: /proc/PID/status . . . . .	12-33
Process memory: /proc/PID/maps . . . . .	12-37
Creating paging space: Partition/LV/RAID . . . . .	12-40
Creating paging space: File . . . . .	12-43
/proc/sys/vm/swappiness . . . . .	12-45
Checkpoint . . . . .	12-48
Exercise: Memory management . . . . .	12-50
Unit summary . . . . .	12-52
<b>Unit 13. Virtualization. . . . .</b>	<b>13-1</b>
Unit objectives . . . . .	13-2
Virtualization overview . . . . .	13-4
Virtualization solutions for Linux . . . . .	13-7
Kernel virtualization module . . . . .	13-10
Quick Emulator . . . . .	13-13
QEMU example . . . . .	13-15
Xen: Overview . . . . .	13-17
Xen: Architecture . . . . .	13-20
Xen: Installation . . . . .	13-22
Booting the Xen Hypervisor and Dom0 kernel . . . . .	13-25
Xen DomU configuration file . . . . .	13-27
Xen configuration file: PV example . . . . .	13-30
Xen configuration file: HVM example . . . . .	13-33
Xen domain management (1 of 2) . . . . .	13-36
Xen domain management (2 of 2) . . . . .	13-38
QEMU-KVM . . . . .	13-40

Starting QEMU-KVM . . . . .	13-42
Libvirt . . . . .	13-44
Virtual Machine Manager screenshot . . . . .	13-47
Virtualization with KVM example . . . . .	13-49
Checkpoint . . . . .	13-51
Exercise: Virtualization . . . . .	13-53
Unit summary . . . . .	13-55
<b>Unit 14. Logging and troubleshooting . . . . .</b>	<b>14-1</b>
Unit objectives . . . . .	14-2
<b>14.1. Logging . . . . .</b>	<b>14-3</b>
Logging concepts . . . . .	14-4
Facilities and priorities . . . . .	14-7
Traditional syslogd configuration . . . . .	14-10
Syslog-NG . . . . .	14-13
/etc/syslog-ng/syslog-ng.conf example . . . . .	14-16
Rsyslog . . . . .	14-18
/etc/rsyslog.conf example . . . . .	14-20
logger command . . . . .	14-23
logrotate command . . . . .	14-25
Sample /etc/logrotate.conf . . . . .	14-28
Analyzing log files . . . . .	14-31
<b>14.2. Troubleshooting . . . . .</b>	<b>14-35</b>
Troubleshooting . . . . .	14-36
Identifying the problem: Part 1 . . . . .	14-39
Identifying the problem: Part 2 . . . . .	14-42
Core dumps . . . . .	14-45
Fixing the problem . . . . .	14-48
<b>14.3. Rescue Mode . . . . .</b>	<b>14-51</b>
What is Rescue Mode? . . . . .	14-52
Use available rescue tools (1 of 3) . . . . .	14-55
Use available rescue tools (2 of 3) . . . . .	14-57
Use available rescue tools (3 of 3) . . . . .	14-59
chroot command . . . . .	14-61
Booting Rescue Mode: RHEL . . . . .	14-64
Using Rescue Mode: RHEL . . . . .	14-67
Booting Rescue Mode: SUSE . . . . .	14-69
Using Rescue Mode: SUSE . . . . .	14-72
Repair installed system: SUSE . . . . .	14-74
Checkpoint (1 of 2) . . . . .	14-78
Checkpoint (2 of 2) . . . . .	14-80
Exercise: Logging and troubleshooting . . . . .	14-82
Unit summary . . . . .	14-84

<b>Appendix A. Checkpoint solutions.....</b>	<b>A-1</b>
<b>Appendix B. Setting up a network installation server .....</b>	<b>B-1</b>
<b>Appendix C. Networking .....</b>	<b>C-1</b>



# Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

AIX®	GPFS™	Notes®
Power®	PowerPC®	System i®
System p®	System x®	System z®
400®		

PostScript is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.



# Instructor course overview

This course will teach the students the differences between UNIX and Linux system administration. The course material and lab exercises support the following distributions of Linux:

- Red Hat Enterprise Linux v6.3 (RHEL6)
- SuSE Linux Enterprise Server v11 (SLES11)

## Course strategy

### Teaching strategy

Each classroom session uses a combination of facilitated lecture, discussions, group exercises, and demonstrations to convey the material.

### Introduce the material

Inform the students of the objectives of the unit and topic. Give them a brief scenario that will help them understand how the presented material will assist them in performing their jobs.

### Facilitate the learning experience

Involve the students in the learning process. Ask them questions and present classroom scenarios in which students use the available resources to solve situations involving process, procedure, or content on the job.

### Review the material

Review objectives at the conclusion of each unit to ensure that the students have a thorough understanding of the material.

Group exercises and labs are used to reinforce knowledge and skills that the students have learned in the previous classroom topics. The instructor serves as a mentor in checking results, answering questions, and providing constructive feedback and evaluation.

### Course evaluation

Evaluation measures the quality, effectiveness, and impact of the course. It enables students to answer the question, "Are the requirements and objectives of the course being met?"

For all classes, students will provide feedback on course quality by completing an end-of-course questionnaire.

### Measurement plan

There are no formal tests administered in the class.

### **Course materials**

- *Student Notebook*
- *Instructor Guide*
- PowerPoint visuals in PDF form to be displayed
- *Student Exercises*
- *Instructor Exercises Guide*

## **Summary of changes in this edition**

This course has been updated to use virtual machines in addition to the students' personal workstations. This allows the course to be taught in an online class, with the Linux lab systems hosted at an IBM or IBM partner location.

# Course description

## Linux Jumpstart for UNIX System Administrators

**Duration:** 4 days

### Purpose

This course will enable you to transfer your generic UNIX system administration skills to Red Hat Enterprise Linux (RHEL), SuSE Linux Enterprise Server (SLES), and Fedora Linux.

### Audience

Any UNIX System Administrator who is responsible for the implementation and system administration of Red Hat Enterprise Linux (RHEL) or SUSE Linux Enterprise Server (SLES).

### Prerequisites

Experience as a UNIX System Administrator and UNIX TCP/IP network administrator including TCP/IP configuration, routing, and networking software such as DNS and NFS.

### Objectives

After completing this course, you should be able to:

- Discuss the history of Linux
- Perform a network and automatic installation
- Use Linux online documentation
- Understand and manipulate Linux startup and shutdown flow
- Identify system administration tools
- Identify and use utilities to install and update software packages
- Identify and use common Linux block devices
- Describe, create, and use Linux file systems
- Create and manage Linux users and groups
- Learn common troubleshooting methods
- Describe and use memory management utilities
- Describe and configure virtualization



# Agenda

## Day 1

- (00:15) Welcome
- (00:30) Unit 1: Introduction to Linux
- (00:45) Unit 2: Installing Linux
- (01:00) Exercise 2: Installing Linux
- (00:15) Unit 3: Linux documentation
- (00:15) Exercise 3: Linux documentation
- (00:45) Unit 4: Startup and shutdown
- (00:45) Exercise 4: Startup and shutdown
- (00:30) Unit 5: System administration tools
- (00:30) Exercise 5: System administration tools

## Day 2

- (00:45) Unit 6: Package management
- (00:45) Exercise 6: Package management
- (00:45) Unit 7: X Window System and VNC
- (00:45) Exercise 7: X Window System and VNC
- (00:45) Unit 8: User administration and security
- (00:30) Exercise 8: User administration security
- (01:00) Unit 9: Disk management, LVM, and RAID

## Day 3

- (01:00) Exercise 9: Disk management, LVM, and RAID
- (00:45) Unit 10: File systems and file system quota
- (00:45) Exercise 10: File systems and file system quota
- (01:00) Unit 11: Kernel services, kernel configuration and device management
- (01:00) Exercise 11: Kernel services, kernel configuration and device management
- (00:30) Unit 12: Memory management
- (00:30) Exercise 12: Memory management

## Day 4

- (01:00) Unit 13: Virtualization
- (01:30) Exercise 13: Virtualization
- (00:45) Unit 14: Logging and troubleshooting
- (01:30) Exercise 14: Logging and troubleshooting



# Unit 1. Introduction to Linux

## Estimated time

00:30

## What this unit is about

This unit covers the history of Linux and names some important people involved in the history of Linux. This unit also discusses the GNU general public license.

## What you should be able to do

After completing this unit, you should be able to:

- Discuss the history of Linux
- Name some important people in the history of Linux
- Discuss the GNU general public license

## How you will check your progress

- Checkpoint questions

## Unit objectives

---

After completing this unit, you should be able to:

- Discuss the history of Linux
- Name some important people in the history of Linux
- Discuss the GNU general public license

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 1-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose** — To introduce the content of this unit.

**Details** —

**Additional information** —

**Transition statement** —

## A short history of Linux (1 of 2)

- 1984: Richard Stallman starts GNU project
  - GNU's not UNIX
  - <http://www.gnu.org>
- Purpose: Free UNIX
  - "Free as in free speech, not free beer"
- First step: Re-implementation of UNIX utilities
  - C compiler, C library
  - emacs
  - bash
- To fund the GNU project, the Free Software Foundation founded
  - <http://www.fsf.org>



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 1-2. A short history of Linux (1 of 2)

LX158.0

### Notes:

The history of Linux starts properly in 1984. In that year, a system administrator working at Massachusetts Institute of Technology (MIT), Richard Stallman, received a new version of the UNIX flavor they were using. But in contrast to previous versions, this time they did not receive the source of the operating system with it, and could not obtain the source separately without signing a Non-Disclosure Agreement (NDA). Richard Stallman was therefore not able to implement a certain additional feature into the operating system, which his users had come to like.

Richard Stallman became so upset with these developments in general that he vowed to write a new UNIX-like operating system from scratch. That new operating system was supposed to be free (as in free speech): Everybody would have the right to use and adapt the software for its own use, and to distribute the software to others. (More about this later.) This project was called GNU, which stands for GNU's not UNIX.

To fund the GNU project and to advocate the use of free software in general, the Free Software Foundation (FSF) was founded.

The first steps taken by the GNU project was to re-implement various essential utilities in a UNIX operating system. Although hundreds of little tools were written, four tools stand out:

- The GNU C Compiler (GCC) was essential for compiling all software, including the kernel and the C compiler itself.
- The GNU C Library (GLIBC) implements a large set of standardized system calls.
- Emacs is a full-featured, world-class editor which can be extended into a sort of application development environment.
- Bash (Bourne again shell) is a command interpreter and programming environment. Having a shell is essential on a UNIX system, because the shell interprets and executes the commands you type.

Later on, the GNU project also started development on a UNIX-like kernel<sup>1</sup>, called Hurd. This kernel has never been important for Linux, however, because it was released for the first time at the end of the 1990s, when Linux was already thriving.

<sup>1</sup> The kernel of an operating system is the program that runs 24 hours a day and takes care of scheduling, device handling, memory management and so forth.

### **Instructor notes:**

**Purpose** — Discuss the GNU project in short.

#### **Details —**

**Additional information** — For more information about the history of UNIX, up to about 1994, read Peter H. Salus' book A Quarter Century of UNIX (Addison-Wesley Publishing Company, Inc. ISBN 0-201-54777-5).

**Transition statement** — The GNU project went merrily along, and the next milestone came in 1991.

## A short history of Linux (2 of 2)

- 1991: Linus Torvalds writes first version of Linux kernel
  - Initially, a research project about the 386 protected mode
  - Linus' UNIX > Linux
  - Combined with the GNU and other tools to form a complete UNIX system
- 1992: First distributions emerge
  - Linux kernel
  - GNU and other tools
  - Installation procedure
- The rest is history



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 1-3. A short history of Linux (2 of 2)

LX158.0

### Notes:

In 1991, a student at the University at Helsinki, Linus Torvalds, started a small research project into the workings of the Intel 80386 processor, which by then was state-of-the-art. He was interested in exploring a new feature, which, up to then, was not present in any Intel processor, namely a memory management unit (MMU). This MMU offered hardware support for running multiple processes simultaneously, each in its own memory segment. With such an MMU, processes cannot access memory areas owned by other processes, and this effectively means that if one process crashes, it cannot take the whole system down with it.

The operating systems available for the 386 (Windows for Workgroups and Minix) all did not use this feature and were therefore very prone to crashing. ("Who is General Failure and why is he reading my hard disk?")

Linus started out writing three small programs:

- A small program that continuously printed the letter A on the screen
- A small program that continuously printed the letter B on the screen

- A slightly larger program that switched the processor to protected mode and scheduled the other two programs to take turns

When Linus finally managed to see the output of both programs on his screen, in turn (ABABABAB...), he knew he had the beginnings of a kernel of a multitasking operating system. Linus continued to improve and refine the kernel, and at the end of 1991, he was able to run the GNU C compiler and the Bash shell under his kernel, which by then was dubbed Linux, for Linus' UNIX. Linus then decided to upload this to the Internet (which by then was still largely a university network) for others to use:

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)  
Newsgroups: comp.os.minix  
Subject: What would you like to see most in minix?  
Summary: small poll for my new operating system  
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>  
Date: 25 Aug 91 20:57:08 GMT  
Organization: University of Helsinki  
Hello everybody out there using minix -  
I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since April, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things). I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-(  
Linus (torvalds@kruuna.helsinki.fi)  
PS. Yes - it's free of any minix code, and it has a multi-threaded fs.  
It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Other people on the Internet started picking up this software, using it, and refining it. The patches were sent to Linus, who incorporated this into the main kernel stream. Starting to use Linux was a major undertaking, however. The Linux kernel, the C compiler, the shell, and all the other tools you need to make a complete operating system were all distributed in source code form. Before you can make use of them, you need to compile them, which requires a C compiler, which itself also needs to be compiled first. To break through this vicious circle, people started creating distributions which contain precompiled versions of the kernel, C compiler, various tools, and some sort of installation program. All this is stored in a convenient format for installation (originally floppy disk images, but today CD-ROM and DVD, or ISO, images are prevalent). The rest, as they say, is history. Because of the close association between Linux (the kernel) and the GNU project, an operating system distribution including the Linux kernel and GNU libraries and utilities is often called "GNU/Linux". This is Richard Stallman's strong preference.

**Instructor notes:**

**Purpose** — Discuss a short history of Linux.

**Details** — Minix is a UNIX variant for the Intel 286, which was written by Andrew Tanenbaum, a professor of Computer Science at the Free University in Amsterdam. He had been using the BSD UNIX source code to illustrate to his students how an operating system was written. When BSD grew and the source code license changed, however, this was no longer practical.

Andrew Tanenbaum therefore decided to write his own UNIX-like operating system, which was to be as small as possible, so that it could be used for teaching. (Minix = Mini Unix) The whole source code of the Minix kernel only is about 12.000 lines of code. Because of this, Minix is not really feature-rich.

Andrew Tanenbaum and Linus Torvalds have been in contact with each other regarding extending Minix to use the 386 memory management, more hardware support, other internal structures and so forth, but in the end, they decided not to work together. (Linux originally actually contained code borrowed from Minix, but this code is all gone now.)

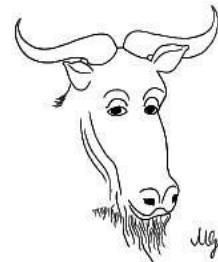
The discussions between Linus Torvalds and Andrew Tanenbaum are cataloged on the Web, for instance at [http://www.dina.dk/~abraham/Linus\\_vs\\_Tanenbaum.html](http://www.dina.dk/~abraham/Linus_vs_Tanenbaum.html) or <http://oreilly.com/catalog/opensources/book/appa.html>.

**Additional information —**

**Transition statement** — Let's see what's so special about Linux (or GNU/Linux).

## What's so special about Linux (GNU/Linux)?

- Most software (including the Linux kernel) is GPL'ed (GNU General Public License)
  - <http://www.gnu.org/copyleft/gpl.html>
- Linux is called copyleft (instead of copyright)
  - You can copy the software.
  - You get the source code.
  - You can alter the source code and recompile it.
  - You can distribute the altered source and binaries.
  - You can charge money for all this.
- You cannot change the license
  - All your customers have the same rights as you.
  - You really cannot make money from selling the software alone.
- Other Open Source licenses (for example, BSD) are also used



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 1-4. What's so special about Linux (GNU/Linux)?

LX158.0

### Notes:

To understand what is so special about Linux, it is necessary to quickly look at international copyright laws. The principle of copyright is very simple: When an author creates a unique piece of work, such as a computer program, then he is the owner of all rights to that piece of work. He may decide what others can and cannot do with it.

What others may do with that piece of work is usually written down in a License Statement, a contract between the creator and the user, which describes the rights that the user has. These rights may be granted for free, but in most cases the user has to pay for them.

A typical license in the world of computer software entitles the user to run the binary program on the number of machines that the license was purchased for. It is not allowed to make more copies of the software than needed for running it, and one extra backup copy. Furthermore, the user cannot claim any rights to the source code and is not allowed to disassemble the binary code to learn and/or alter its inner workings. In short, a typical copyright statement does not give you the right to copy.

In contrast, the GNU General Public License or GPL for short, turns this around. The aim of the GPL is to keep all software free so that everybody can adapt the software to their own

---

needs, without being dependent on the goodwill of the author. This means that any piece of software that has been placed under the GPL by the original author gives the user the following rights:

- The user can copy the (binary) software as often as the user wishes.
- The user has the right to obtain the source code.
- The user has the right to alter the source code and recompile the source code into binary form.
- The user can distribute the sources and the binaries.
- The user can charge money for all of this.

Basically, the only restriction that the GPL imposes on all users is that the license statement may not be changed. This means that all your customers have the same rights to the software as you do. And as a practical aside, that means that in general, it is impossible to make any money from selling the software (apart from a nominal fee for media and distribution).

The GPL is the most-often used license statement for open source projects, but other licenses, such as the BSD license, are also being used. BSD-style licenses have very few restrictions.

***Instructor notes:***

**Purpose** — Discuss the GPL.

**Details** —

**Additional information** —

**Transition statement** — Let's see what the effects of this license model are.

## Effects of the license model

- Everybody has access to the source
  - Volunteer software development on the Internet with centralized coordination.
  - Linus Torvalds coordinates new core kernel development.
  - Others coordinate other pieces of the kernel and OS.
- Peer reviews are possible
  - Security
  - Performance
  - Reliability: “Given enough eyeballs, all bugs are shallow.”
- License cannot change
  - Your changes (and name) will stay in forever.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 1-5. Effects of the license model

LX158.0

### Notes:

The effects of this license model are far reaching.

The first effect is that, since everybody has access to the source code, everybody interested can improve the code, or add new features. This means that software development is potentially very rapid, with possibly hundreds of developers working on the same piece of code. People in the Linux community understand the inherent risk of a code fork with a development model like this, and a lot of effort is spent in coordinating the work of various developers. (A *code fork* is when one project becomes two independent projects, likely with different goals, so the source code will begin to diverge from a common base.) This usually comes down to two things.

A volunteer or group of volunteers are responsible for coordination of the development. Linus Torvalds, for instance, hardly writes any code anymore, but spends most of his time coordinating others who write code for the kernel. Other people coordinate the development on other programs.

Some sort of automated support for distinguishing and integrating contributions of developers. Common source code control systems include git (currently used for the Linux kernel), cvs, subversion.

Another effect of having the source available is that peer reviews are possible. It is easy for people to look through the code and identify performance or security problems. Eric Raymond, in his essay “The Cathedral and the Bazaar”, coined what he called “Linus’s Law”, more formally stated “Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix will be obvious to someone.”

A third effect of the license model is that if you make significant changes, or add a feature, then that feature is owned by you, and not by the original author of the software. This means that your name (as part of the copyright statement for that feature) stays in forever. This is usually a great motivation factor for people.

***Instructor notes:***

**Purpose** — Discuss the effects of the license model.

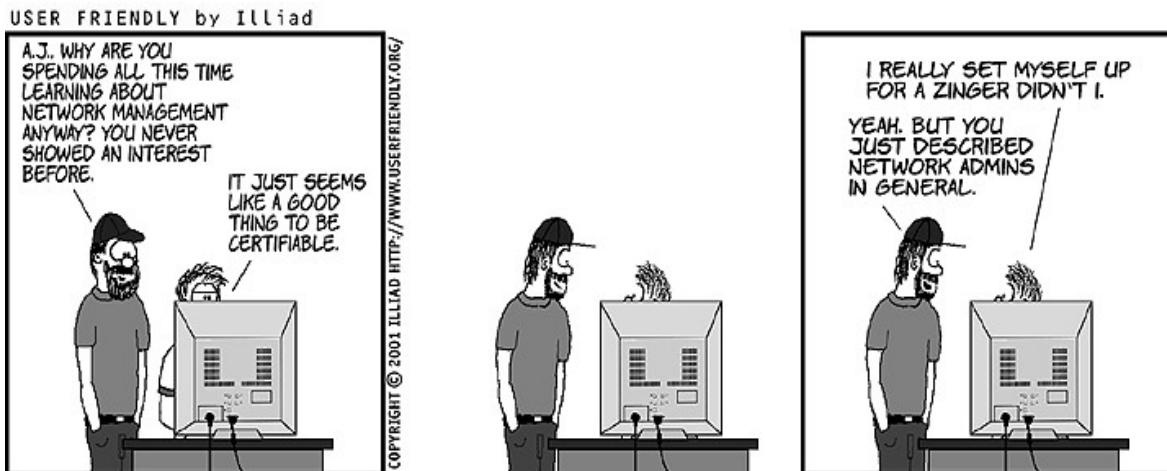
**Details** —

**Additional information** —

**Transition statement** — Linux has become a way of life for a lot of people. Let's see some examples of that.

# Linux has become a way of life

- Culture
- Celebrities
  - Linus Torvalds
  - Richard Stallman
  - Eric Raymond
- Humor
  - User friendly
  - XKCD
- Mascot
  - Tux



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 1-6. Linux has become a way of life

LX158.0

## Notes:

For a large number of people, Linux is not just another operating system, but it has become a way of life for them. It is something they believe in, and they want to express that belief. Devotees often associate Linux with “freedom”, as expressed with the GPL license, as well as freedom from the “tyranny of a monopoly”, usually with reference to Microsoft. Linux is also associated with the following:

- **Capability:** Any newly desired feature can be realized with a simple matter of programming.
- **Flexibility:** You can do whatever you want since you have all of the source code.
- **Performance:** Since you can recompile everything, you can optimize everything to suit *your* specific environment.

As part of the Linux identity, early in 1996, people felt the need for a logo for Linux. After having discussed various designs over the Internet, Linus stepped in and said that he would like to see a penguin as logo for Linux. Simply because he liked penguins. Several authors then started drawing penguins to use as the Linux logo, and by popular acclaim the

---

penguin drawn by Larry Ewing was chosen as the official logo; however, the penguin should be seen more as a mascot than as a logo, and people are free to create their own penguin-based logos (or adapt Larry Ewing's picture) for their own purposes.

The penguin was eventually named Tux, which officially stands for [Torvalds [U]NI[X]. A real Tux exists as well: A number of UK Linux fans, lead by Alan Cox, and the Linux World magazine have sponsored a live penguin at the Bristol Zoo as a birthday present for Linus.

For a complete overview of the history of Tux, including links to other sites, see <http://www.sjbaker.org/tux>.

**Instructor notes:**

**Purpose** — Show that for many people Linux has become a way of life.

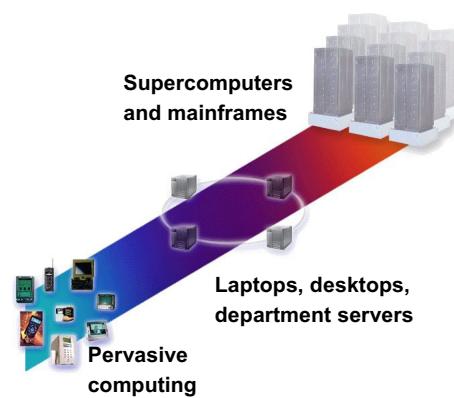
**Details** —

**Additional information** —

**Transition statement** — Even if it is not your way of life, it is still a fairly good operating system. Let's just see what Linux is today.

## Linux today

- Linux covers the whole spectrum of computing.
  - Embedded devices, smartphones
  - Laptops
  - Desktop systems
  - Development systems
  - Small and large servers
  - Mega clusters and supercomputers
- Linux is used throughout the world and in space.
- Linux is used by home users, by most of the largest companies in the world, and by many governments and institutions.



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 1-7. Linux today

LX158.0

### Notes:

The smallest implementations of Linux can be found in embedded devices: the microchips that control your VCR, microwave, router, and so forth. The Tivo (<http://www.tivo.com>), for instance, runs on Linux. And so does every Android-based smartphone.

In the regular IT-world, Linux runs on laptops, desktops and servers. And it is even used to run most of the largest supercomputers in the world: As of November 2011, Linux ran on 457 of the Top 500 supercomputers in the world (source: [www.top500.org](http://www.top500.org)).

Linux is used by home users as well most of the largest companies in the world, numerous governments and educational institutions. Linux powers most of the web servers on the Internet.

Linux is used throughout the world, and in space: Various experiments on board of the International Space Station are controlled by systems running Linux.

Linux powered the Deep Blue supercomputer that beat reigning world chess champion Garry Kasparov. And recently, IBM created a Power-based supercomputer called

“Watson”, which defeated the two most successful contestants of the Jeopardy! game show.<sup>2</sup>

---

<sup>2</sup> <http://www.ibm.com/innovation/us/watson/index.html>

***Instructor notes:***

**Purpose** — Give some indication of where and how Linux is used today.

**Details** —

**Additional information** —

**Transition statement** — Let's look at the hardware Linux runs on.

# Linux hardware support



- Also supported:
  - ARM, Itanium, Sparc, PA-Risc, 68000 and so forth

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 1-8. Linux hardware support

LX158.0

## Notes:

Almost all operating systems that are currently available are written for one specific hardware architecture. Originally, that was the case for Linux as well. Linus Torvalds only had an Intel-based PC. This changed in 1994 when Digital Equipment Corporation (DEC, later bought by Compaq, which subsequently merged with HP) gave a DEC Alpha to Linus Torvalds, no questions asked. A few months later, Linux ran on the DEC Alpha.

People took on the effort of porting Linux to other architectures and fed back the architecture-dependent code to Linus. Later, companies with a commercial interest in running Linux on their hardware architecture started contributing as well. This means that Linux now runs natively on a very large number of platforms, including the following:

```
$ ls /usr/src/linux/arch
alpha  blackfin  h8300  m68k        mips      PowerPC  sh      x86
arm    cris       ia64   m68knommu mn10300  s390     sparc   xtensa
avr32  frv        m32r   microblaze parisc    score    um
```

---

The list in the visual is just the list of architectures that are supported in the mainline kernel. Patches exist for other architectures as well, but have not all been made part of the mainline kernel.

Within an IBM context, the most important architectures are x86/ia64 (Intel, both in 32 and 64 bit, for IBM System x), PowerPC (for IBM Power-based systems, such as IBM System p and IBM System i) and s390 (for IBM System z mainframes). All these architectures are natively supported, meaning that the same source code base is used on all architectures. Furthermore, all are tier-1 architectures for Red Hat and SUSE, meaning that the availability of new distributions, service packs, patches and so forth happens for all these architectures at (virtually) the same time.

**Instructor notes:**

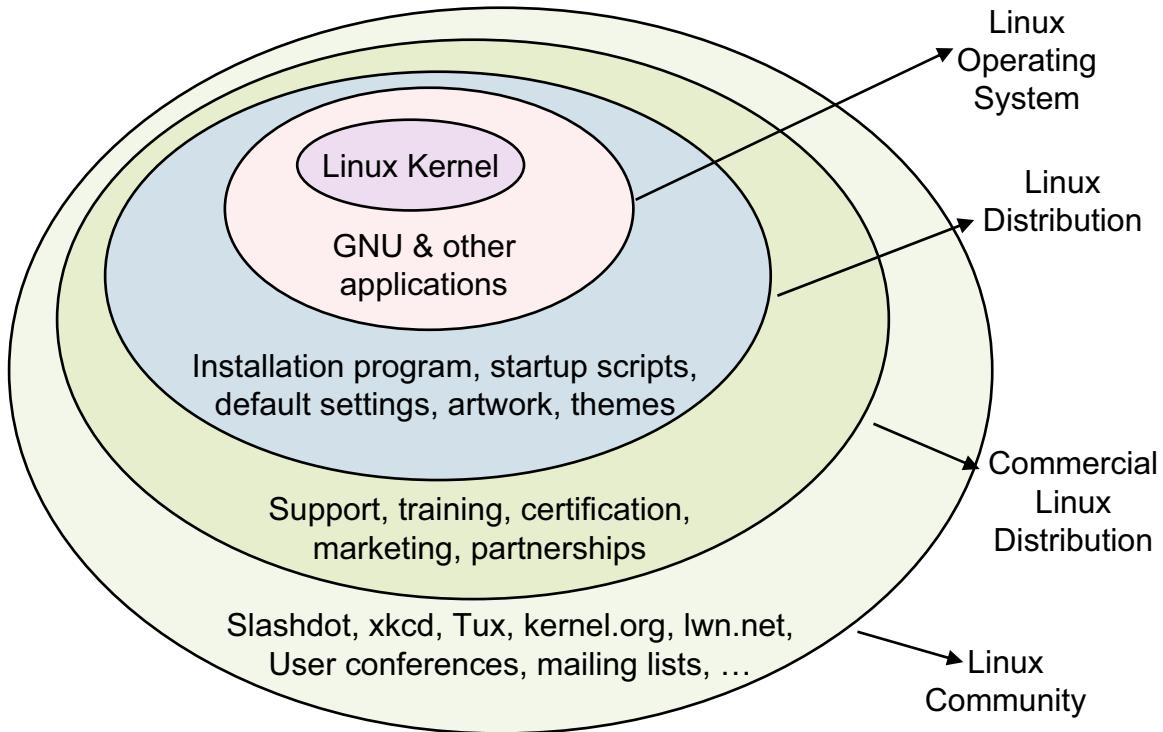
**Purpose** — Show the various hardware architectures that are supported by Linux.

**Details** —

**Additional information** —

**Transition statement** — That's it. I hope you got the idea about Linux. We've got some checkpoint questions to do, and then we're going to run Linux ourselves in the next unit.

# Looking at Linux



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 1-9. Looking at Linux

LX158.0

## Notes:

Linux, in its most narrow definition, refers to the kernel, the heart of the operating system that was originally written by Linus Torvalds. But today, the word "Linux" is used in much broader definitions too.

Any operating system that uses the Linux kernel, augmented by GNU and other tools, can be thought of as a "Linux Operating System".

Recompiling the Linux kernel and the hundreds of tools that are required to run a Linux Operating System can be a very daunting task. That's why "Linux Distributions" appeared quickly after the first releases of the Linux kernel. A distribution generally consists of an installation program to install the precompiled Linux kernel and other tools quickly and easily on your system. It will also include things like startup scripts, default settings, management tools, package installation tools, artwork, themes and so forth.

Creating a Linux Distribution is a large task. Some distributions, most notably Debian, manage to enlist the help of sufficient volunteers so that their distribution can be downloaded and supported for free. Most distributions today however have a model where the users of that distribution need to pay for support, training, certification, marketing,

partnerships and so forth. And this income stream is then used to pay for the further development of the distribution itself. Examples of companies that have created a commercial enterprise surrounding their distribution are Red Hat, SUSE and Ubuntu.

The last way of looking at Linux is by looking at the Linux Community. This obviously includes the code itself, and the people developing and supporting that code. But it includes more. There are various websites that are very Linux-oriented, there are user conferences, mailing lists, support forums and so forth. And of course there's Tux, the mascot of Linux.

## Unit review

- The Linux kernel, combined with the GNU and other tools, forms a complete UNIX-like operating system.
- A distribution pulls together versions of the Linux kernel, libraries, and tools, tests them as a cohesive operating system, and adds an installation procedure and a convenient format for distribution.
- Most software in a Linux distribution is licensed under Open Source licenses such as the GNU GPL.
- Linux is developed as a world-wide collaboration of corporations and volunteers.
- Linux has been ported to more than 20 hardware architectures including virtually all PC hardware.
- Linux is used in a variety of small and large applications, homes, schools, small and large businesses, and governments.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 1-10. Unit review

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Checkpoint

1. True or false: Linus Torvalds wrote the Linux operating system all by himself.
2. Which of the following statements is not true about software licensed under the GNU GPL?
  - a. You have the right to obtain and review the source code.
  - b. You cannot charge any money for the software.
  - c. You cannot change the license statement.
  - d. You can modify the source code and subsequently recompile it.
3. Who is the mascot of Linux?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 1-11. Checkpoint

LX158.0

### Notes:

**Instructor notes:**

Purpose —

Details —

## Checkpoint solutions

---

1. True or false: Linus Torvalds wrote the Linux operating system all by himself.

The answer is false.

2. Which of the following statements is not true about software licensed under the GNU GPL?

- a. You have the right to obtain and review the source code.
- b. You cannot charge any money for the software.
- c. You cannot change the license statement.
- d. You can modify the source code and subsequently recompile it.

The answer is you cannot charge any money for the software.

3. Who is the mascot of Linux?

The answer is Tux.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —**

**Transition statement —**

## Unit summary

Having completed this unit, you should be able to:

- Discuss the history of Linux
- Name some important people in the history of Linux
- Discuss the GNU general public license

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 1-12. Unit summary

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

# Unit 2. Installing Linux

## Estimated time

00:45

## What this unit is about

This unit describes preparing a system for installation, installing Linux from CD-ROM, performing a network installation, setting up a RHEL or SLES network install server, setting up and performing a RHEL Kickstart install, and setting up and performing a SLES AutoYaST install.

## What you should be able to do

After completing this unit, you should be able to:

- Prepare a system for installation
- Install Linux from CD-ROM
- Perform a network installation
- Set up and perform a RHEL Kickstart install
- Set up and perform a SLES AutoYaST install

## How you will check your progress

- Checkpoint questions
- Lab exercises

## Unit objectives

---

After completing this unit, you should be able to:

- Prepare a system for installation
- Install Linux from CD-ROM
- Perform a network installation
- Set up and perform a RHEL Kickstart install
- Set up and perform a SLES AutoYaST install

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**



## 2.1. Manual installation

### Instructor topic introduction

**What students will do —** Learn how to do a manual installation.

**How students will do it —**

**What students will learn —**

**How this will help students on their job —**

# Preparing a system for installation

---

- Know your hardware
  - CPU, memory, keyboard, mouse
  - Hard disks, CD-ROM players
  - Graphical adapters, monitor capabilities
  - Network adapters, IP addresses
  - Printers
- Is all your hardware supported?
  - Linux Hardware-HOWTO
  - Distributor's hardware compatibility list
  - Hardware manufacturer
  - If unsure, just try it!
- Determine where Linux will be installed

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 2-2. Preparing a system for installation

LX158.0

## Notes:

Before you install a Linux system, there are some things you should do. One of the most important steps is knowing what hardware you have, and all the characteristics and configuration options of that hardware.

Furthermore, you need to verify that all your hardware is supported by Linux. Since not all hardware manufacturers make the specifications of their hardware public, some hardware is not supported, or not supported in full. For a detailed list of hardware supported by Linux, refer to the Hardware-HOWTO at <http://www.tldp.org>. Also, your distributor might have several restrictions on the hardware that their distribution supports. You might also be able to obtain information from the hardware manufacturer itself. If you are unsure whether your hardware is supported, then just go ahead and try it.

Another important step is making space for Linux partitions. Linux must have its own physical or virtual disk partitions available for install.

**Instructor notes:**

**Purpose** — Discuss the preparation of a system.

**Details —**

**Additional information** — Some distributions allow you to install Linux in existing Windows partitions. In this case, two large files are created on the Windows partition. One is used as swap space, and the other as root partition for Linux. Linux is then started from within Windows with a special command. This allows you to test drive Linux without repartitioning Linux. It is, however, not recommended for production use.

For information about this, see your distributions installation guide.

**Transition statement** — Let's determine how to start a Linux installation.

## Installing Linux

---

- Boot system from bootable media
  - CD/DVD
  - USB
  - PXE
- After booting, install from:
  - CD/DVD
  - Local hard disk
  - Network

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-3. Installing Linux

LX158.0

### Notes:

Installing Linux starts with booting a very tiny Linux system from some sort of bootable media. Most systems can boot from the installation CD-ROM/DVD directly (depending on the Firmware version and settings), but older systems might need to boot from a boot diskette.

In addition to this, certain distributions require additional diskettes/CDs to be available. Modern Linux distributions (RHEL, SLES, Fedora) no longer include diskette boot images, but do provide images for USB thumb-drives.

***Instructor notes:***

**Purpose** — Discuss the start of the installation process.

**Details** —

**Additional information** —

**Transition statement** — Let's look at the next steps of the installation process.

## Network installations

- Installations where packages to install are downloaded from the network
- Network protocols supported depends on distribution
  - Network File System (NFS)
  - File Transfer Protocol (FTP)
  - Hypertext Transfer Protocol (HTTP)
- Requires a network install server
- Usually requires special network-enabled boot media
  - Preboot Execution Environment (PXE) boot requires no media



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-4. Network installations

LX158.0

### Notes:

Most Linux systems are installed from the distribution CD-ROMs (or DVDs). This is a convenient method if you only need to install one or a few systems but quickly becomes tedious if you need to install ten or more systems, especially if each system has to be installed with the same settings.

More advanced installation methods exist which are convenient for these situations, and in all but a few cases, this comes down to network installations, where the Red Hat Package Manager software packages (RPMs) to be installed are downloaded from the network.

### Network protocols

Various network protocols exist to retrieve the installation RPMs, and the protocols that are supported depend on your distribution. Support might be included for Network File System (NFS), File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP), and Server Message Block (SMB).

An obvious requirement for a network-based install is that somewhere on the network you need to configure a network install server, which holds all the RPMs for your distributions.

---

Finally, you will need some network-enabled boot media. This can be the first CD (or DVD) of your regular installation or a minimal install CD or DVD ISO image. If your system supports the Preboot Execution Environment (PXE), you can boot and install your distribution over the network without the need for physical boot media.

For PPC distributions, the distribution media includes a network bootable install kernel that can be used to install the machine over the network.

***Instructor notes:***

**Purpose** — Introduce network installations.

**Details** — We are still introducing the general concept of installing through a network. Use this time to discuss why a network would be more convenient than installing from CD-ROM.

**Additional information —**

**Transition statement** — Let's look at that network install server.

## Installation steps

- All installation programs need to perform essentially the same steps:
  - Choose language, keyboard type, mouse type
  - Create partitions/logical volumes
  - Set up a boot loader
  - Configure network
  - Configure users and authentication
  - Select package groups
  - Configure X
  - Install packages
  - Register
- The order of steps might vary from distribution to distribution
- Other steps might also be included:
  - For example, firewall, printers, and sound

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-5. Installation steps

LX158.0

### Notes:

After the system is booted, the installation program takes over, and asks you a number of questions regarding the Linux configuration. It then installs Linux and configures it according to the options you specified.

Obviously, every distribution has its own installation program; so the exact order in which questions are being asked is different from distribution to distribution. However, every distribution basically needs to do the same things in its installation process; so if you look beyond the order of the various menu screens, the installation programs do not differ that much from each other.

## **Instructor notes:**

**Purpose** — Discuss the installation steps.

**Details** —

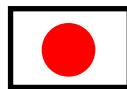
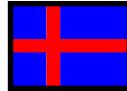
**Additional information** —

**Transition statement** — Let's view each step in a little more detail.

**Tip:** Experience has shown that it is a good idea at this point not to continue the lecture from the visuals, but to perform a live install and cover the theory while the installation progresses. If needed, you can point the students to the correct page in the manual.

## Select language, keyboard, and mouse

- Select the language to be used during installation process.
  - Different distributions support different languages.
- Select the keyboard layout.
  - Different countries use different keyboard layouts.
  - Dead (compose) keys allow you to input accented or special characters, such as é, ç, ß, and so forth.
- Select your mouse.
  - A mouse can be connected using a PS/2, USB, or serial connector.
  - If your mouse has only two buttons, you can emulate the third (middle) button by clicking both buttons simultaneously.



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-6. Select language, keyboard, and mouse

LX158.0

### **Notes:**

One of the first things the installation program needs to do is to determine the language to be used during installation, and to determine the keyboard layout and mouse type.

***Instructor notes:***

**Purpose** — Discuss the language, keyboard, and mouse configuration screens.

**Details** —

**Additional information** —

**Transition statement** — Let's look at the next step.

## Install class

- Some distributions have installation classes for typical users.

- Desktop



- Laptop



- Server



- Developer

- A custom class allows you to make all decisions yourself.
  - Packages to be installed
  - Various configuration options

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-7. Install class

LX158.0

### Notes:

Some distributions allow you to select an installation class. These classes allow you to quickly install a typical system. Among other things, a class determines the packages that are installed and various configuration options. If a distribution uses these classes, then it always supports a custom class, too, which allows you to make all decisions yourself.

Note that some distributions also make assumptions regarding partitioning, depending on the class chosen. As an example, a Red Hat Workstation install removes all existing Linux partitions, and a Red Hat Server install removes ALL existing partitions, including any non-Linux partitions. If you choose to use an installation class, make sure to read the documentation.

Most distributions also support an Upgrade class, which does not make any configuration changes but upgrades all currently installed software to the latest level.

***Instructor notes:***

**Purpose** — Discuss installation classes.

**Details** —

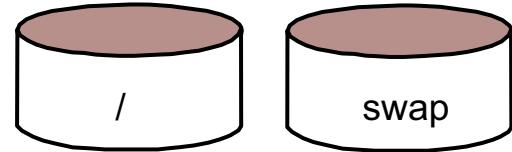
**Additional information** —

**Transition statement** — The next step is usually disk partitioning.

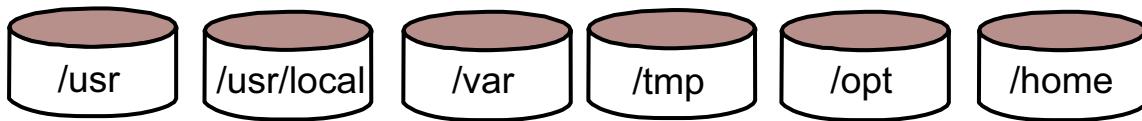
## Disk partitioning

- Linux installation requires you to create Linux partitions

- At a minimum, create / and swap:



- You might need or want to create other partitions
  - For example, /usr, /usr/local, /var, /tmp, /opt, and /home



- Some distributions will use Logical Volume Management (LVM) by default

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-8. Disk partitioning

LX158.0

### Notes:

Almost every Linux distribution allows you to partition your disks during the installation process. Most distributions use **fdisk** for this, but some distributions include their own partitioning tool. No matter what tool you use, you need to create at least two partitions:

- The first partition to create is your root partition. This partition holds the file system, which in turn holds all your data. Sizing depends on whether other file systems are going to be created. If this is the only file system, it should be a minimum of 5 Gb in size. Note that it is possible to create more partitions (/usr, /tmp and so forth). In that case, your root partition does not have to be this large.
- The second partition does not hold a file system but is used as swap space. This is virtual memory, which is used when your system exhausts its real memory. The opinions on the size of this swap space vary, but it's usually best to take the amount of real memory on your system as swap space.<sup>1</sup>

<sup>1</sup> When your system starts using the swap space, it generally means that you do not have enough real memory to run all your processes in. This leads to a huge performance loss, since hard disk accesses are far slower than memory accesses. Memory today is so cheap that you should size your system so that you do not need swap space, except in a rare situation.

Note that the swap space needs to have another partition type (type 82), and does not get a mountpoint.

Even though it is not strictly needed on most systems, it is a good idea always to create a /boot partition (that is short for saying “a partition which holds a file system that is mounted at the /boot mountpoint”) of about 100 Megabytes<sup>1</sup>. This partition holds everything that is needed by the Linux boot process. The most important program here is the Linux kernel itself, but you also need to have some components of LILO or GRUB stored here. The reason to store these components on a separate partition is complex and outside the scope of this course.

Depending on what you are going to do with your system, you might also need to create other partitions for /usr, /usr/local, /var, /tmp, /opt, /home and so forth. When this is needed and how these are created is outside the scope of this course; they're not needed for a Linux workstation anyway.

When partitioning, make sure you don't delete any existing Windows partitions, and make sure that you format only all newly created Linux partitions.

---

<sup>1</sup> Red Hat requires your /boot partition to be at least 100 Megabytes, so that you can install several kernel images.

**Instructor notes:**

**Purpose** — Discuss disk partitioning.

**Details —**

**Additional information** — A /boot partition is not strictly necessary on most systems, but it is a good idea (and thus, a good habit) to create it nevertheless. The reason for this is rather complex.

When the system boots, at some point LILO or GRUB is being run. This program is tasked to load the Linux kernel and possibly some other files into memory, and then to start the Linux kernel. Now, LILO is a minimalist program and has no concept of the structure of a file system, and neither does it know how to handle the low-level complexities of IDE- and SCSI disks. GRUB is only marginally more intelligent, but suffers from the same problem in this context.

Both LILO and GRUB therefore work in stages. The first stage is always located in the Master Boot Record<sup>1</sup> (the first 512 bytes<sup>2</sup> on disk), and basically consists of two things:

- The disk block locations of the next stage (but not the filename)
- A small program, which loads these disk block locations into memory, and executes them.

The next stage is not limited to 512 bytes, and is typically able to load the kernel into memory and execute it. With LILO, the kernel file is still referenced using disk block locations. GRUB has more knowledge about file system structures and filenames, and is able to reference the kernel image using its name, but earlier stages of GRUB need to be loaded using disk block locations. In addition to this, LILO and GRUB make use of the BIOS read/write routines for loading these blocks into memory.

This whole scheme leads to three problems, which can be solved individually by other means, but are just as easily solved by creating a /boot partition.

The first problem is that the traditional BIOS specification does not allow for disks larger than 1024 cylinders (if Cylinder/Head/Sector addressing is used) or 8 GB (if Linear Block Addressing is used). By creating a small /boot partition and making sure that this partition is completely located below the 1024 cylinder/8 GB limit, this problem is eliminated.

Most newer BIOSes provide the "Extended INT13" function, to allow programs to read beyond the 1024 cylinders/8 GB boundary. LILO, in most cases, supports this with the lba32 parameter. GRUB supports this by default.

The second problem is that, because the first stage of both LILO and GRUB know only the block locations of the files to be loaded, a reorganization of the file system might cause the wrong blocks to be loaded. Such a reorganization does not happen often, but certain events (such as a disk reporting soft errors) might trigger such a reorganization without you even knowing it. And certain administrative tasks, such as copying or moving files, or

<sup>1</sup> Unless another bootloader, such as BootMagic is used. In that case, LILO or GRUB are loaded in the first sector of a primary partition.

<sup>2</sup> Actually, 446 bytes, since the 64-byte partition table and a 2-byte magic number also reside in the Master Boot Record.

restoring a backup, might also cause file blocks to be moved. And this might lead to boot problems when the system is eventually rebooted, which might be months later.

By creating a small /boot partition with only files in it that are needed when the system boots, and not accessed otherwise, you reduce the chance of the above happening to a minimum.

Most distributions currently use the ext3 file system as their default file system. This file system stores files in regular sized blocks. In the future, newer file systems might not do this. In addition to which LVM and RAID also use other and possibly incompatible ways of organizing and reorganizing partitions.

By getting into the habit of creating a /boot file system which does not use LVM or RAID or both, and which is formatted as ext3, you save yourself from nasty surprises in the future.

**Transition statement** — Let's look at boot loader configuration.

## Configure a boot loader

- A boot loader loads and starts the Linux kernel.
- It can boot other operating systems as well.
  - Windows, BSD, and so on
  - Give each OS a unique label
- It can be password protected.
  - Prevents users from passing boot parameters to Linux or booting any OS
- Should generally be configured in the MBR unless another boot loader is used.
- Common boot loaders include:
  - GRUB: Grand Unified Boot Loader
  - YaBOOT: Yet Another Boot Loader

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-9. Configure a boot loader

LX158.0

### Notes:

One of the next screens allows you to configure a Boot Loader. This is a program that loads and starts the Linux kernel. It can also pass various boot parameters to the Linux kernel, such as device information.

A boot loader can also be used to boot non-Linux operating systems, such as Windows. For this to work, your boot loader, your boot loader typically needs to be stored in the master boot record (/dev/hda). If you use another boot loader, such as BootMagic, then the boot loader that loads Linux is usually stored in the Linux root partition (the partition that holds the root file system).

Every OS that needs to be bootable is identified with a label, which you can choose yourself. This label is used to select the operating system that is booted when your system is switched on. If you don't make a selection, then after a number of seconds (usually 5), the default OS is booted. Currently, two boot loaders are in use: GRUB and YaBOOT.

GRUB, the GRand Unified Bootloader, is a successor to LILO, an older boot loader specifically written to load Linux. YaBOOT (Yet Another Boot Loader) It is an Open Firmware executable that bootstraps the Linux kernel on ppc hardware.

All boot loaders support passwords. These passwords, if configured, are required if the user wants to pass parameters to the kernel when the system is booting. These parameters could, for instance, be used to boot the system into single user mode, where the user automatically becomes root without having to log in.

**Instructor notes:**

**Purpose** — Discuss boot loaders.

**Details** — The Initial RAM disk is a compressed disk image, which is loaded into a RAM disk and uses as initial root disk. This disk image contains the modules and scripts to mount the real root disk on /initrd. The script then calls a special system call pivot\_root, which exchanges the real and the initial ram disks mount points so that the real root disk is now mounted as /, and the initrd as /initrd. From that point on, /sbin/init takes over.

**Additional information —**

**Transition statement** — Let's look at network configuration.

## Configure network

---

- Most distributions configure your network adapter as part of the installation process.
- You need the following information:
  - IP address
  - Subnetmask
  - Network address
  - Broadcast address
  - Host name
  - Default router/gateway
  - DNS server addresses
- They can also be configured to use DHCP.
- Most distributions do not support wireless adapters at install time.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-10. Configure network

LX158.0

### Notes:

Most distributions can configure your (Ethernet or Token Ring) network adapter during the installation. For this to work, you need to obtain the following information from your network manager:

- IP address
- Subnetmask
- Network address
- Broadcast address
- Host name
- Default router/gateway
- DNS server addresses

If your workstation resides on a network where a DHCP server is present, you can also configure your system to use this DHCP server to obtain this information.

***Instructor notes:***

**Purpose** — Discuss network configuration.

**Details** —

**Additional information** —

**Transition statement** — You also need to configure some accounts.

## Configure root and user accounts

---

- *root* is the superuser of the system.
  - It can do anything.
  - It needs a strong password.
  - **Do not use your system as root unless you need to!**
- Most distributions allow you to add user accounts during installation as well.
  - **Create a user account for every individual user that is going to use the system.**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-11. Configure root and user accounts

LX158.0

### Notes:

On a Linux system, the superuser is called *root*. If you (or anybody else) is logged into the system as this user, you can do anything to the system. This is considered very dangerous, and that is why you need to configure a strong password for this account. A good policy to live by is never to use the *root* account unless you really need to. You can use authorization elevation tools such as *sudo* instead.

Most distributions also allow you to add regular user accounts during the installation. If your distribution does this, then create user accounts for every user that is going to use your system.

Some distributions also allow you to configure your workstation as an NIS, LDAP, or Kerberos client, or as a client of some other network authentication method. Only do this if your network supports this (ask your network administrator).

***Instructor notes:***

**Purpose** — Discuss the authentication screens.

**Details** —

**Additional information** —

**Transition statement** — Let's look at the package group selection.

## Select package groups

---

- Most distributions group individual packages in package groups.
- Only select the package groups you need on your workstation.
- Selecting individual software packages is usually still possible but tedious.
  - A typical distribution has over 1000 packages.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 2-12. Select package groups

LX158.0

### Notes:

A typical distribution consists of over 1000 individual packages (software components) that should or can be installed. To save the user from having to make 1000 or more informed decisions, these packages are often grouped into package groups. Instead of having to decide on each and every individual package, you decide whether to install a package group. This greatly reduces the complexity of selecting the packages you want to install. Most distributions still offer the select individual packages option though, in case you have nothing better to do or need to run a really tight system (security-wise or hard disk space-wise).

***Instructor notes:***

**Purpose** — Discuss the package selection screens.

**Details** —

**Additional information** —

**Transition statement** — Another component that needs to be configured is X.

# Configure X

- X (X Window System) is the graphical user interface of Linux.
- It needs to be configured for your system.
  - Graphical adapter
  - Monitor
- Most adapters and monitors are autodetected.
  - If not autodetected, select manually or use a generic adapter or monitor
- Usually, customization is allowed.
  - Resolution, refresh rate
  - Color depth
- Test settings if possible.
- If nothing works, skip X configuration.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-13. Configure X

LX158.0

## Notes:

The *X Window System* (X for short) is the graphical user interface (GUI) of Linux. It needs to be configured for your graphical adapter and monitor to provide optimal performance. Most distributions incorporate auto detection mechanisms, which detect your adapter and monitor automatically. If this fails, select your adapter and monitor manually or use a generic adapter or monitor.

Within the limits of your adapter and monitor, you can customize your resolution (the amount of pixels on your screen), your refresh rate (the number of times your screen is refreshed, per second) and the color depth (the amount of colors that can be displayed simultaneously). There usually is a trade-off to be made here: a higher resolution usually means less color depth, and a higher resolution also means a lower refresh rate. It is therefore important that, if possible, you test the configuration before you continue the installation process.

Occasionally, it happens that X cannot be configured from the installation process at all. Trying to configure X might in some rare case even hang the system altogether.

***Instructor notes:***

**Purpose** — Discuss X configuration.

**Details** —

**Additional information** —

**Transition statement** — Different distributions can add other screens as well. Let's take a look at that too.

## Other (optional) installation screens

- Some distributions offer additional installation screens
  - Printer configuration
  - Firewall configuration
  - Sound card configuration
  - Modem configuration
  - Time zone configuration
- These are usually straightforward



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-14. Other (optional) installation screens

LX158.0

### Notes:

Some distributions offer other configuration screens in addition to the ones we have covered. This might include configuration of printers, firewalls, sound cards, modems, time zones and so forth. These screens are usually straightforward, and if they are not, there is usually help available.

**Instructor notes:**

**Purpose** — Discuss additional screens.

**Details** —

**Additional information** —

**Transition statement** — After we've made all configuration choices, the system starts installing packages.

## Installing packages

---

- The installation will take several minutes to complete.
  - Most distributions provide a progress bar or total time indication or both.
- While the installation is going on, various virtual terminals provide background information on the progress.
  - Switch between VTs using **Ctrl-Alt-F1**, **Ctrl-Alt-F2**, and so forth.
- Insert additional media when prompted.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 2-15. Installing packages

LX158.0

### Notes:

After all configuration choices have been made, the system starts installing packages. This installation might take a short or a long time, depending on your hardware and amount of packages to install. Most distributions, however, display some sort of progress bar or time indication so that you can estimate how long your coffee or lunch break is going to be.

While the installation is going on, you can get additional information about it from various virtual terminals. Virtual terminals are pseudo-monitors, which can be accessed using **Ctrl-Alt-F1**, **Ctrl-Alt-F2**, and so forth. Make sure to insert additional media when prompted.

**Instructor notes:**

**Purpose** — Discuss package installation.

**Details** —

**Additional information** —

**Transition statement** — When all packages are installed, there are usually a few things left to do. Usually the last thing to do is to reboot the system.

## Post-install configuration

---

- After installation has finished, your system will reboot to activate the newly installed kernel.
  - SLES performs the reboot during installation
- After reboot, some post-installation configuration might happen.
  - Configure graphics
  - Configure sound card
  - Install documentation, updates, and drivers
  - Create user accounts
  - Register

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-16. Post-install configuration

LX158.0

### Notes:

For most distributions, after the packages have been installed and configuration has been done, the install is over. The only thing left to do is to reboot the system to activate the newly installed kernel. SUSE is an exception to this: SUSE already activates the kernel (through a reboot) during the installation.

After the reboot, some post-installation configuration might happen, depending on the distribution. Red Hat, for instance, attempts to register your system with RHN, the Red Hat Network.

## 2.2. Automated installations

### Instructor topic introduction

**What students will do —** Learn how to perform automated installations.

**How students will do it —**

**What students will learn —**

**How this will help students on their job —**

***Instructor notes:***

**Purpose** — Discuss the reboot.

**Details** —

**Additional information** —

**Transition statement** —

# RHEL Kickstart installations

- RHEL method of automating installations
- Text file with three sections:
  - Install commands
  - %packages section
  - %pre, %post sections
- File creation:
  - Manually
  - `system-config-kickstart`
  - `/root/anaconda-ks.cfg` (created during installation)
- Location:
  - Boot floppy or NFS server
    - NFS also requires a Dynamic Host Configuration Protocol (DHCP) server
- Initiation:
  - `linux ks=ks.cfg URL at syslinux boot: prompt`

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-17. RHEL Kickstart installations

LX158.0

## Notes:

Kickstart provides the ability to create a hands-off installation. This means that if the `ks=` option is passed to the installer, the install program will take input from the configuration file. The contents of the kickstart configuration file can contain all of the answers to questions posed by the Anaconda installer, plus post-install directives. If the configuration is missing information, the install process will prompt the installer for additional information.

### File creation

The `ks.cfg` file is a flat text file. After a system has been installed, a kickstart file that configures the system to the way that it was installed is located in root's home directory in a file `/root/anaconda-ks.cfg`. To create a kickstart file from scratch, use the `system-config-kickstart` utility.

### Location

The kickstart file can be presented either on a diskette, or from a network source, such as an NFS server.

### Initiation

Kickstart is Red Hat and Fedora's method of automating installations. It involves creating a `ks.cfg` file, which contains three sections:

- The first section, which starts at the top of the file, contains the answers to all questions of the installation process. For instance, if the statement lang en\_US is present in the kickstart file, the question “What language do you want to use during the installation process?” will not be asked, and US English is used.
- The second section starts with the %packages identifier. It contains a list of all packages (RPMs) to be installed. Just as with the install process itself, it can also use the package groups that are defined in the component groups XML file provided by the distribution (comps-rhel5-server-core.xml in RHEL5, Fedora-8-comps.xml in Fedora), located in the repodata/ directory. These package groups are identified with an at sign (@), for instance “@ Printing Support”.
- The third section starts with the %post identifier. It contains a series of shell commands that are executed once the installation has finished. These are executed on the newly installed system, with all paths, networking, and so forth intact. This means that virtually anything is possible, including mounting remote file systems, creating user accounts, and so forth.

It is also possible to create a %pre section, which is executed before the installation starts. This is generally used only to implement custom partition schemes. Kickstart files can be created by hand, but Red Hat has also released a tool which might help you generate kickstart files: system-config-kickstart (formerly known as ksconfig). This tool is available on the distribution CDs in the system-config-kickstart RPM. As an added bonus, the RHEL/Fedora installer, Anaconda, generates a kickstart file for you based on the choices made during the installation process itself. This file is called /root/anaconda-ks.cfg.

The kickstart configuration file can be stored on the boot diskette, or can be stored on a network server. Kickstart installs are then started by typing linux ks=URL where URL is the location where the ks.cfg file is stored. Examples are ks=floppy and ks=http://10.0.0.1/kickstart/ks.cfg. If you do not supply a URL (linux ks), then the location of the kickstart file is taken from the DHCP next-server and filename option fields in the DHCP reply from the DHCP server.

**Instructor notes:**

**Purpose** — Discuss RHEL/Fedora kickstarts.

**Details** — Discuss the student notes, pointing out the simple nature of a kickstart file. This visual is intended to set the overall concept of a kickstart file. The following visual will provide an example you can use to point out the structure of the file.

**Additional information —**

**Transition statement** — Let's look at an example of a kickstart file.

## RHEL Kickstart example (abbreviated)

```
install
nfs --server=192.168.2.1 --dir=/export/rhel63
lang en_US.UTF-8
langs support --default=en_US.UTF-8 en_US.UTF-8
keyboard us
network --onboot yes --device eth0 --bootproto dhcp --ipv6 auto
rootpw --iscrypted $1$Q1EsuwfB$aowfCXdJRUcpW/8h4Jl0c.
firewall --disabled
...
.

%packages
@base
@core
@java-platform
...
.

%post
useradd -m -c "Tux the Penguin" tux
echo penguin | passwd --stdin tux

%end
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-18. RHEL Kickstart example (abbreviated)

LX158.0

### Notes:

The visual shows a small portion of a kickstart configuration file. The following is the complete example of the kickstart file that is shown in the visual. Note the name is anaconda-ks.cfg. This file was created during the installation of RHEL63. Can you tell what software was selected for this installation? What time zone will the system use?

```
[root@sys2 ~]# more anaconda-ks.cfg
# Kickstart file automatically generated by anaconda.
install
cdrom
lang en_US.UTF-8
langs support --default=en_US.UTF-8 en_US.UTF-8
keyboard us
xconfig --startxonboot --defaultdesktop gnome --startxonboot
--defaultdesktop gnome
network --device eth0 --bootproto static --ip 10.0.0.3 --netmask
255.255.255.0 --gateway 10.0.0.100 --hostname sys3
rootpw --iscrypted $1$Q1EsuwfB$aowfCXdJRUcpW/8h4JlOc.
firewall --disabled
selinux --enforcing
authconfig --enableshadow --enablemd5
timezone America/Los_Angeles
bootloader --location=mbr --append="rhgb quiet"
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
#clearpart --all --drives=hda
#part /boot --fstype ext3 --size=100 --ondisk=hda
#part pv.4 --size=0 --grow --ondisk=hda
#volgroup VolGroup00 --pesize=32768 pv.4
#logvol / --fstype ext3 --name=LogVol00 --vgname=VolGroup00
--size=1024 --grow
#logvol swap --fstype swap --name=LogVol01 --vgname=VolGroup00
--size=512 --grow
--maxsize=1024
%packages
@ everything
e2fsprogs
grub
kernel
lvm2
kernel-devel
```

The final part of the file shows that all software (everything) was selected for this installation. The time zone is set near the beginning of the file. This system is set to America/Los Angeles.

## **Instructor notes:**

**Purpose** — Provide an example of a kickstart file.

**Details** — Discuss the example file presented. Use this chance to confirm the student understands the general structure of the file and the fact it is very easy to read (much easier than the XML file we will discuss in the following visuals that is used in SLES). Verify the students see the answers to the questions about the time zone and software selection that are asked in the student notes section.

## **Additional information —**

**Transition statement** — Let's look at how SLES does this too.

## SLES AutoYaST installations

- SUSE Linux method of automating installs
- XML file containing all installation information:
  - General settings for keyboard and so forth
  - Partition settings
  - Packages
  - Pre- and post-install scripts
- File creation:
  - `yast autoyast`
- Location:
  - Store file on network server
- Initiation:
  - `install=nfs://10.0.0.1/export/files/sles11 \autoyast=nfs://10.0.0.1/export/files/myprofile.xml`

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-19. SLES AutoYaST installations

LX158.0

### Notes:

SLES also supports auto installations. On the most recent distributions, this is done through AutoYaST. Earlier SLES distributions used other, more complicated and limiting ways of performing auto installations.

**File creation:** AutoYaST installations revolve around an XML-based file containing all the installation information. This file can technically be created by hand, but that's a huge task. It is far easier to use `yast autoyast` to create this file.

**Location:** This file is saved on a network server. You then need to boot the system from regular boot media.

**Initiation:** In order to start the install, you need to supply two URLs to the boot loader:

- The first URL is the URL where the installation images can be found. This URL generally has the form `install=nfs://10.0.0.1/export/sles10sp1`
- The second URL is the URL where the AutoYaST file can be found. This URL generally has the form `autoyast=nfs://10.0.0.1/autoyast/myprofile.xml`

In addition to this, you might also need to specify the network adapter and type. This typically looks like: `insmod=eepro100 netdevice=eth0`. Just as with RHEL and Fedora, it is possible to modify the `syslinux.cfg` file on the boot floppy to start the installation manually.

### **Instructor notes:**

**Purpose** — Discuss the SLES autoyast feature.

**Details** — Discuss student notes. Point out the XML file used under SLES is much more complex than the kickstart file used under Red Hat.

**Additional information** — Stress that, unlike Red Hat, there is no sample autoyast file created at initial installation. You will have to create your own file using the tool provided.

**Transition statement** — Let's look at an example of an AutoYaST file.

## AutoYaST example (abbreviated)

```

<?xml version="1.0"?>
<!DOCTYPE profile SYSTEM "/usr/share/autoinstall/dtd/profile.dtd">
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://www.suse.c
om/1.0/configns">
  <bootloader>
    <activate config:type="boolean">false</activate>
    <global>
      <embed_stage1.5 config:type="boolean">true</embed_stage1.5>
      <gfxmenu>/boot/message</gfxmenu>
      <lines_cache_id>0</lines_cache_id>
      <prompt>1</prompt>
      <stage1_dev>/dev/hda7,/dev/hda</stage1_dev>
      <timeout config:type="integer">8</timeout>
    </global>
    <initrd_modules config:type="list">
      <initrd_module>
        <module>piix</module>
      </initrd_module>
      <initrd_module>
        <module>processor</module>
      </initrd_module>
      <initrd_module>
        <module>thermal</module>
      </initrd_module>
    . . .
  
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
 US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-20. AutoYaST example (abbreviated)

LX158.0

### Notes:

The visual shows a small portion of an AutoYaST XML-formatted configuration file. Note the tag format.

**Instructor notes:**

**Purpose** — Provide an example of an AutoYaST file.

**Details** — Point out the more complicated structure of this XML file as compared to the RHEL kickstart file. Emphasize that the file should be built with the SLES tools instead of from scratch (unless the student is *very* comfortable with XML documents).

**Additional information —**

**Transition statement —**

## Checkpoint (1 of 2)

1. True or false: Linux can coexist with Windows on the same hard disk.
2. Which of the following steps is not essential in the installation process:
  - a. Create partitions for Linux.
  - b. Configure your printer.
  - c. Select your keyboard type.
  - d. Identify the packages to install.
3. What is the best source of information about your hardware?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-21. Checkpoint (1 of 2)

LX158.0

### Notes:

**Instructor notes:**

**Purpose —**

**Details —**

## Checkpoint solutions (1 of 2)

---

1. True or false: Linux can coexist with Windows on the same hard disk.

The answer is true.

2. Which of the following steps is not essential in the installation process:

- a. Create partitions for Linux.
- b. Configure your printer.
- c. Select your keyboard type.
- d. Identify the packages to install.

The answer is configure your printer.

3. What is the best source of information about your hardware?

The answer is a currently installed and configured operating system, such as Windows.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —**

**Transition statement —**

## Checkpoint (2 of 2)

4. True or false: A network install server needs to be a Linux system.
5. Which of the following install methods does not require a network server?
  - a. NFS
  - b. SMB
  - c. FTP
  - d. CD-ROM

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-22. Checkpoint (2 of 2)

LX158.0

### Notes:

**Instructor notes:**

**Purpose —**

**Details —**

## Checkpoint solutions (2 of 2)

---

4. True or false: A network install server needs to be a Linux system.  
The answer is false. It can be any operating system that provides NFS, FTP, or HTTP services.
  
5. Which of the following install methods does not require a network server?
  - a. NFS
  - b. SMB
  - c. FTP
  - d. CD-ROM

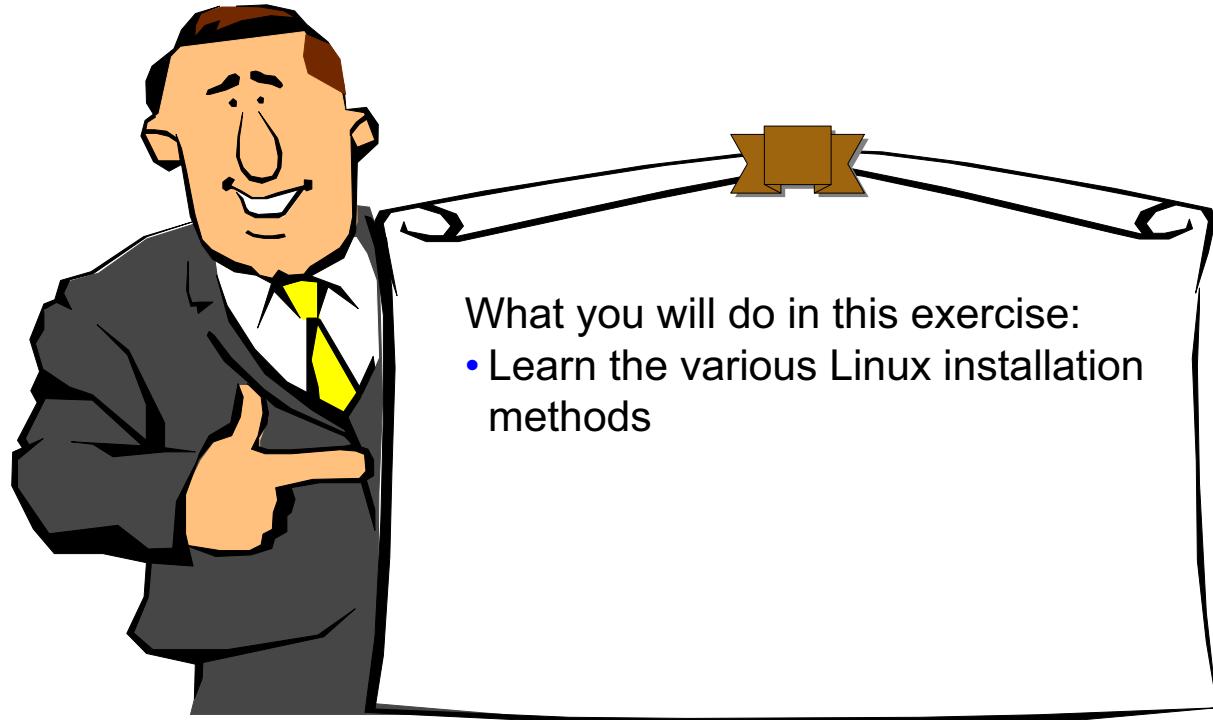
The answer is CD-ROM.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —**

**Transition statement —**

## Exercise: Installing Linux



What you will do in this exercise:

- Learn the various Linux installation methods

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-23. Exercise: Installing Linux

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Unit summary

Having completed this unit, you should be able to:

- Prepare a system for installation
- Install Linux from CD-ROM
- Perform a network installation
- Set up and perform a RHEL Kickstart install
- Set up and perform a SLES AutoYaST install

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 2-24. Unit summary

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

# Unit 3. Linux documentation

## Estimated time

00:15

## What this unit is about

This unit describes the use of the **man** and **info** commands as well as the HOWTO documentation. This unit will also explain the importance of the Internet for gathering information about Linux.

## What you should be able to do

After completing this unit, you should be able to:

- Use the **man** command to view information about Linux commands
- Describe the use of **info**
- Describe the HOWTO documentation
- Explain the importance of the Internet for gathering information about Linux

## How you will check your progress

- Checkpoint questions
- Lab exercises

## Unit objectives

---

After completing this unit, you should be able to:

- Use the man command to view information about Linux commands
- Describe the use of info
- Describe the HOWTO documentation
- Explain the importance of the Internet for gathering information about Linux

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## The man command

---

- With the **man** command, you can read the manual page of commands.
- Manual pages are stored in `/usr/share/man`.
- The manual page consists of the following:
  - **Name:** The name of the command and a one-line description
  - **Synopsis:** The syntax of the command
  - **Description:** Explanation of how the command works and what it does
  - **Options:** The options used by the command
  - **Files:** The files used by the command
  - **Bugs:** Known bugs and errors
  - **See also:** Other commands related to this one

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-2. The man command

LX158.0

### Notes:

The **man** command shows the manual page of the commands and subroutines given as an argument to the **man** command. Most manual pages consist of:

- **Name:** The title and a one-line description of the command
- **Synopsis:** The syntax of the command
- **Description:** Many pages of information about the function and usage of the command
- **Options:** An explanation of the options
- **Files:** Any system files associated with the command
- **Bugs:** Any information about the behavior/performance of the command in unusual circumstances
- **See also:** Other commands that are related to the same topic; viewing them can tell you more about the working of this particular command

You can search for a pattern in a manual page with the / key.

**Instructor notes:**

**Purpose** — Describe the main features of the man command.

**Details** — Explain how the man command can be used and when it is most useful to gain further information about a known command. It can provide information about commands that are specified, and it can provide information on all commands whose description contains a set of user-specified keywords.

A manual page is divided into a number of headings, which almost all have been listed in the student notes. Go through each heading explaining what might be seen in each section.

**Additional information** — To obtain more information about man enter `man man`.

**Transition statement** — Let's look at an example of man.

## man example (1 of 2)

```
$ man finger
FINGER(1)          BSD General Commands Manual      FINGER(1)

NAME
    finger - user information lookup program

SYNOPSIS
    finger [-lmsp] [user ...] [user@host ...]

DESCRIPTION
    The finger displays information about the system
    users.

    Options are:

        -s      Finger displays the user's login name, real
    Manual page finger(1) line 1
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-3. man example (1 of 2)

LX158.0

### Notes:

This is only the first screen of the manual page of the **finger** command. You can now use the less commands (spacebar, b, q, and so forth) to browse the page.

**Instructor notes:**

**Purpose** — Illustrate an example of the **man** command.

**Details** — Go through the synopsis diagram and ensure that students are familiar with the symbols used here, such as:

- **[]**: Optional fields (can be nested)
- **{a|b}**: Mandatory field: either a or b
- **...:** Repeat

**Additional information** — Sometimes the manual page of a command begins with a sentence such as: "This documentation is no longer being maintained and may be inaccurate or incomplete." The Texinfo documentation is now the authoritative source.

This means that this manual page is not up to date and you should use the **info** command to look at the up-to-date documentation. The **info** command is discussed further on in this unit.

**Transition statement** — So far we used **man** to view information about one specific command. We also can use the **man** command to search for a command that matches a keyword.

## man example (2 of 2)

- The **-k** option of the **man** command or the **apropos** command prints out a description of all entries that match the given keyword.

```
$ man -k print
arch (1) - print machine architecture
date (1) - print or set the system date and time
logname (1) - print user's login name
lpc (8) - line printer control program
lpd (8) - line printer spooler daemon
lpr (1) - off line print
lprm (1) - remove jobs from the line printer queue
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-4. man example (2 of 2)

LX158.0

### Notes:

The **man -k** command shows the commands that have manual pages that contain any of the given keywords in their title.

The **apropos** command can also be used and is equivalent to using the **man -k** command.

To allow the use of **man -k** and **apropos**, the superuser (root) must have run the **/usr/sbin/makewhatis** command to create the **/var/cache/man/whatis** file. Typically, a distribution or an administrator sets up a **cron** job so that this is done each night. This is covered in the LX03 course.

**Instructor notes:**

**Purpose** — Demonstrate how **man** can be used to search on a keyword.

**Details** — Explain the output in the example and that the manual page documentation is broken up into a number of different sections (as indicated by the number after the command).

**Additional information** — The **apropos** command can also be used and is equivalent to using the **man -k** command.

**Transition statement** — There are several types of man pages. Let's take a look at them.

## man sections

---

- The collection of manual pages is divided into nine sections:
  - 1. Executable or shell commands
  - 2. System calls
  - 3. Library calls
  - 4. Special files (usually found in /dev)
  - 5. File formats and conventions
  - 6. Games
  - 7. Miscellaneous (macro packages, and so on)
  - 8. System administration commands
  - 9. Kernel routines (non-standard)
- Certain subjects appear in multiple sections.
- To select the correct section, add the section number.
  - man 1 passwd (to learn about the **passwd** command)
  - man 5 passwd (to learn about the passwd file)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 3-5. man sections

LX158.0

### Notes:

Manual pages are stored in nine different sections. The first eight of them are standard across UNIX, and section nine is used for Linux kernel documentation. In some cases, a single subject might appear in multiple sections. As an example, **passwd** is both a command and a file; so a man page appears in two different sections. To retrieve a manual page from a specific section, specify the section number as the first argument to **man**.

**Instructor notes:**

**Purpose** — Show another command to view Linux documentation.

**Details** — The **info** command is best learned by using it yourself. Encourage students during the course to use the **info** command if they have questions about commands.

**Additional information** — The **info** command uses its own set of documentation files, which can be found in **/usr/share/info**. If you specify an argument that **info** cannot find in its own documentation, it looks in the manual pages. If the manual pages contain information about the command you specified, the **info** command displays the manual page of the requested command.

**info** is written as part of the GNU project, and most GNU utilities have an **info** page. However, a lot of non-GNU utilities do not have these pages.

**Transition statement** — Now that you have seen the **man** command, let's look at another tool to view at the information of commands.

## The info command

- The **info** command is sometimes a replacement for manual pages.
- It is widely used by the GNU project.
- Information for info is stored in /usr/share/info.
- Some info commands include:
  - <space> Next screen of text
  - <del> or <bs> Previous screen of text
  - n Next node
  - p Previous node
  - q Quit info

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-6. The info command

LX158.0

### Notes:

Another tool for viewing documentation is the **info** command.

The syntax of the info command is `info cmd_name`.

To view the documentation of the **info** command, enter `info info`. This displays the info documentation about the **info** command.

The **info** command works with entities named nodes. A node is one piece of information about a command or function. In **info**, you navigate through nodes to find and read information. The main difference between **info** and **man** is that these nodes can contain hyperlinks to other info pages, similar to the World Wide Web.

**info** has a lot of commands that help you navigate through the documentation. Some of these commands are:

- <space>: Next screen of text
- <del> or <bs>: Previous page of text
- N: Next node

- **P:** Previous node
- **U:** Go to up node
- **B:** To top of node
- **E:** To end of node
- **S:** Search for a string in the current node
- **?:** Go to the help
- **L:** Leave the help and go back to the node
- **Q:** Quit info
- **<tab>:** Jump to the next cross reference
- **F:** Follow this cross reference (this brings you to another node)
- **M:** Pick menu item specified by name.
- **<Ctrl-I>:** Refresh screen

Cross references are indicated by an asterisk (\*) on a line. With the **tab** key, you can jump to this cross reference. Pressing the **f** key makes **info** follow the cross reference and show you another node.

Another way of moving through **info** is by specifying menu items. A node only contains a menu when you see \* Menu : in the text. Menu items are also indicated by an asterisk (\*). Again use the **tab** key to jump to a menu item. Then press the **m** key and **info** asks you what menu item you want to go to. Just pressing **enter** makes you follow the link for the currently selected menu item. You could also enter another menu item, to follow its link.

The **info** command can be used to obtain the up-to-date information when a manual page starts with a sentence such as: “This documentation is no longer being maintained and may be inaccurate or incomplete. The Texinfo documentation is now the authoritative source.”

**Instructor notes:**

**Purpose** — Explain the screen contents of the **info** command.

**Details** — For the commands, students can use in info, point them to the student notes of the previous chart.

**Additional information** —

**Transition statement** — Let's look at an example of **info**.

## info example

- # info pwd

```
File: coreutils.info,  Node: pwd invocation,  Next: stty invocation, \
Up: Working context

19.1 `pwd': Print working directory
=====
`pwd' prints the name of the current directory. Synopsis:

  pwd [OPTION]...

The program accepts the following options. Also see *note Common
options::.

`-L'
`--logical'
  If the contents of the environment variable `PWD' provide an
  absolute name of the current directory with no `.' or `..'
  components, but possibly with symbolic links, then output those
--zz-Info: (coreutils.info.gz)pwd invocation, 38 lines --Top--
```

Figure 3-7. info example

LX158.0

### Notes:

The **info** command is invoked with an argument that is the command of which you want to view the documentation. On the screen, you see the following:

- **File:** The file that contains the node you are looking at
- **Node:** The current node
- **Next:** The next node. You can use the **n** command to jump to this node
- **Up:** Besides a next node, a node can also have an up node. Use the **u** command to jump to the up node.

The node you are viewing:

- **Lines:** The total number of lines for this node
- **Position:** All you see all the lines of the node.
- **TOP:** You are at the top of the node.
- **BOT:** You are at the bottom of the node.
- **75%:** You are at 75% of the node.

## **Instructor notes:**

**Purpose** — Introduce manual page sections.

**Details** —

**Additional information** —

**Transition statement** — The last two examples (man and info) are used to find information about commands. Other kinds of documentation are the FAQ and HOWTO documentation. Let's see what these are.

## The --help option

- This is another way of getting help about a command.
- Help is built in the command itself (if supported).

```
$ cat --help
Usage: cat [OPTION]... [FILE]...
Concatenate FILE(s), or standard input, to standard output.

-A, --show-all           equivalent to -vET
-b, --number-nonblank    number nonempty output lines
-e                         equivalent to -vE
-E, --show-ends          display $ at end of each line
-n, --number              number all output lines
-s, --squeeze-blank      suppress repeated empty output lines
-t                         equivalent to -vT
-T, --show-tabs          display TAB characters as ^I
-u                         (ignored)
-v, --show-nonprinting    use ^ and M- notation, except for LFD and TAB
--help                   display this help and exit
--version                output version information and exit
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-8. The --help option

LX158.0

### Notes:

As we already saw, the **man** and **info** commands can be used to obtain information about the working of a command. This information is stored in a separate file in **/usr/share/man** or **/usr/share/info**. Obviously, this manual page has to be installed.

Another way of getting help about a command is using the **--help** option of the command itself. This option shows you a brief explanation of the synopsis of the command and the options that can be used with the command. The information shown is part of the command itself, and does not require the presence of a separate file.

The visual shows some lines of the help the **who --help** command would give you. The actual output probably does not fit on your screen. To read the complete help, issue **who --help | less**, which shows the output by page.

Note that not all commands support the **--help** option. Conveniently, for most commands that do not support **--help**, invoking them with **--help** will produce an invalid parameter error message, and the usage information will be printed anyway.

## **Instructor notes:**

**Purpose** — To show an alternate way to get help for a command.

**Details** — Almost all commands in Linux have the --help option. This option is very useful when you already know what command you want to use but forgot which option you should use.

The **man** and **info** commands give more in depth information about the command.

### **Additional information —**

**Transition statement** — Let's turn our attention to other kinds of information in your system. First let's see the FAQ.

## HOWTO documents

- These are documents that describe in detail a certain aspect of configuring or using Linux.
- They include detailed information about how to perform a given task.
  - DHCP support
  - Kernel compilation
  - Dual boot with other operating systems
- HOWTO documents are text files in /usr/share/doc/HOWTO.
  - Need to be installed manually
- On the Internet:
  - <http://www.tldp.org/index.html>

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-9. HOWTO documents

LX158.0

### Notes:

Linux HOWTOs are documents that describe in detail a certain aspect of configuring or using Linux. For example, there is the installation HOWTO, which gives instructions on installing Linux, and the Mail HOWTO, which describes how to set up and configure mail under Linux. Other examples include the NET-3 HOWTO and the Printing HOWTO. HOWTOs are comprehensive docs, much like an FAQ but generally not in question-and-answer format. However, many HOWTOs contain an FAQ section at the end. There are several HOWTO formats available: plain text, PostScript, DVI, and HTML. In addition to the HOWTOs, there are a multitude of mini-HOWTOs on short, specific subjects. They are only available in plain text and HTML format.

## **Instructor notes:**

**Purpose** — Illustrate the HOWTOs.

**Details** — The difference between FAQ and HOWTO is that FAQ are questions like "What is a... ?," "Where can I find a... ?" and so forth. FAQ have relatively short answers.

HOWTOs, on the other hand, give answers on questions like "How do I configure a.....?", "What is the procedure to recover from a... ?" HOWTOs have relatively large answers. Technically speaking, HOWTOs are complete scenarios for completing a specific task.

**Additional information** — If you have a Linux system and a projector, you could show the students the contents of the FAQ and HOWTO documents on a running system.

**Transition statement** — Let's see an example of a HOWTO document.

# HOWTO example

```
$ zless /usr/share/doc/HOWTO/en-txt/UPS-HOWTO.gz
UPS HOWTO

Eric Steven Raymond
[http://www.catb.org/~esr/] Thyrsus Enterprises

Nick Christenson

Revision History
Revision 2.2      2007-05-22      Revised by: esr
An Uninterruptible Power Supply (UPS) is an important thing to have if
you live in an area where power outages are at all common, especially if
you run a mail/DNS/Web server that must be up 24/7. This HOWTO will
teach you things you need to know to select a UPS intelligently and

/usr/share/doc/HOWTO/en-txt/UPS-HOWTO.gz
```

Figure 3-10. HOWTO example

LX158.0

## Notes:

The example on the visual shows you the HOWTO on how to install and configure XFree86 on your system.

XFree86 is the graphical environment of a Linux system.

The example on the visual does not show the complete HOWTO. You see only the first 12 of 792 lines.

**Instructor notes:**

**Purpose** — Show the contents of a HOWTO document.

**Details** — Remind students that they are only seeing the first 12 lines of 792 lines.

**Additional information** — If your students want to become world famous, this is what they have to do: write a HOWTO and put it on the Internet. If it is useful then it becomes part of distributions and their name is part of the Linux history. Another way of becoming immortal is to write useful programs for Linux. As you should know from Unit 1, Linux is made of contributions from people all over the world.

**Transition statement** — Some programs also offer documentation written in HTML.

## Other documentation

- Certain programs also offer other kinds of documentation.
  - HTML
  - PDF
  - PostScript
  - Plain text
- These are usually stored in  
`/usr/share/doc/<package_name>`.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-11. Other documentation

LX158.0

### Notes:

When a programmer creates a program, the programmer usually includes the standard documentation, such as manual or info pages or both, and implements the --help option. But most programmers also write some non-standardized pieces of documentation. These are typically README files, with up-to-date release information, or CHANGELOGS, which list the changes since the previous versions. Other programmers might write large amounts of HTML-based documentation, or Postscript-based installation instructions, and so forth.

A typical distribution leaves this documentation intact and stores it in  
`/usr/share/doc/<programname>`.

In practice, the value of this documentation varies greatly. There are programmers who only use the standardized tools (man, info) and as a consequence, `/usr/share/doc/<programname>` is virtually empty. Other programmers have created a whole Web site about their program, consisting of more than 20 HTML pages with supporting graphics, example configuration files, and so forth. So your mileage might vary here.

***Instructor notes:***

**Purpose** — Cover other documentation.

**Details** —

**Additional information** —

**Transition statement** — All information about Linux can also be found on the Internet.

## Internet

- All Linux documentation is available on the Internet.
- Google: <http://www.google.com/linux>
- Other sites:
  - [www.tldp.org](http://www.tldp.org)
  - [www.linux.org](http://www.linux.org)
  - [www.redhat.com](http://www.redhat.com)
  - [www.novell.com/linux](http://www.novell.com/linux)
  - [www.fedoraproject.org](http://www.fedoraproject.org)
  - [www.kernel.org](http://www.kernel.org)
  - [www.lwn.net](http://www.lwn.net)
  - [www.ibm.com/developerworks/linux](http://www.ibm.com/developerworks/linux)
  - And many more



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-12. Internet

LX158.0

### Notes:

All information about Linux can also be found on the Internet. There are scores of Web pages on Linux. For more personal and up-to-date help, you can also go to Usenet news and other forums.

**Instructor notes:**

**Purpose** — Again, explain the importance of the Internet for getting information about Linux.

**Details** —

**Additional information** —

**Transition statement** — There are some checkpoint questions. Let's try and answer them.

## Unit review

- The **man** command can be used from the command line to view the proper syntax of Linux commands.
- Some commands have more complete documentation available by using the **info** command.
- Specific system administration tasks are described in the HOWTO documents.
- The Internet is the place for the latest information about Linux.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-13. Unit review

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Checkpoint

1. True or false: A HOWTO document is the best source of documentation if you want up-to-date information about a specific command.
  
2. The main Linux documentation Web site is:
  - a. <http://www.tldp.org>
  - b. <http://www.linux.org>
  - c. <http://www.lwn.net>
  - d. <http://www.kernel.org>
  
3. In which sections are manual pages divided?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-14. Checkpoint

LX158.0

### Notes:

**Instructor notes:**

Purpose —

Details —

## Checkpoint solutions

---

1. True or false: A HOWTO document is the best source of documentation if you want up-to-date information about a specific command.

The answer is false.

2. The main Linux documentation Web site is:

- a. <http://www.tldp.org>
- b. http://www.linux.org
- c. http://www.lwn.net
- d. http://www.kernel.org

The answer is <http://www.tldp.org>.

3. In which sections are manual pages divided?

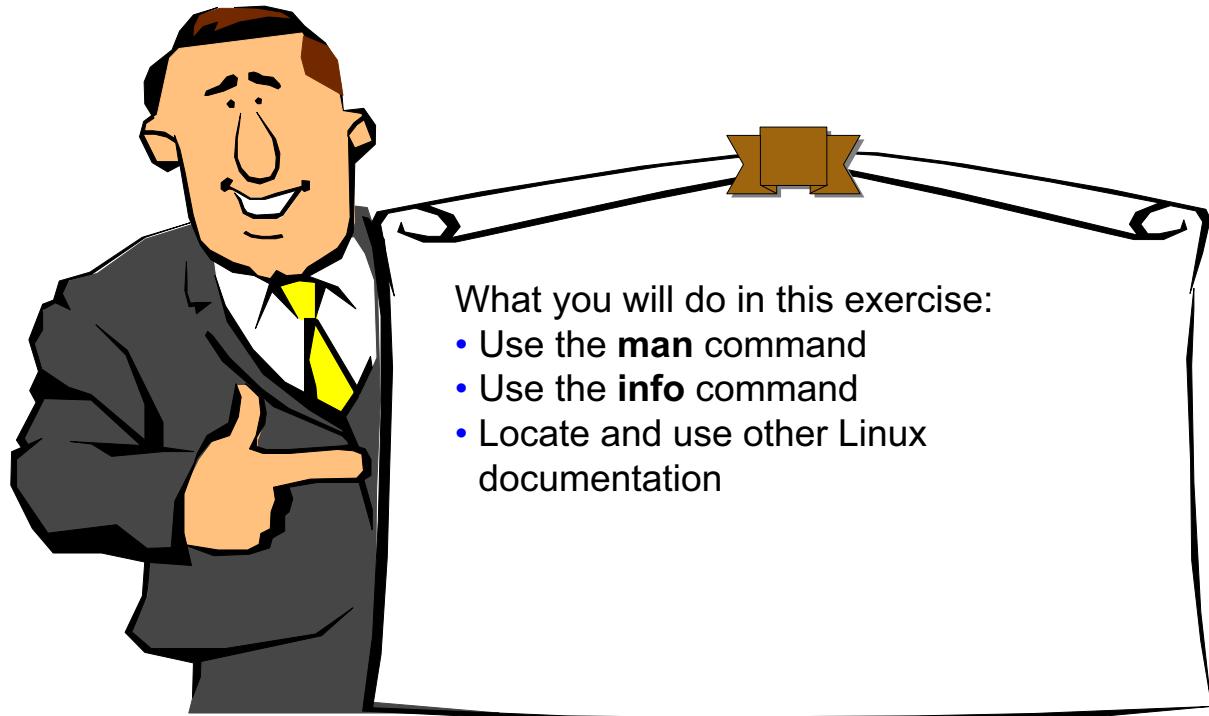
The answers are user commands, system calls, library calls, devices, file formats and protocols, games, conventions, macro packages, and so forth, system administration, and Linux kernel.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —**

**Transition statement —**

## Exercise: Linux documentation



What you will do in this exercise:

- Use the **man** command
- Use the **info** command
- Locate and use other Linux documentation

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-15. Exercise: Linux documentation

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Unit summary

Having completed this unit, you should be able to:

- Use the man command to view information about Linux commands
- Describe the use of info
- Describe the HOWTO documentation
- Explain the importance of the Internet for gathering information about Linux

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 3-16. Unit summary

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

# Unit 4. Startup and shutdown

## Estimated time

00:45

## What this unit is about

This unit teaches you how the startup process of a Linux system actually works and how to shut down a Linux system properly.

## What you should be able to do

After completing this unit, you should be able to:

- Describe the Linux startup flow
- Configure autostarting services
- Boot Linux in single-user mode
- Perform a proper shutdown of a Linux system

## How you will check your progress

Accountability:

- Checkpoint questions
- Machine exercises

## Unit objectives

---

After completing this unit, you should be able to:

- Describe the Linux startup flow
- Configure autostarting services
- Boot Linux in single-user mode
- Perform a proper shutdown of a Linux system

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 4-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

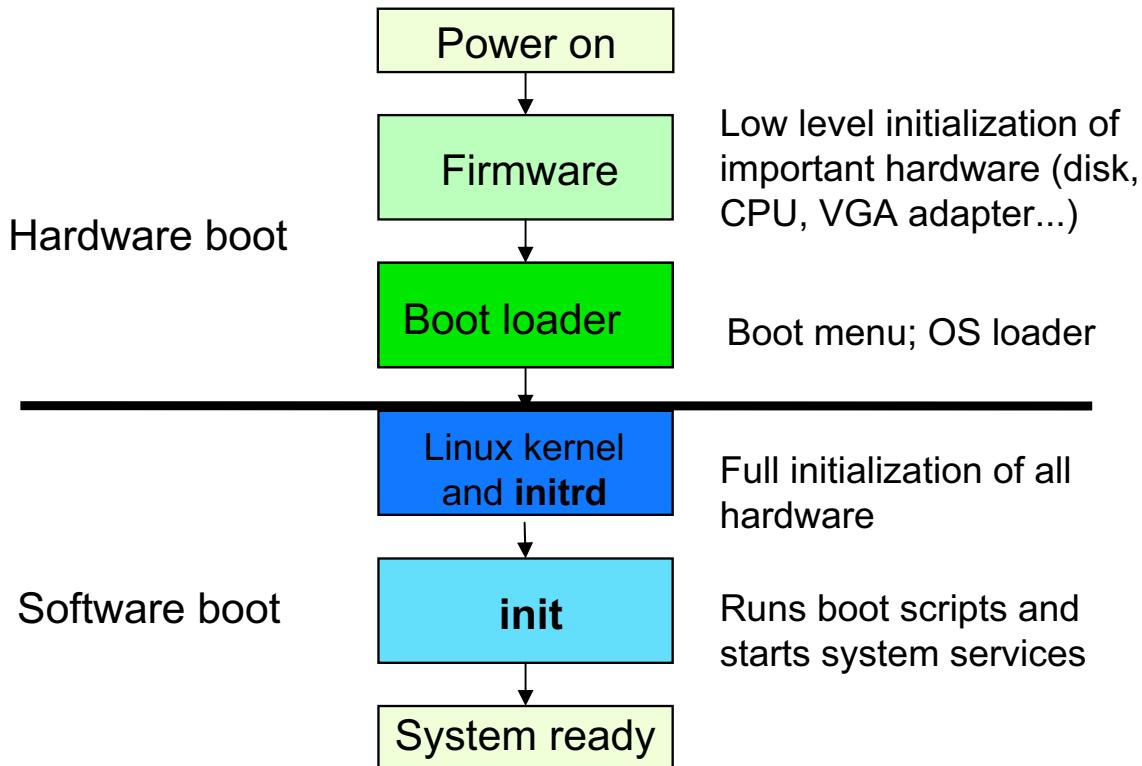
**Purpose** — Define unit objectives.

**Details** —

**Additional information** —

**Transition statement** — Let's start with a look at the startup flow of Linux.

## Linux startup flow



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-2. Linux startup flow

LX158.0

### Notes:

#### Introduction

This visual gives an overview of the Linux startup flow. In the subsequent visuals, details about each step will be covered.

**Instructor notes:**

**Purpose** — Introduce the basic startup flow.

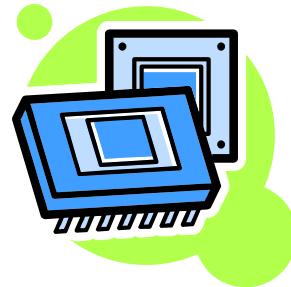
**Details** — Do not go into details just yet. This visual is just a short overview. If possible, however, it might be a good idea to leave this visual on a separate overhead projector for the rest of the lecture.

**Additional information —**

**Transition statement** — Next, let's take a look at the initial step of entering into the firmware level.

# Basic Input/Output System

- Firmware stored in NVRAM
- Checks memory and hardware (POST)
- Loads options from nonvolatile memory
  - Memory timings
  - Order of boot devices
- Checks for boot devices
  - Floppy disks
  - CD-ROM
  - Hard disks
- Loads boot loader from boot device and executes it
  - Normally stored in first sector; additional code may be loaded based on disk location (C/H/S or LBA)
  - CD/DVD store boot loader according to the El Torito standard



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-3. Basic Input/Output System

LX158.0

## Notes:

### Introduction

Every computer has a low level firmware system that defines the interfaces of that computer.

The firmware system is stored in an Electrical Erasable Programmable Read Only Memory (EEPROM), (sometimes also called non-volatile memory or NVRAM) on the motherboard. It is the first program that runs once the power is switched on.

On pre-2010 x86-based computers, the firmware system is typically the *Basic Input/Output System* (BIOS).

System firmware performs a number of basic tasks:

- Completes Power On Self Test (POST) to check memory and hardware.
- Loads various options from non-volatile memory, for instance, memory timing parameters, interrupt request (IRQ) assignment to devices, and the order of boot devices. These options can be set by the user when pressing **Del**, **F1**, **F2**, or some other key while the memory is being tested.

- Checks for the availability of boot devices.
- Loads the boot loader from the first available boot device. If the boot device is a floppy, hard disk or USB key, the boot loader will be loaded from the first sector. If the boot device is a CD/DVD, then the boot loader will be stored in a location defined by the El Torito standard. For a network boot, the BIOS typically hands control to the network adapter, which will boot the system using the PXEBoot standard.

***Instructor notes:***

**Purpose** — Explain the role of system firmware.

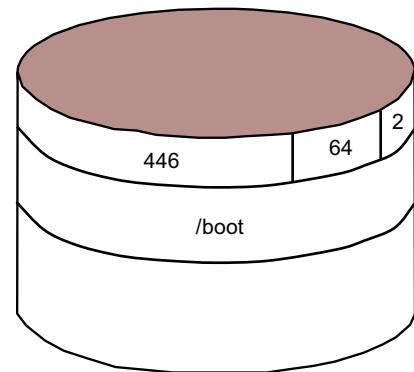
**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Of course, you wonder what this boot loader is. Let's look at that.

# BIOS Boot Loader location

- Master Boot Record
  - Size: 512 bytes (first sector of HD)
  - Addressed by BIOS
  - Content:
    - 446-byte program code (Stage 1 of the boot loader)
    - 64-byte partition table with max. four entries
    - 2-byte "magic number" (0xAA55)
- /boot partition
  - Regular partition, typically 64-100MB
  - Contain other files related to the boot process
    - Boot loader other stages
    - Boot loader configuration file
    - Kernels, initrd



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-4. BIOS Boot Loader location

LX158.0

## Notes:

Traditionally the boot loader of an Intel-compatible operating system is stored in the Master Boot Record, which is located in the very first sector (512 bytes) of the boot disk. However, since the MBR also contains the partition table and a magic number, the size of the boot loader code is limited to 446 bytes.

446 bytes is not a lot to contain a program. Modern bootloaders do not fit in there. Instead, what goes into the MBR is only the first stage. This stage is just intelligent enough to load further stages, typically located in a special /boot partition<sup>1</sup>.

With these further stages, the boot loader is complete. It will be able to read its own configuration file, wait for user input and then load the proper kernel and initrd. These other files are typically stored in /boot as well.

<sup>1</sup> If you don't have a separate /boot partition, these further stages will be stored in the root file system. Your BIOS however may not be able to load these stages if they are located on disk beyond 1024 cylinders or 8GB. It is therefore good practice to create a small /boot partition at the start of the disk.

***Instructor notes:***

**Purpose** — Talk about the boot partitions and what they do.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Since about 2010 a competing firmware has hit the market. Let's take a look.

# Unified Extensible Firmware Interface

- Extensible Firmware Interface 1.0: Intel Standard
- Unified Extensible Firmware Interface 2.1+: Managed by United EFI Forum, adopted by many PC vendors
  - Most post-2010 PCs will have UEFI
- Replaces BIOS, but most implementations are downwards compatible ("Legacy mode")
- Requires disk to have a GUID Partition Table (GPT) layout
  - All EFI code (boot loaders) should be stored in first partition
  - First partition should be in FAT format
  - Multiple boot loaders can be stored in EFI/<vendor>/\*.EFI
  - Boot loaders can be registered or autodetected
- May include a boot menu to select the proper boot loader
- Boot loaders to present in menu can be configured from the UEFI menus and from the OS (Linux tool **efibootmgr**)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-5. Unified Extensible Firmware Interface

LX158.0

## Notes:

To support more capable PC systems, particularly those that were Itanium-based, in the mid-1990s Intel developed an alternative to the BIOS standard, called Intel Boot Initiative, which was later renamed Extensible Firmware Interface. Intel ceased development of this standard in 2005 and contributed it to the United EFI Forum. This forum is now the governing body for the Unified Extensible Firmware Interface, and this standard has been implemented in modern PCs and Intel-based servers since about 2010.

For downwards compatibility reasons, most UEFI implementations will also support "Legacy mode", which will allow you to boot from a hard disk that's not setup for UEFI.

The UEFI standard requires the boot hard disk to have a specific layout called the GUID Partition Table (GPT) layout. We will look at that layout in the next visual.

UEFI firmware supports loading of multiple boot loaders, which each in turn may support loading multiple OSs. Because of this, UEFI firmware may offer the user a boot menu to select the proper boot loader.

To support these multiple boot loaders, the UEFI code needs to be able to access the hard disk to autodetect which boot loaders are present, store the entry points for these boot loaders in NVRAM and apply a user-defined order. All this can be configured from the UEFI menus, but there is also a defined interface so that the UEFI NVRAM can be manipulated from the operating system itself. In Linux, this can be done with the tool **efibootmgr**.

The UEFI standard offers a lot of other features as well. One of these is "Secure Boot", which was introduced with the UEFI 2.2 standard. If Secure Boot is enabled, the UEFI firmware will only load and execute boot loaders which are signed with an acceptable digital signature. This will effectively prevent the loading of "rogue" boot loaders. If the boot loader subsequently verifies the digital signatures of the operating system files (kernel and initrd/initramfs) and the operating system subsequently verifies the digital signatures of the executables, then the only way a hacker would be able to install a rootkit or other malware, would be to go all the way back to the UEFI setup and manipulate all code and digital signatures from there. This will be next to impossible, so Secure Boot has the potential to greatly reduce the number of system infections with malware. However, a criticism of Secure Boot is that it makes installing a new operating system (one created by a different vendor than the OS the system originally shipped with, and thus signed by a different entity) very hard, since you will need to load new keys into the UEFI firmware.

***Instructor notes:***

**Purpose** — Discuss the UEFI standard

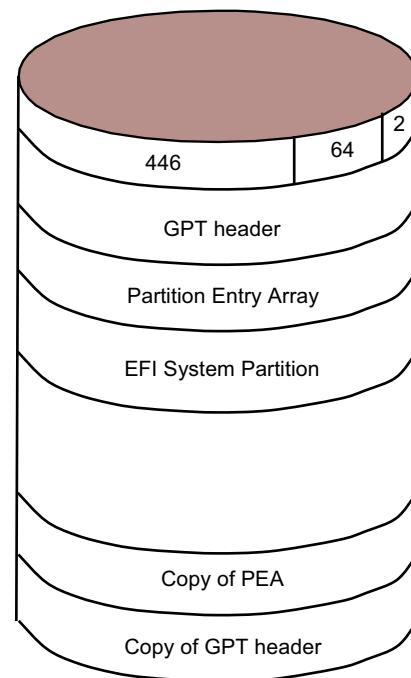
**Details** —

**Additional information** —

**Transition statement** — Let's look at the disk layout of an UEFI boot disk, as that will explain a lot.

# GUID Partition Table

- Wholly based on Logical Block Addressing
  - One LBA is normally a sector of 512 bytes
- LBA 0: Protective MBR
  - Prevents non-GPT tools from manipulating the GPT
- LBA 1: GPT header
- LBA 2-33: Partition Entry Array
- LBA 34+: Partitions
  - First partition is the EFI System Partition and contains the boot loader(s)
- GPT header and PEA is mirrored at the end of the disk



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-6. GUID Partition Table

LX158.0

## Notes:

The UEFI standard specifies that the boot disk has to be using a different partitioning scheme compared to the old "msdos" partitioning scheme which we saw earlier. This partitioning scheme is called "GUID Partition Table" or GPT.

GPT is wholly based on 64-bit Logical Block Addressing (LBA), so there is no requirement for a translation between Cylinder/head/sector and LBA addresses anymore. Furthermore, since the LBA addresses are 64 bit, instead of the 32-bit of the BIOS standard, disks that are larger than about 2.2 TB are now fully supported. In fact, the maximum disk size is about 9.4 Zetabytes.

The GPT standard defines the following uses for the sectors:

- The first sector, or "LBA 0", is the "Protective MBR" sector. It contains an msdos style partition table which only defines a single, whole disk GPT partition. On an UEFI system this table is never used as part of the boot sequence, but it prevents non-UEFI systems from booting improperly, and non-GPT tools (such as Linux `fdisk`) from accidentally deleting the GPT disk layout.

- The second sector, or "LBA 1" contains the GPT header. It contains generic information about the disk, such as the Globally Unique IDentifier (GUID), an EFI signature, some checksum information and the size/layout of the Partition Entry Array.
- LBA 2 through 33 then form the Partition Entry Array. This is the table of partitions. Each partition description is 128 bytes in size.

In fact, the PEA may be larger than 32 LBAs, but 32 LBAs is the minimum size and caters for 128 partitions in most circumstances. The situations where the PEA needs to be larger than 32 LBAs will be very rare indeed.

- Starting from LBA 34 the partition data can be found.

In an GPT-partitioned boot disk, the first partition is the EFI System Partition. This partition is formatted using the FAT12, FAT16 or FAT32 file system. All boot loaders should be contained in this partition, and should be stored as \EFI\<vendor>\\*.EFI. The UEFI firmware is able to read this file system, make a list of all available boot loaders, present this to the user to choose from (or use a predefined list) and then load the right \*.EFI file into memory to execute it. This means there is no requirement anymore to access boot files based on their LBA or C/H/S location, which is a significant improvement over the BIOS standard.

- At the end of the disk, a copy of the GPT header and PEA data is stored. This will be used in case of disk corruption at the start of the disk - such as would occur if you were to manipulate a GPT disk with **fdisk**.

## **Instructor notes:**

**Purpose** — Discuss the layout of a GPT disk

**Details** — A lot more detail can be found on Wikipedia:

[http://en.wikipedia.org/wiki/Unified\\_Extensible\\_Firmware\\_Interface](http://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface)

[http://en.wikipedia.org/wiki/GUID\\_Partition\\_Table](http://en.wikipedia.org/wiki/GUID_Partition_Table)

[http://en.wikipedia.org/wiki/EFI\\_System\\_partition](http://en.wikipedia.org/wiki/EFI_System_partition)

## **Additional information —**

**Transition statement** — Okay, so we've seen how either the BIOS or the UEFI loads the boot loader into memory. Let's now look at the most important Linux boot loader, GRUB.

## GRand Unified Bootloader

- Loaded into memory by BIOS or UEFI and started
- Understands file system structure
  - Can read configuration file from /boot filesystem
- Configuration file is /boot/grub/menu.lst (linked to /boot/grub.conf)
- When system boots:
  - Selects predefined OS to boot or
  - Uses command language to boot non-predefined OS
  - Command language compatible with configuration file
- Additional features:
  - MD5 encrypted passwords
  - Hiding/unhiding partitions
  - Graphical "splash" screens

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-7. GRand Unified Bootloader

LX158.0

### **Notes:**

The default bootloader of an Intel-based Linux system is the GRand Unified Bootloader (GRUB).

GRUB can be loaded both from a BIOS and from UEFI.

When loaded by the BIOS, GRUB is loaded in multiple stages:

- The first stage, called stage1 on disk, is usually stored in your MBR.
- The 1.5th stage, called \*\_stage1\_5 (e2fs\_stage1\_5, fat\_stage1\_5, minix\_stage1\_5, reiserfs\_stage1\_5, and so forth) is stored on disk, typically in /boot/grub. Several 1.5th stage files exist, each for a different file system.

This stage is used to add file system capabilities to GRUB so that GRUB is able to use regular filename references when loading configuration files, kernels and such, instead of disk block locations. Because of this stage, GRUB is able to read its configuration file directly and does not need to be configured beforehand.

This stage is loaded into memory by the stage1 code, based on its C/H/S or LBA location, as stage1 is not intelligent enough to understand file system structures.

- The second stage, called stage 2. This gives a menu interface which allows you to boot your predefined operating systems or enter commands to boot a non-predefined operating system.

Stage2 is loaded by stage1.5 based on the file system location (directory, filename, inode, ...). This is made possible because the stage1.5 is intelligent enough to understand a file system structure.

When loaded through UEFI, all the GRUB code is contained in a single executable, usually stored as /boot/efi/EFI/<vendor>/grub.efi.

## GRUB configuration file

The GRUB configuration file is typically stored in /boot/grub (BIOS) or /boot/efi/EFI/<vendor> (UEFI) and called menu.lst or grub.conf. It contains all predefined operating systems and their options and peculiarities.

The GRUB configuration file may also include references to "splash images" and other files to enhance the user experience.

## GRUB installation

To install GRUB on a BIOS system, either use the shell script **grub-install** or start the **grub** program and use GRUB commands to install GRUB manually. On an UEFI system, all you need to do is make sure the grub.efi file is in the proper location, and then enter the UEFI menus to configure GRUB as a boot loader.

## GRUB features

GRUB has some additional features that make it far more useful than earlier Linux bootloaders such as LILO:

- GRUB supports MD5-encrypted passwords to protect normal users from supplying parameters and options to predefined operating system or defining their own operating system boot procedure.
- GRUB can perform hiding and unhiding of Windows partitions. This can be used to keep Windows installations from interfering with one another.
- If configured properly, GRUB can be used to boot from the network. This requires the **netboot** package and setting up DHCP and TFTP servers. Network booting is outside the scope of this course.

***Instructor notes:***

**Purpose** — Introduce GRUB.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Let's now look at the GRUB configuration file.

# GRUB configuration example

```
#boot=/dev/hda
default=0
timeout=5
password --md5 $1$8auIK/$x7egV31ST8tCGVbuHn5y11
title Red Hat Enterprise Linux ES (2.6.9-34.EL)
root (hd0,0)
kernel /vmlinuz-2.6.9-34.EL ro root=/dev/hda5 rhgb quiet
initrd /initrd-2.6.9-34.EL.img
title Fedora Core (2.6.15-1.1955_FC5)
root (hd0,0)
kernel /vmlinuz-2.6.15-1.1955_FC5 ro root=/dev/hda6 rhgb quiet
initrd /initrd-2.6.15-1.1955_FC5.img
title SUSE SLES 11
root (hd0,0)
kernel /vmlinuz root=/dev/hda7 selinux=0 splash=silent showopts
initrd /initrd
title Windows XP
rootnoverify (hd0,1)
chainloader +1
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-8. GRUB configuration example

LX158.0

## Notes:

### GRUB configuration options

The GRUB configuration file, menu.lst or grub.conf, is nothing more than a predefined series of commands that could just as well have been entered on the GRUB command line. Storing these commands in a file, though, makes booting far more convenient.

The file in the visual is an example. minimal configuration file. It starts with a few general configuration options:

- **default=0** This specifies the default operating system to be started.
- **timeout=5** Timeout before starting the default operating system, in seconds.

When general options are all defined, specific operating systems need to be predefined.

For this, the following options might be needed:

- **title** The title of the operating system, as it shows up in the GRUB boot screen.

- **root** The root partition of the file system. All files that are referenced later on are stored on this file system. Specifying **root** is not required, but you will have to identify the root partition every time you mention a file instead, as is done with the SuSE stanza.
- **kernel** The kernel image that is to be loaded, and all options that need to be passed to the kernel.
- **initrd** An initial root disk that needs to be loaded.
- **rootnoverify** The root of the operating system is the partition specified, but do not try to verify and access this as GRUB does not support the file system type.
- **chainloader +1** To boot this operating system, invoke the chainloader, which needs to load the first sector of the specified root partition. This is typically used for operating systems such as Windows, who need to be loaded using their own boot loader.

In extreme cases, you can also use the **chainloader** directive to load a second instance of GRUB. The first instance is then installed in /dev/hda (the MBR), using the command **grub-install /dev/hda**, while the second instance is installed in the first sector of another partition, for instance /dev/hda5, with the command **grub-install /dev/hda5**.

There are several other GRUB options that may be useful and/or present in your distributions configuration file:

- **password** allows you to specify a password. This password should be entered to alter any boot options from existing stanzas, or to boot configurations that are not listed in the configuration file. In other words, if you set a password the users are only able to boot the operating systems exactly as listed in the configuration file, unless they know the password.

The password can be specified in plain text, but this is considered insecure. More commonly, the password is encrypted using MD5, and then entered in the GRUB configuration file as follows:

**password --md5 \$1\$8auIK/\$x7egV31ST8tCGVbuHn5yl1**

More elaborate settings are also possible, where only certain users are allowed to boot certain operating systems. For these examples, consult the GRUB manuals.

- **splashimage** allows you to specify a splash image that is used as the background for the GRUB menu. It is a gzipped XPM file. **splashimage** is mostly used on Red Hat or Red Hat derived systems such as Fedora and CentOS.
- **gfxmenu** performs more or less the same role as **splashimage**, but is a SUSE extension.
- **hiddenmenu** hides the complete GRUB menu. Instead it just displays a quick countdown timer. If you interrupt this timer, you will see the menu anyway.
- **makeactive, hide, unhide**: These three lines, which are found in a non-Linux boot stanza, configure the partition table so that the operating system you are trying to boot does not get confused by partitions that belong to other operating systems.

More specifically, hiding and unhiding of partitions makes it possible to have two or more Windows operating systems installed without conflict, and without confusion over which partition is your C:-drive and so forth. In addition to this, **makeactive** ensures that this particular C: partition is identified as "active" in the partition table. This is again a Windows requirement: You can only boot from the "active" (or "boot") partition.

An example GRUB configuration with two Windows XP configuration would be as follows:

```
title Windows XP partition 1
rootnoverify (hd0,1)
unhide (hd0,1)
hide (hd0,2)
makeactive
chainloader +1
title Windows XP partition 2
rootnoverify (hd0,2)
unhide (hd0,2)
hide (hd0,1)
makeactive
chainloader +1
```

## Partition addressing

As you can see, GRUB knows about file system structures, so it is legal to refer to a file as, for instance, /initrd. However, GRUB knows nothing about mount points, mounting of file systems or the /dev file system structure.

Instead, GRUB refers to partitions directly through stanzas like (hd0,0). This means: the first partition on the first hard disk. (In Linux-speak this would be /dev/hda1.)

You can specify a partition with the "root" stanza as in the examples above. Subsequent files within the block are then assumed to reside on this partition. But you can also supply a partition and filename as one stanza: "(hd0,0)/initrd" for example.

Note that the "root" stanza should refer to the /boot partition, as the subsequent files for GRUB are located there. It should NOT refer to the root file system of the operating system to be booted: That parameter is supplied as an option on the kernel line.

**Instructor notes:**

**Purpose** — Show students the GRUB configuration file and key options.

**Details** — Once you have the structure of this file covered, it's a nice mental exercise for the students to see to what extent they can reconstruct the partition table from this file.

The solution is as follows:

Primary partition 1 is a shared /boot file system.

Primary partition 2 contains the C: drive for Windows XP

Logical partition 1 (which is /dev/hda5, or (hd0,4)) contains the root file system for Red Hat

Logical partition 2 (/dev/hda6 or (hd0,5)) contains the root for Fedora

Logical partition 3 (/dev/hda7 or (hd0,6)) contains the root for SUSE SLES11

**Additional information —**

**Transition statement** — That covers GRUB. Let's see what happens once the kernel is loaded and executed.

## Starting the kernel

- Once the kernel is loaded, it is started by the boot loader.
- On most architectures, the kernel is compressed with a decompress program included.
- When the kernel starts, it detects all hardware and switches the CPU to multitasking, multiuser mode.

```
Inspecting /boot/System.map-2.6.16-rc1-git3-7-default
Loaded 21547 symbols from /boot/System.map-2.6.16-rc1-git3-7-default.
Symbols match kernel version 2.6.16.
No module symbols loaded - kernel modules not enabled.
klogd 1.4.1, log source = ksyslog started.
<5>Linux version 2.6.16-rc1-git3-7-default (geeko@buildhost) (gcc version 4.1.0
20060123 (prerelease) (SUSE Linux)) #1 Mon Jan 30 21:52:12 UTC 2006
<6>BIOS-provided physical RAM map:
<4> BIOS-e820: 0000000000000000 - 000000000009fc00 (usable)
<4> BIOS-e820: 000000000009fc00 - 00000000000a0000 (reserved)
<4> BIOS-e820: 000000000000e000 - 0000000000010000 (reserved)
<4> BIOS-e820: 0000000000010000 - 0000000000017ee06c0 (usable)
<4> BIOS-e820: 0000000000017ee06c0 - 0000000000017ee66c0 (ACPI data)
<4> BIOS-e820: 0000000000017ee66c0 - 0000000000017eee700 (ACPI NVS)
<4> BIOS-e820: 0000000000017eee700 - 0000000000018000000 (reserved)
<4> BIOS-e820: 0000000000fec00000 - 0000000000fec01000 (reserved)
<4> BIOS-e820: 0000000000fee00000 - 0000000000fee01000 (reserved)
<4> BIOS-e820: 0000000000fff80000 - 00000000100000000 (reserved)
<5>0MB HIGHMEM available.
<5>382MB LOWMEM available.
<6>found SMP MP-table at 0009fe00
<7>On node 0 totalpages: 98016
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-9. Starting the kernel

LX158.0

### Notes:

#### Introduction

When the user selects a Linux operating system in the boot loader, then the boot loader will load the Linux kernel.

#### Compressed versus uncompressed kernel images

A kernel image is either non-compressed (vmlinuz), or compressed (vmlinuz) and is normally located in the /boot directory. The naming convention for a kernel image that is compressed is that the kernel image file name will have the letter z. Uncompressed kernel images will end in the letter x.

#### Loading the kernel image

The boot loader loads a specified kernel image in memory and starts the kernel executing. At this point, the kernel initializes system hardware which has built-in support. This includes hard disks, serial devices, mice, graphical adapters, keyboards, network adapters, and so on. By far, most of these adapters can indeed be autodetected, but some cannot. In that case, their configuration parameters (most notably, IRQ, I/O, and DMA levels) need to be

passed to the kernel as boot options. If this is the case, consult the Hardware-HOWTO for details.

After the kernel has detected all hardware, it switches the processor to the so-called “protected mode,” which basically means that from that point on multitasking is possible in a multiuser environment.



### Note

While booting, the kernel generates a lot of messages that will scroll off the screen very fast. Since no file system is available on which to store these messages, they vanish.

If you wish to retrieve these messages later however, you can run the **dmesg** command to see them. The kernel stores its messages in the kernel ring buffer. The **dmesg** command is used to print messages in this buffer.

**Instructor notes:**

**Purpose** — Show the kernel startup flow.

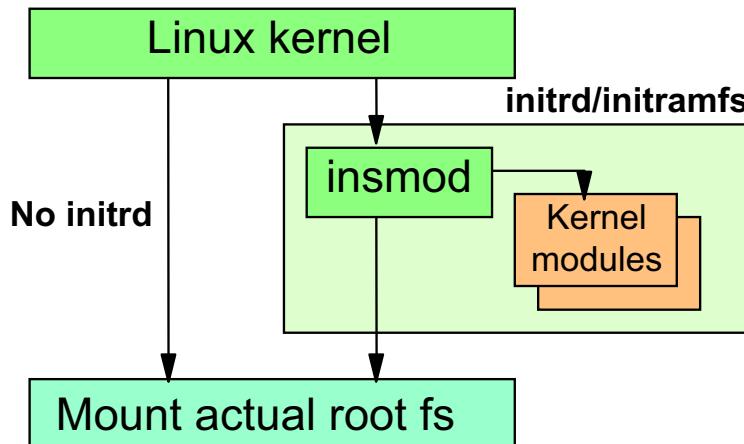
**Details** — Discuss the student notes.

**Additional information** —

**Transition statement** — Let's look at something called an *initrd* now.

## Initial RAM disk

- An Initial RAM Disk (initrd) or Initial RAM Filesystem (initramfs) is needed if the kernel cannot access the root file system without first loading the proper modules (such as SCSI, LVM, RAID, ext3, Reiser).
- The initrd/initramfs is a small gzip-compressed ext2 or cpio image.
- The initrd/initramfs is loaded into memory by the boot loader and decompressed by the kernel.



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-10. Initial RAM disk

LX158.0

### Notes:

#### Introduction

Not all hardware is supported in the core kernel image. In fact, almost all hardware support in Linux today comes in the form of modules. These modules are pieces of code that are loaded into kernel memory only if required.

This works well, but leads to a minor problem if kernel modules are needed to mount the root file system. This can happen, for instance, because:

- The root file system sits on a hard disk type for which support was not compiled into the kernel image. This applies mostly to SCSI.
- LVM or RAID was used, and LVM or RAID support was not compiled into the kernel image.
- The root file system uses a more recent file system type, and support for these file systems was not compiled into the kernel image.

In these cases, you are going to need an *initial RAM disk* (sometimes also called an *initial root disk*). This is a file containing a compressed image of an ext2 file system, which in turn contains two things:

- A **linuxrc** script
- The kernel modules that are needed

Newer kernels do not use an *initial RAM disk* but use a compressed cpio archive which is loaded onto a Linux RAM disk. There are some technical advantages to this. For instance, it is not required to have ext2 support in the core kernel image. You can recognize these newer kernels since the image is called **initramfs** instead of **initrd**.

The initrd or initramfs image is loaded into memory by the boot loader, just like the Linux kernel. When the Linux kernel starts, it detects the presence of the initrd/initramfs and will use it.

For an **initrd**, this works as follows: The initrd is uncompressed in memory and then mounted as temporary root. The kernel will then proceed to execute the **linuxrc** script.

The **linuxrc** script loads all the required modules, mounts the true root file system, and then executes a system call **pivot\_root**. This switches the position of the initrd and the true root file system. From that point on, the actual root file system is mounted at its correct location, and **linuxrc** is able to continue the boot process by starting the **/sbin/init** program.

For an **initramfs** image, the procedure is slightly different. The kernel creates a RAM disk (which is a standard kernel feature) and will unpack the initramfs onto this RAM disk. Part of this compressed image is an **init** program, which will be started. From that point on, it depends on the configuration of this **init** process. It is possible to run a full Linux system from this initramfs (which is another advantage of initramfs over initrd), but it is also possible that the code on the initramfs will locate and mount another root file system and execute a **pivot\_root** just like in the initrd.

***Instructor notes:***

**Purpose** — Discuss the initial ram disk (initrd).

**Details** — Discuss the student notes.

**Additional information** —

**Transition statement** — The next step thus is **init**.

## init

---

- **init** is started by the kernel after the root file system is mounted.
- **init** reads configuration file /etc/inittab.
- Decides on default runlevel if no runlevel is given.
- Runlevels have different meaning:
  - 0: System halt
  - S: Single-user mode (no scripts run) (SUSE)
  - 1: Single-user mode (some scripts run)
  - 2: Local multiuser without network
  - 3: Full multiuser with network
  - 4: Not used
  - 5: Full multiuser with network and xdm (GUI)
  - 6: System reboot
- **init** will start all programs for that runlevel.

**Note:** Once the system has started, you can switch runlevels with **init runlevel** or **telinit runlevel**.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 4-11. init

LX158.0

### Notes:

#### init process

The **init** process started by the kernel reads its configuration file /etc/inittab to:

- Identify the first script to run during system startup
- Identify the default run level if no runlevel is given at the boot prompt
- Determine which scripts to be run at the various run levels
- Determine how to handle certain key sequences
- Determine how to handle a power failure

#### Runlevels

There are seven (eight for SUSE) runlevels, but on most distributions only runlevel 3 and 5 are really important for us. 3 means full multiuser mode with a text-based login (you will need to start X yourself), and 5 is the same, but with an X-based login screen.

The following run levels apply to Red Hat, Fedora, and SUSE:

- 0 - System halt
- S - Single-user mode (no scripts run) (SUSE)
- 1 - Single-user mode (some scripts run)
- 2 - Local multiuser without network
- 3 - Full multiuser with network
- 4 - Not used
- 5 - Full multiuser with network and xdm (GUI)
- 6 - System reboot

### Default runlevel

The default runlevel is specified in the /etc/inittab file. The entry **id:** identifies the default runlevel. For example:

```
# grep id /etc/inittab
id:5:initdefault:
```

The default runlevel for the system in the example will be runlevel **5**.

### Run-level determination

The current system run level can be determined by using either of the following commands:

```
# who -r
run-level 5 Jun 29 09:29 last=S
# runlevel
N 5
```

### Switching runlevels

Switching runlevels can be accomplished by using either the **init** or **telinit** commands. For example, to change the runlevel of the system to runlevel **1**:

```
# runlevel
N 5
# init 1
# runlevel
5 1
```

In the example, by using the **runlevel** command, you can identify the current runlevel state before and after the **init** command was issued.

***Instructor notes:***

**Purpose** — Introduce the **init** command.

**Details** — Discuss the student notes.

**Additional information** —

**Transition statement** — Let's take a further look at the /etc/inittab file.

## /etc/inittab example

```

# Default runlevel
id:3:initdefault:

# System initialization.
si::bootwait:/etc/init.d/boot

10:0:wait:/etc/init.d/rc 0
11:1:wait:/etc/init.d/rc 1
12:2:wait:/etc/init.d/rc 2
13:3:wait:/etc/init.d/rc 3
#14:4:wait:/etc/init.d/rc 4
15:5:wait:/etc/init.d/rc 5
16:6:wait:/etc/init.d/rc 6

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -r -t4 now

# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty --noclear 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

```

- The default runlevel is 3.

Always run **/etc/init.d/boot**.

- Run **/etc/init.d/rc** with the runlevel as parameter.
- Trap the three-finger salute **Ctrl+Alt+Delete**.
- Allow users to log in on six virtual consoles. (Virtual consoles can be activated with **Alt+F1** through **Alt+F6**.)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-12. /etc/inittab example

LX158.0

### Notes:

#### Introduction

The visual shows the most important lines of the /etc/inittab file. Because there are minor differences in RHEL/Fedora and SLES /etc/inittab files, they are shown side by side.

#### Default runlevel

As mentioned earlier, the entry **id:** identifies the default runlevel unless it was specified during the boot process. In the example shown in the visual, the default runlevel is **3**.

#### System initialization

The second entry directs **init** to always run the **/etc/rc.d/rc.sysinit** (RHEL/Fedora) or **/etc/init.d/boot** (SLES) script. This script does a number of important low-level tasks, such as:

- Activating swap spaces
- Setting the hostname
- Checking the root file system for errors, and remounting it read-write

- Turning on quota support
- Loading important kernel modules
- Checking all other file systems and mounting them
- Deleting various lockfiles which might have been left over from a crash
- Enabling the clock

## Defined runlevels

The next set of lines tells **init** to run the **/etc/rc.d/rc** or **/etc/init.d/rc** in runlevels 0 through 6, with the runlevel as a parameter. We will look at this script in the next visual.

## Key sequence trap

After that, the trap for the **Ctrl+Alt+Delete** three-finger salute is set. This means that if you press this key combination, the **shutdown** command is executed, effectively rebooting your system.

## Terminal gettys

Finally, six gettys are started on tty1 through tty6. This means that there are six virtual terminals configured, allowing you to log in as different users six times. These six virtual terminals can be reached by pressing **Alt+F1** through **Alt+F6**.

On a Red Hat system in runlevel 5, only the **mingetty** processes for tty2 through tty6 will be started, leaving tty1 free for your graphical environment.

**Note:** Some commands have the prefix **once**, some have **wait** as prefix, and others have **respawn**. This identifies what **init** should do after it has started the command:

- **wait** means that **init** should wait for the command to finish before it is allowed to go on with the rest of the init sequence.
- **once** means that **init** is allowed to go on with the init process even before the command has finished.
- **respawn** means that **init** should start this process, put it in the background, and monitor its existence. Once the process dies, **init** should start a new one. This is commonly used for login processes because a new login screen will then automatically appear, even if the user manages to kill off all its processes.

***Instructor notes:***

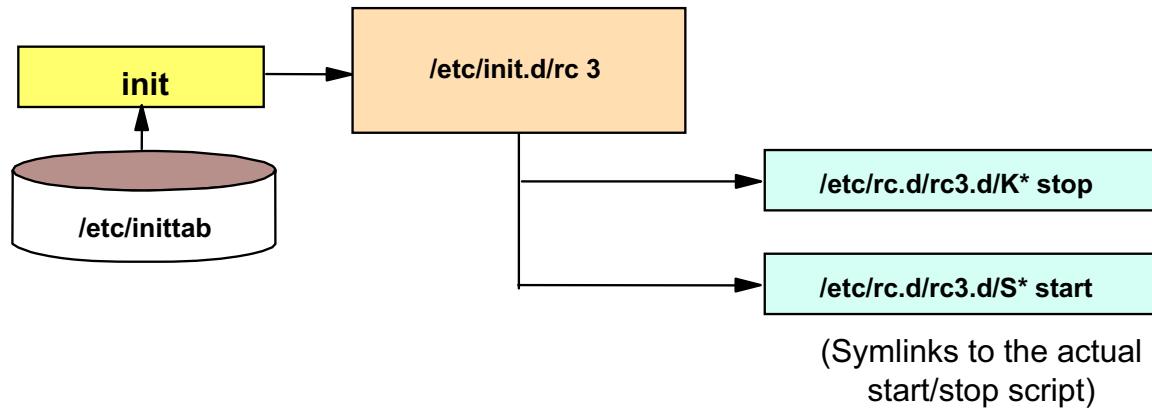
**Purpose** — Show the /etc/inittab file.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Let's look at that **rc** script because that one has a lot of work to do.

# Starting services: System V init style



```

# ls -l /etc/rc.d/rc3.d
lrwxrwxrwx 1 root root 24 Mar 15 10:47 K02NetworkManager -> ../../init.d/NetworkManager
lrwxrwxrwx 1 root root 14 Mar 15 11:45 K05innd -> ../../init.d/innd
lrwxrwxrwx 1 root root 19 Mar 15 10:45 K05saslauthd ->
../../init.d/saslauthd
...
lrwxrwxrwx 1 root root 15 Mar 15 10:48 K15httpd -> ../../init.d/httpd
lrwxrwxrwx 1 root root 15 Mar 15 11:45 K16rarpd -> ../../init.d/rarpd
...
  
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-13. Starting services: System V init style

LX158.0

## Notes:

### Introduction

The **rc** script is a very important script. Although small, it is responsible for starting almost all services that are active in the runlevel that was specified as parameter.

What this script basically does is the following:

- It changes to the directory `/etc/rc.d/rc<runlevel>.d`<sup>1</sup>
- In this directory, it makes a list of all scripts that start with a K, sorts this list on the two digits after the K, and executes these scripts with the **stop** parameter.<sup>2</sup>
- Then, it makes a list of all scripts that start with an S, sorts it, and executes them with the **start** parameter.

These scripts are in fact not scripts at all, but are symbolic links to generic scripts in `/etc/rc.d/init.d` or `/etc/init.d`<sup>3</sup>. Every server program that is installed on a Linux system is

<sup>1</sup> This directory is a symlink to `/etc/init.d/rc<runlevel>.d` in SLES.

<sup>2</sup> Obviously, kill scripts are not relevant when booting straight into a runlevel. It is possible, however, to change runlevels in a live system by running the command `init <new runlevel>`. In that case, it might be necessary to stop services, for instance when switching from a multiuser to a single-user runlevel.

supposed to have a corresponding control script in this directory, with the same name as that service. By making a symbolic link from /etc/rc.d/rc3.d to that particular script, the administrator ensures that a particular service is started (or stopped) in a certain runlevel. And by specifying a two-digit number after the S or K, the administrator can even influence the order in which services are started and stopped.

For example, when entering run level 2 (Full multiuser without network) two scripts that are executed on a Red Hat installation are:

- **K15httpd** - kills http related processes
- **S90crond** - starts the cron process

This scheme was first used in AT&T's System V (five) UNIX. That is why it is called the System V init style. It is used, among others, by Red Hat and SLES. Other Linux distributions might use other init styles. However, for all distributions, the principle holds: **init** reads the /etc/inittab files and starts all the programs that are listed there. There is never a magic or secret program or script being started. That means that it does not really matter which distribution you use. Take a look at the /etc/inittab file and read the scripts that are listed here. This will tell you how the system is started.

<sup>3</sup> Depends on the distribution used.

### **Instructor notes:**

**Purpose** — Explain the setup of the /etc/rc.d directory.

**Details** — Discuss the student notes.

**Additional information** — Note that in the visual only runlevel 3 is specified. Of course, the same is true for every other runlevel as well.

**Transition statement** — **init** has been with us a long time, but it will be replaced soon by something else. Let's take a quick look at that.

## Starting services: Upstart

- Replacement for init in RHEL 6.x
  - Will likely be replaced by **systemd** in RHEL 7
- Asynchronously, can run tasks in parallel
- Configuration directory /etc/init
- Event-based
  - Can start/stop services when events occur (such as inserting a USB key)
  - Can also be used as a **cron/at** replacement
- Backwards compatible with System V init
  - Can run all System V init scripts
  - Uses the same symlink structure as System V init (using /etc/init/rc.conf)
  - Uses /etc/inittab only to determine default runlevel

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-14. Starting services: Upstart

LX158.0

### Notes:

**init** has been the workhorse of UNIX system initialization for a long time, but it has a number of drawbacks:

- It is rather static. Other than playing with the runlevels (which is rarely done) you don't have a lot of opportunity to react to different circumstances. For instance, if hot-pluggable devices (USB for instance) are inserted, you might want to start services "on demand".
- **init** runs all its processes essentially sequentially. It does not really support running processes in parallel. However, running processes in parallel can greatly speed up the boot time of a system.
- **init** itself is a rather simple program, and relies heavily on shell scripts to get other services started. The sysinit script, for instance, is between 350 and 850 lines alone (depending on the distribution), and may call several other shell scripts as child processes. Shell scripts are notoriously slow, since they depend on a large number of external programs to function.

For these reasons, people have been working on successors for **init**. The first viable successor was **upstart**. At the moment, **upstart** is used in RHEL 6.x.

Upstart works by reading the /etc/inittab file for the default runlevel (if no runlevel was entered on the command line), and then it retrieves a number of configuration files from the /etc/init directory. Each of these configuration files contains a list of tasks to perform (such as scripts to run), and the runlevel for which they are required.

**upstart** is able to run tasks in parallel, thus speeding up the startup process. In theory, **upstart** is also able to react to events such as the insertion of USB keys, and even to start time-based events similar to **cron** and **at**, but in practice that functionality is rarely used.

Distributions that use **upstart**, particularly Red Hat, have gone through great pains to make sure that **upstart** is downwards compatible with System V init style scripts. This is mostly done through two **upstart** configuration files, /etc/init/rcS.conf and /etc/init/rc.conf. These two configuration files ensure that /etc/rc.sysinit and the /etc/rc<runlevel>.d scripts are started.

***Instructor notes:***

**Purpose** — Discuss upstart

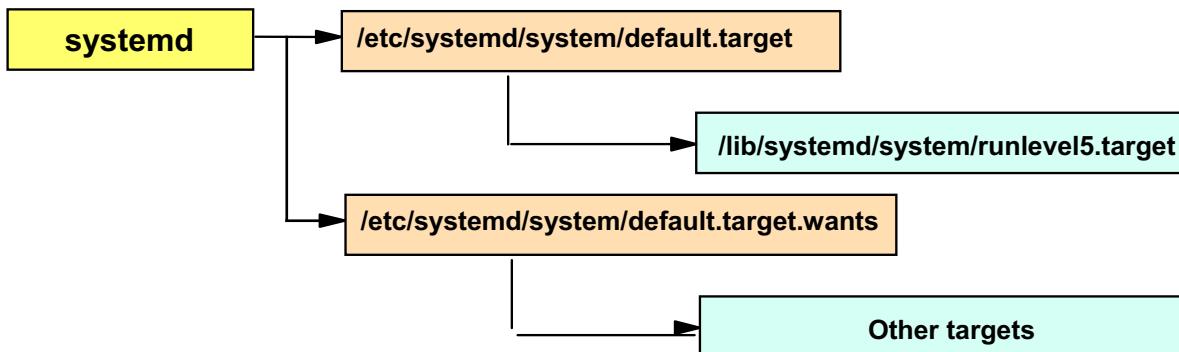
**Details** —

**Additional information** —

**Transition statement** — upstart will not be around much longer. The successor is systemd.

## Starting services (systemd)

- Upcoming replacement for init in RHEL and SLES
  - Already present in Fedora 15 and OpenSUSE 12.1
- Works by defining "targets" which have other targets as dependencies
- Far less reliant on shell scripts
- Can start tasks in parallel and "on demand"
- Downwards compatible with System V init style scripts



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-15. Starting services (systemd)

LX158.0

### Notes:

At the moment it looks like the enterprise distributions will, in their next version, start to use **systemd** instead of **init** or **upstart**. Fedora 15 and OpenSUSE 12.1 are already using **systemd**, so it is very likely that it will also be used in RHEL 7 and SLES 12.

We are not going to cover **systemd** in full in this course. We only want to cover the principles.

**Systemd** works by defining "targets". A target is something that **systemd** needs to achieve in certain circumstances. This may be the starting or stopping of a service, but also, for instance, the mounting or unmounting of a specific device/file system.

Each target has a description in **/lib/systemd**, and targets may be dependent on other targets.

By creating strategic symbolic links from the **/etc/systemd** directory to the **/lib/systemd** files (through the use of proper tools, obviously) you can modify the set of targets that the system will try to achieve.

Systemd will construct a tree of targets based on the main target of, for instance, "runlevel5.target". All dependencies are noted, and all targets will be activated, in the proper order, until eventually the "runlevel5.target" is achieved. Targets that do not rely on each other can possibly be executed in parallel as well.

Targets can be identified during system boot, but targets can also be activated as a reaction to an external event, such as the plugging in of USB storage.

And to speed up things even more, targets can also start deferred. For instance for a networking service, it means that systemd will open the network socket itself, wait for traffic to arrive on this socket, and will only start the service once there is indeed traffic arriving<sup>1</sup>.

Implementing systemd is a huge task, and particularly third-party software may not be fully systemd-compatible from the start. For this reason, a lot of effort has gone into making systemd fully System V compatible. This means that you can still use the /etc/init.d/\* scripts with the start/stop parameters, that the management tools we're going to discuss in the next few visuals still work and so forth.

<sup>1</sup> This can be thought of as a replacement for `xinetd`.

### **Instructor notes:**

**Purpose** — Introduce **systemd**.

**Details** — Make sure to stress that **systemd** is NOT found in the enterprise distributions that are on the market today (RHEL6 and SLES11) but will most likely be found in the next version.

**Additional information** —

**Transition statement** — Okay, let's see how we manage things.

# Configuring services per runlevel

- Use **chkconfig** to create the appropriate links for each service.

```
# chkconfig --list
...
acpid      0:off  1:off  2:off  3:off  4:off  5:off  6:off
atd        0:off  1:off  2:on   3:on   4:off  5:on   6:off
...
# chkconfig acpid on
# chkconfig --list
...
acpid      0:off  1:off  2:on   3:on   4:off  5:on   6:off
atd        0:off  1:off  2:on   3:on   4:off  5:on   6:off
...
```

- In a **systemd** environment, **chkconfig** is a wrapper for **systemctl**.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-16. Configuring services per runlevel

LX158.0

## Notes:

### Introduction

The system runlevel determines what processes to be active at any given time. As the system enters a runlevel, **init** will start or stop processes defined by symbolic links to the associated runlevel rc directory structure. However, managing these scripts by hand is really tedious. That is why several tools exist for this:

- **chkconfig** command
- **system-config-services** (RHEL/Fedora)
- **yast** (SLES)

### Chkconfig command

**chkconfig** provides a simple command-line tool for maintaining the /etc/rc[0-6].d directory hierarchy by relieving system administrators of the task of directly manipulating the numerous symbolic links in those directories.

**chkconfig** has five distinct functions: adding new services for management, removing services from management, listing the current startup information for services, changing the startup information for services, and checking the startup state of a particular service.

```
chkconfig --list [name]  
chkconfig --add name  
chkconfig --del name  
chkconfig [--level levels] name <on|off|reset>  
chkconfig [--level levels] name
```

The visual above shows various operational examples of the **chkconfig** command in use.



**Note**

The **chkconfig** command only maintains the links for services it does not start or stop them.

**Instructor notes:**

**Purpose** — Discuss the way services are configured.

**Details** — Discuss the student notes.

**Additional information —**

**Transition statement** — These scripts in the init.d directory can be used manually as well.

## Starting and stopping services manually

- Scripts in init.d directory can be used to start/stop services manually
  - On RHEL/Fedora, the service command calls this script
  - On SLES, rc service is a symlink to the init.d script
- Default options: **start, stop, status, restart**
- Other options might also be available

```
RHEL/Fedora # service atd restart
Stopping atd: [ OK ]
Starting atd: [ OK ]
```

```
SLES # rcatd restart
Shutting down service at daemon [ done ]
Starting service at daemon [ done ]
```

- In a **systemd** environment, **service** is a wrapper for **systemctl**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-17. Starting and stopping services manually

LX158.0

### Notes:

#### Introduction

The scripts in the init.d directory can perfectly be used to start and stop individual services manually, for instance, after changing configuration files. All scripts will always accept the **status**, **start**, **stop**, and **restart** parameters. In addition to that, some scripts will also accept other parameters, like **reload** (only reread the database without restarting the server).

You can call the script directly using its full pathname<sup>1</sup>, but that requires typing a lot of slashes and dots. Most distributions, therefore, have created some sort of shortcut which is faster to type:

- On a Red Hat or Fedora system, you can also use the **service** command. This does nothing more than calling the script for you with the parameters you specified.
- On a SLES system, a symbolic link with the name **rc<service>** is automatically created. This links to the **init.d** script.

<sup>1</sup> The init.d directory is not in your \$PATH, and for good reason: The scripts sometimes have the same name as the daemon itself.

---

For example, to restart the **atd** daemon:

**# service atd restart** (RHEL/Fedora)

**# rcatd restart** (SLES)

**Instructor notes:**

**Purpose** — Show that the scripts in init.d can also be used manually.

**Details** — Discuss the student notes.

**Additional information —**

**Transition statement** — So far, we have covered the ordinary Linux startup flow. Now let's look at a special case.

## Booting Linux in single-user mode

- Single-user mode
  - No networking (so no incoming hackers)
  - No services being started
  - No root password required (RHEL/Fedora)
- Very useful for system maintenance
- To start from GRUB: Add single to the kernel line of the corresponding menu entry

GRUB

[Minimal BASH-like line editing is supported. For the first word, TAB lists possible command completions. Anywhere else TAB lists the possible completion of a device/filename. ESC at any time cancels. ENTER at any time accepts your change.]

```
grub append> ro root=/dev/VolGroup00/LogVol00 single
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-18. Booting Linux in single-user mode

LX158.0

### Notes:

#### Introduction

Sometimes it is necessary to have full control over your system, with no users or other programs doing all kinds of unexpected things. This is possible in Linux and is called single-user mode.

For single-user mode, you will need to specify the **single** option to the kernel when your system boots. The Linux kernel will then boot as normal, but **init** will only run **/etc/rc.d/rc.sysinit** or **/etc/init.d/boot** and then start a **bash** shell. It will not start all the normal services, so users cannot log in over the network.

On a RHEL/Fedora system, the single-user mode will not even ask for a root password. This is done so that it can be used if you forgot your root password and need to set a new one.

## Entering single-user mode with GRUB

To enter single-user mode on a system configured to use the GRUB boot loader, interrupt the boot process by hitting the space bar. Once interrupted, use the **a** key to append the option **single** to the kernel parameter line.

## Exiting single-user mode

Once you have completed the system maintenance activity in single-user mode, exit single-user mode by using one of the following commands:

```
# init <runlevel>  
# shutdown -r now
```



### Note

The safest course of action is to do a full reboot of the system using the **shutdown** command. This will cause the system to go through the normal boot sequence and execute the required scripts.

**Instructor notes:**

**Purpose** — Explain single-user mode.

**Details** — I always ask the students which method is more secure: Red Hat's method of not asking for a root password or the SLES method of requiring the root password. Students always claim that SLES method is more secure, and then I tell them about the boot option **init=/bin/bash**, which allows me to get into a SLES system anyway. It is a little more complicated (you need to do a **remount,rw** of the root file system, mount /proc manually, and so forth) but the net effect is the same. Moral of the story? If you have access to the console, you can break into the system and change the root password, and nobody can stop you. So physical security is more important than protecting single-user mode with a root password.

**Additional information** — There is a slight difference between runlevel s (which is started when a user types s or *single* on the boot prompt) and runlevel 1: In runlevel 1 the scripts in /etc/rc.d/rc1.d are run. Typically, the only script activated from here is the script that sets the keyboard mapping. Thus, for users of US/English keyboards, there is no actual difference. Users of other keyboards will probably want to use runlevel 1, and not runlevel s.

**Transition statement** — Oops, unknowingly we have already talked about shutting down a Linux system twice now. Let's just formalize that, and then we are finished.

## Shutting down a Linux system

---

- Do not switch power off to shut down
- Use **shutdown** command or **Ctrl+Alt+Delete**
  - Warns users
  - Stops all running processes
  - Unmounts file systems
  - Does an orderly shutdown
  - Reboots if necessary
- Example:
  - To reboot: `shutdown -r now, reboot, or init 6`
  - To halt: `shutdown -h now, halt, init 0, or poweroff`
- Some display managers allow a user to perform a shutdown as well

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-19. Shutting down a Linux system

LX158.0

### Notes:

#### Introduction

If you need to shut down a Linux system, do not just pull the plug, but ensure that somehow the **shutdown** command runs. In fact, we have already seen how to do that: by pressing **Ctrl+Alt+Delete** on the console, which was trapped in /etc/inittab, or by entering the command itself on the command line. Other alternatives are the commands **reboot**, **halt** and **poweroff**.

Some graphical window managers and display managers allow the console user to perform a shutdown as well. This seems like a security exposure, but think of this: the console user can just as easily yank the power cord if he wants to do a shutdown. Allowing him to do a proper shutdown is probably a better way of doing things.

**Instructor notes:**

**Purpose** — Cover the **shutdown** command.

**Details** — Discuss the student notes.

**Additional information —**

**Transition statement** — Let's pause for a moment and answer some checkpoint questions.

## Checkpoint

---

1. Name the four steps that form the startup order of a Linux system.
  
2. How would you select a graphical login screen (**xdm**, **kdm**, or **gdm**)?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 4-20. Checkpoint

LX158.0

### Notes:

**Instructor notes:**

**Purpose** — Check student's knowledge of presented materials.

**Details** —

## Checkpoint solutions

1. Name the four steps that form the startup order of a Linux system.

The answers are firmware, boot loader, kernel, and init.

2. How would you select a graphical login screen (**xdm**, **kdm**, or **gdm**)?

The answer is by setting runlevel 5 as the default runlevel in /etc/inittab.

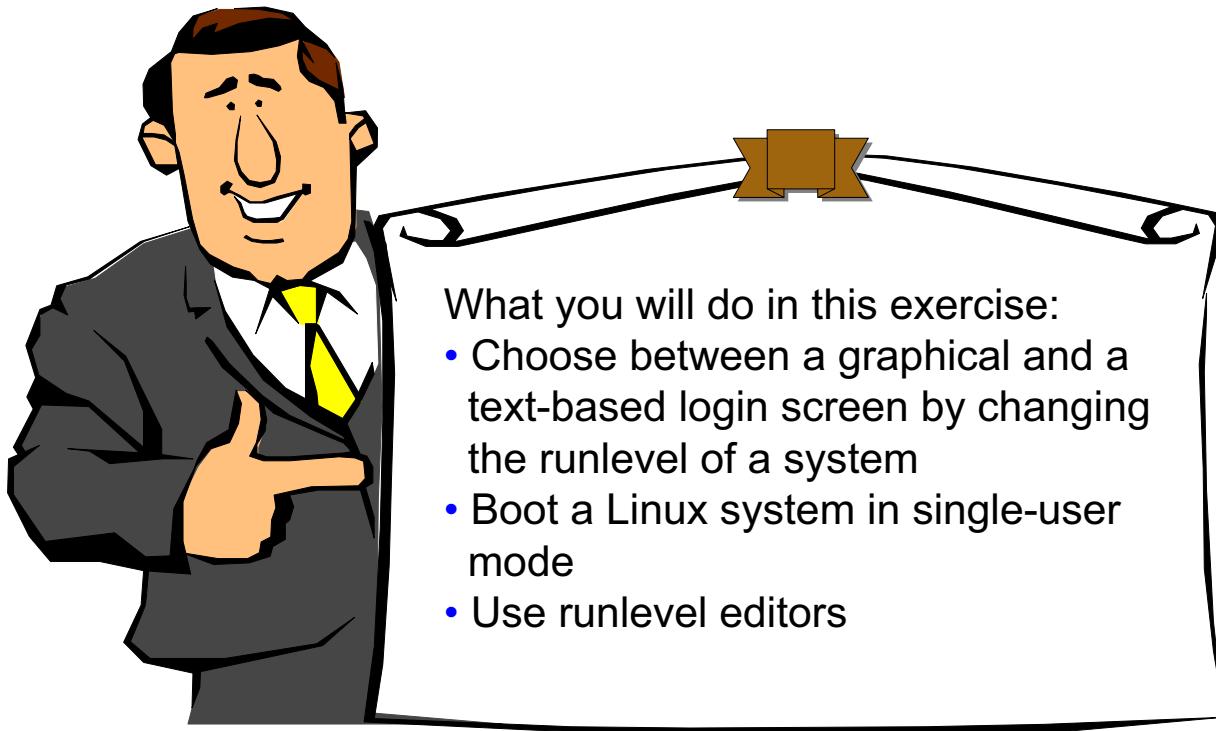
© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information** —

**Transition statement** — Let's use what we have learned in a class exercise.

## Exercise: Startup and shutdown

---



- What you will do in this exercise:
- Choose between a graphical and a text-based login screen by changing the runlevel of a system
  - Boot a Linux system in single-user mode
  - Use runlevel editors

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 4-21. Exercise: Startup and shutdown

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Unit summary

---

Having completed this unit, you should be able to:

- Describe the Linux startup flow
- Configure autostarting services
- Boot Linux in single-user mode
- Perform a proper shutdown of a Linux system

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 4-22. Unit summary

LX158.0

### Notes:

Having completed this unit, you should understand:

- The Linux startup flow is as follows:
- When power is switched on, the firmware is loaded.
- Firmware loads the boot loader and executes it.
- The boot loader presents a menu of OSs to boot
- The boot loader loads the OS initial code. In case of Linux, this is a kernel and
- The first process started by the kernel is init.
- init starts the rest of the processes.
- RHEL6 uses upstart instead of init
- In the future, systemd will replace init and upstart
- Booting in single-user mode is done by adding the "single" keyword to the boot-line in GRUB

- Shutting down a Linux system is done with the shutdown command or with Ctrl+Alt+Delete

***Instructor notes:***

**Purpose** — Summarize unit material.

**Details** —

**Additional information** —

**Transition statement** — Now, we can move on to the next unit.

# Unit 5. System administration tools

## Estimated time

00:30

## What this unit is about

This unit gives you an overview of the different integrated system administration tools that might be available on your distribution.

## What you should be able to do

After completing this unit, you should be able to:

- Discuss the main characteristics of system administration tools
- List some distribution-specific administration tools
- List some general purpose administration tools

## How you will check your progress

Accountability:

- Checkpoint questions
- Machine exercises

## Unit objectives

---

After completing this unit, you should be able to:

- Discuss the main characteristics of system administration tools
- List some distribution-specific administration tools
- List some general purpose administration tools

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 5-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

# System administration tools

- Integrated tools for system management
- Allow you to make configuration changes throughout the system from within one tool
- Multiple interfaces possible:
  - Text-based
  - X-based
  - Web-based
- To decide on a tool to use, consider:
  - Type of interface required
  - Distribution-specific or generic?
  - Only base system configuration or application configuration too?
  - Can the tool be extended easily?
- Does the perfect tool exist yet?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 5-2. System administration tools

LX158.0

## Notes:

### Introduction

System administration tools provide the system administrator with the means to easily perform tasks/operations on the system. Without the use of tools, system administration tasks would require a number of manual steps such as:

- Editing configuration files
- Starting/stopping services
- Running commands

**Note:** Without such tools, the chances of missing a crucial step are increased, and therefore the use of such tools are highly recommended. For example, adding a user to the system requires a number of steps:

- Adding the user name (**useradd** command)
- Adding the user to various groups (**useradd** or **usermod** command)
- Setting the user's password (**passwd** command)

System administration tools typically use one or more different interfaces, based on the way you connect to them. Typical choices include:

- Text-based: The tool typically uses the curses library to present a menu-driven interface in a text-based terminal. This is typically used when logged in through a text console or through a telnet or ssh session.
- X-based: The tool typically uses some X library to present a graphical interface. This can only be used in an X-based environment.
- Web-based: The tool typically listens on a Transmission Control Protocol (TCP) port for HTTP traffic. The menu screens themselves are generated using HTML. This requires you to use a browser which connects to the right port.

## System administration tool availability

The landscape of system administration tools is constantly changing. There is a number of reasons for this:

- Writing a system administration tool is a good project for graduate students.
- Currently, there is no authoritative configuration frameworks on the market which allow and encourage software developers to write their management tools using that framework. That means that the tool developers have to write the menu screens that allow you to manage various applications, such as Apache, Samba and so forth.

This costs a lot of effort and the past has shown that it virtually impossible to keep up with changes in the applications if you are not part of the project yourself.

To understand this better, consider the **man** tool. This has become the de facto tool for manual pages. Every software developer can write manual pages and have them automatically included in the set of manual pages that already exist on a system (simply by copying them to /usr/share/man). The developers of the **man** command themselves therefore do not have to write the manual pages for all commands anymore, except the manual page for the **man** command itself.

- When a distribution makes a change to, for instance, the way an IP address of an interface is stored on disk, the tool needs to develop too.

Since distribution manufacturers will want the tools to be available when the distribution is released, they typically will write their own tools that are able to perform base system configuration on their distribution. These tools change from one version to the next, tracking closely the configuration setup from the distribution.

All this means that the perfect tool does not yet exist. You therefore have to decide for yourself whether to use these tools at all, or do all configuration by hand. Also, if you decide to use a tool, you need to decide for which tasks you are going to use it and what interface you are going to use.

Another consideration in a large installation might be whether the tool is easily extendible so that menu screens which control your own, locally developed applications can be added to the tool.

***Instructor notes:***

**Purpose** — Introduce system administration tools.

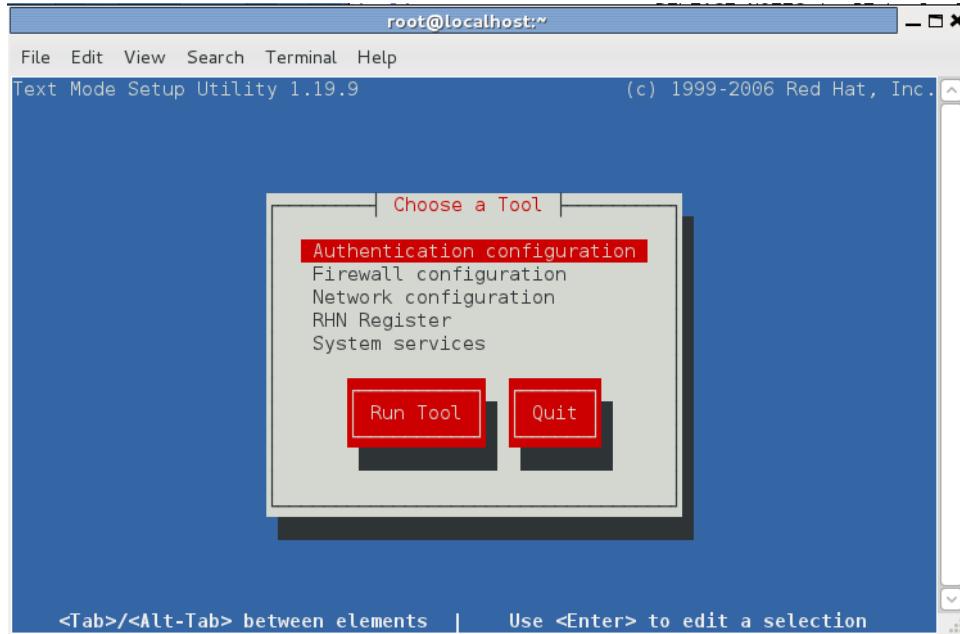
**Details** — Discuss with students the various types of system administration tools that are available.

**Additional information —**

**Transition statement** — Let's look at a few distribution-specific tools first and start off with Red Hat/Fedora.

## RHEL setup

- Menu-based front end for various tools that are part of the text-based installation



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 5-3. RHEL setup

LX158.0

### Notes:

#### Introduction

The command setup is Red Hat's/Fedora's text mode menu system that allows you to start the various text mode configuration programs. The following shows the tools available with version 1.18.1 of the setup tool:

**Authentication configuration:** Configures authentication services. This menu option calls the text menu command:

**/usr/share/authconfig/authconfig-tui.py**

An alternative command is the command line interface command:

**/usr/sbin/authconfig**

**Firewall configuration:** Configures the network firewall (implemented with iptables). This menu option calls the text menu command:

**/usr/bin/system-config-securitylevel-tui**

An alternative command is the command line interface command:

**/sbin/iptables**

**Network configuration:** This menu selection configures the network. It calls the text menu command:

**/usr/sbin/netconfig**

**RHN Register:** This menu allows you to register with the Red Hat Network. It calls the text menu command:

**/usr/sbin/rhn\_register**

**System services:** Configures runlevel services. This menu selection calls the text menu command:

**/usr/sbin/ntsysv**

**Note:** All these tools can also be started directly from the command line. Depending on the additional tools installed, there may be more menu options showing.

**Instructor notes:**

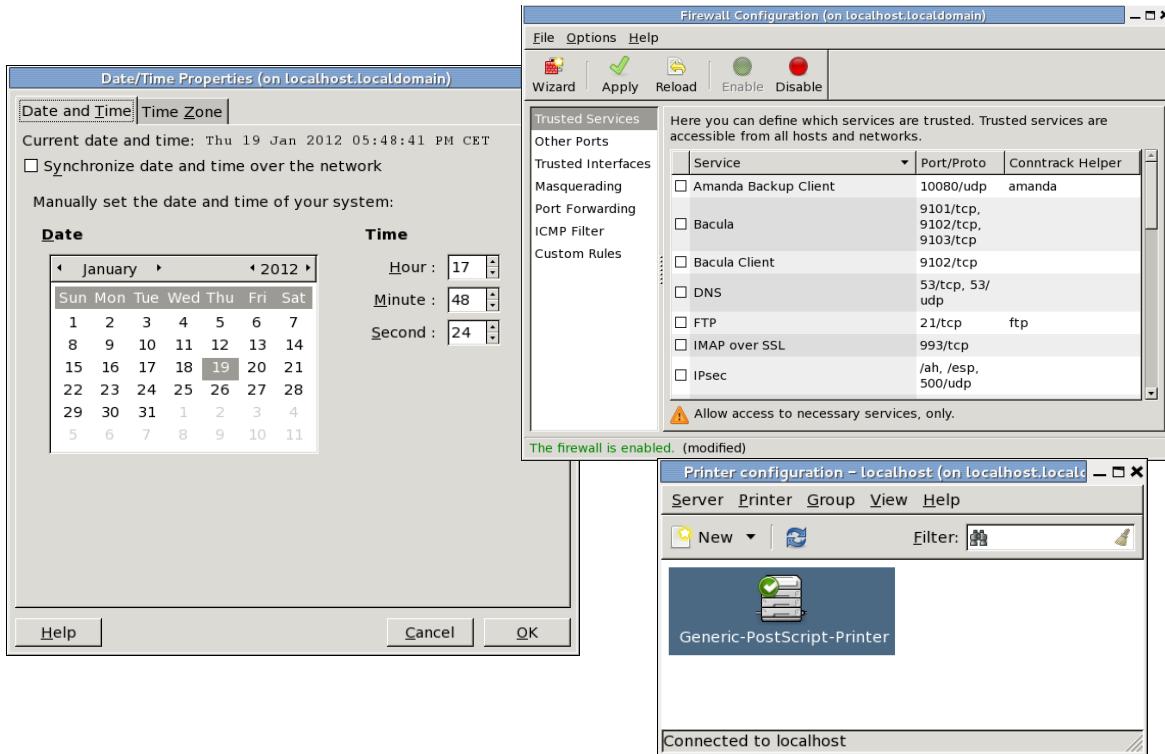
**Purpose** — Discuss RHEL's/Fedora's setup tool.

**Details** — Discuss with students that the setup tool is a front-end text mode menu system that calls the various commands listed in the table.

**Additional information —**

**Transition statement** — Next, let's look at the GUI tools that are available on RHEL/Fedora.

# RHEL system-config-\*



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 5-4. RHEL system-config-\*

LX158.0

## Notes:

### Introduction

Both Red Hat and Fedora distributions provide a set of Graphical User Interface (GUI) tools to be used for various system administration tasks. Each tool is a separate program and starts with system-config. For example, the visual shows screen shots of the tools system-config-date, system-config-firewall, and system-config-printer.

The following tools exist in RHEL and Fedora:

**system-config-authentication:** Configuration of system authentication resources

**system-config-date:** Local time, timezone and time server configuration

**system-config-firewall:** Configuration of the kernel firewall (iptables)

**system-config-keyboard:** Local keyboard configuration

**system-config-lvm:** Logical Volume Management configuration

**system-config-printer:** Printer configuration

**system-config-services:** System V services configuration

**system-config-users:** User and group management



**Note**

Tools must be run in an X-based environment. In addition, there is no front-end (like setup) to integrate these tools. Instead, they are integrated in the K Desktop Environment (KDE) and GNOME Start button menus.

**Instructor notes:**

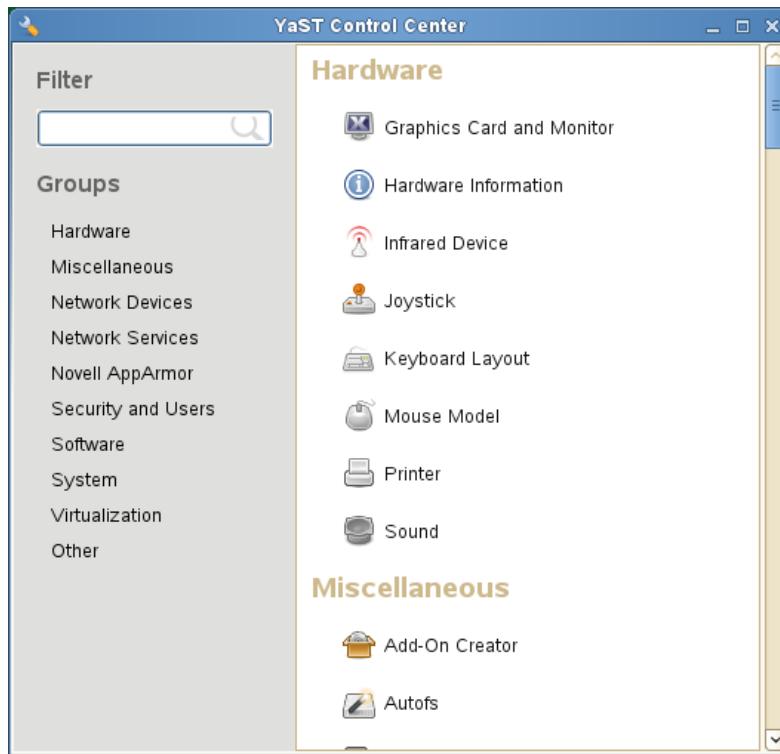
**Purpose** — Introduce students to the various GUI system administration tools found on RHEL/Fedora.

**Details** — Discuss with students the various system-config tools available on RHEL/Fedora.

**Additional information** —

**Transition statement** — Next, let's look at what SUSE provides.

# SUSE YaST



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 5-5. SUSE YaST

LX158.0

## Notes:

### Introduction

SUSE has provided Yet another Setup Tool (YaST) as a GUI interface/text menu tool to be used for various system administration tasks. It is comprised of a number of configuration modules that enable the system administrator to manage a particular aspect of the system.

Configuration modules are grouped into areas of administration. For example, the visual shows an icon on the left hand navigation window named Software. By clicking on the **Software** icon, the right hand contents pane will be populated with control modules related to software administration activities, including **Online update**, **Software management**, **System update**, and so forth.

### Starting YaST

YaST can be started either in a GUI interface or text menu mode.

### **Instructor notes:**

**Purpose** — Introduce students to the YaST tool provided by SUSE.

**Details** — Discuss with students that YaST is a single interface that provides access to configuration modules with which to manage the system. As it is discussed here, it is a distribution-specific tool.

**Additional information** — Novell has released YaST under the GPL license to enable its use on other distributions, and additional control modules are to be added.

**Transition statement** — Both the system-config and yast tools are distribution-dependent. A distribution-independent tool is getting increasingly popular as well. Let's take a look.

## Webmin

- <http://www.webmin.com>
- Open Source initiative to create an independent configuration framework
- Berkeley Software Distribution (BSD) Open Source License
- Modules to configure specific items
  - Modules can be created by anybody, using any license
- Support for all major UNIX versions, not just Linux
- Web-based interface only
- Not installed on all distributions by default
  - Might need to install yourself



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 5-6. Webmin

LX158.0

### Notes:

#### Introduction

Webmin is a Web-based interface for system administration for UNIX/Linux. It is designed from the ground up as an open-source, cross-platform system administration framework. This means that it does not include the actual administration tools itself but is only a series of perl scripts that allow people to write administration modules for various operating systems and administration tasks. The default Webmin distribution comes with a large number of administration modules, though.

Webmin is licensed according to the BSD Open Source license, but modules might be licensed with other licenses, such as the GPL.

Webmin comes as an RPM, which you need to install yourself. Installing RPMs is covered in a later unit.

***Instructor notes:***

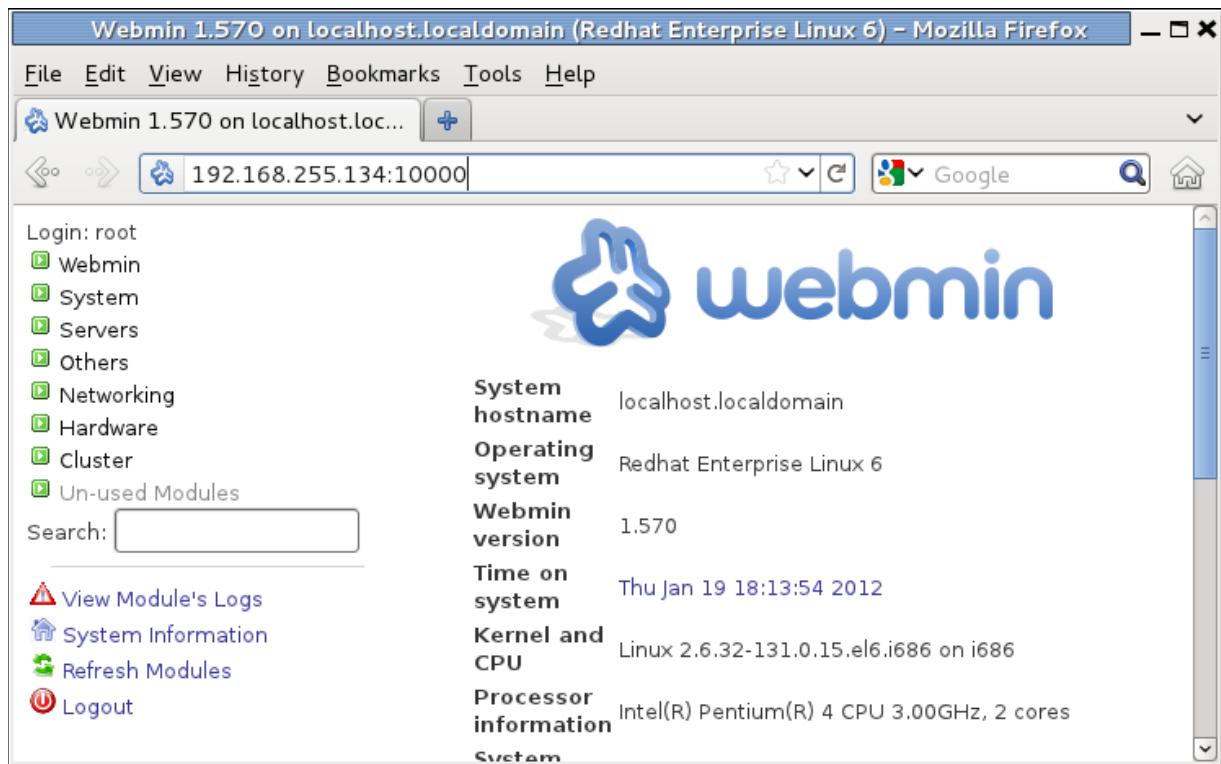
**Purpose** — Introduce Webmin as a cross-platform system administration framework.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Let's take a look at what Webmin looks like.

## Webmin screenshot



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 5-7. Webmin screenshot

LX158.0

### Notes:

#### Introduction

This is an example screenshot of Webmin.

**Instructor notes:**

**Purpose** — Show a screenshot of Webmin.

**Details** —

**Additional information** —

**Transition statement** — Obviously, there are more administration tools available (both distribution and cross-platform). Let's take a look at a few other popular ones.

# Other system administration tools

- **Nagios**
  - IT Infrastructure Monitoring
  - <http://www.nagios.org>
- **Ganglia**
  - Distributed Monitoring for clusters and grids
  - <http://ganglia.sourceforge.net>
- **Puppet**
  - Configuration Management
  - <http://www.puppetlabs.com/>
- **Spacewalk**
  - Software content management and updates
  - <http://spacewalk.redhat.com>
- **100s of others...**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 5-8. Other system administration tools

LX158.0

## Notes:

There are 100s of other system administration tools. The visual lists a few of the more popular ones.

**Nagios** is an open source IT Infrastructure Monitoring tool. It is able to monitor various performance characteristics from all sorts of devices and operating systems, and present them on a central console. This allows you to keep track of your complete IT infrastructure from a central location.

**Ganglia** is also an open source monitoring tool like Nagios, but more aimed at homogeneous clusters and grids.

**Puppet** is a configuration management tool. Instead of specifying what action to take, you specify what a certain machine is supposed to be configured like. Puppet will then execute the necessary actions to achieve that configuration.

**Spacewalk** is an open source alternative to the Red Hat Network. It allows software content management, including installing, updating and removing RPMs, across a large number of machines from a central location.

**Instructor notes:**

**Purpose** — List a few other tools that may be interesting.

**Details** — This is a good time to ask the students what they are using in their environment.

**Additional information —**

**Transition statement** — Okay, that's it. Time for the checkpoint questions.

## Checkpoint

1. The RHEL \_\_\_\_\_ tool provides a menu-based interface for various tools used during a text-based installation.
  
2. True or false: RHEL provides separate tools that start with system-config to administer the system with a GUI interface.
  
3. SUSE provides a tool called \_\_\_\_\_ as a GUI interface/text menu tool to be used for various system administration tasks.
  
4. What is the default port number to connect with the Webmin administration tool using a Web browser?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 5-9. Checkpoint

LX158.0

### Notes:

**Instructor notes:**

Purpose —

Details —

## Checkpoint solutions

---

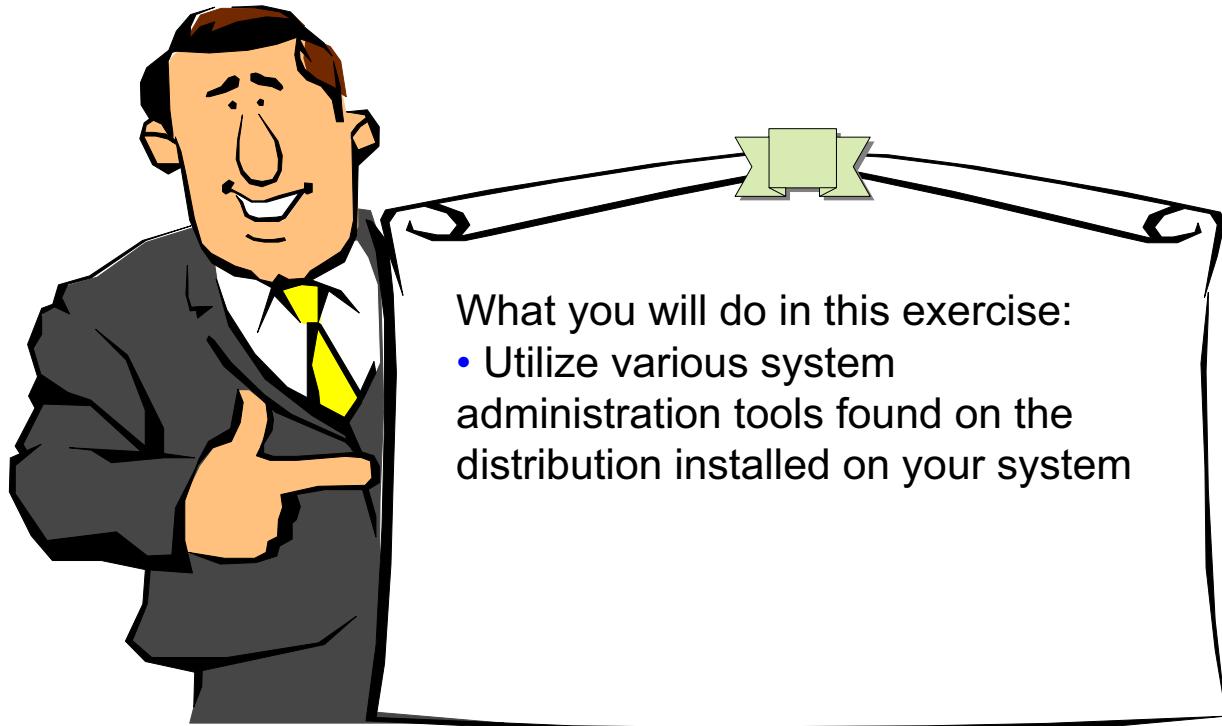
1. The RHEL [setup](#) tool provides a menu-based interface for various tools used during a text-based installation.  
The answer is [setup](#).
2. [True](#) or false: RHEL provides separate tools that start with system-config to administer the system with a GUI interface.  
The answer is [true](#).
3. SUSE provides a tool called [YaST](#) as a GUI interface/text menu tool to be used for various system administration tasks.  
The answer is [YaST](#).
4. What is the default port number to connect with the Webmin administration tool using a Web browser?  
The answer is [10000](#).

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —**

**Transition statement —**

## Exercise: System administration tools



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 5-10. Exercise: System administration tools

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Unit summary

Having completed this unit, you should be able to:

- Discuss the main characteristics of system administration tools
- List some distribution-specific administration tools
- List some general purpose administration tools

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 5-11. Unit summary

LX158.0

### Notes:

Having completed this unit, you should understand:

- System administration tools allow you to make system-wide configuration changes from a single tool
- System administration tools typically support multiple interfaces such as text, X, and Web
- Most Linux distributions have their own system administration tools for base configuration
- A general-purpose administration tool is Webmin

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

# Unit 6. Package management

## Estimated time

00:45

## What this unit is about

This unit teaches you how to use the most common packaging tool on a Linux system, the Red Hat/RPM Package Manager (RPM).

## What you should be able to do

After completing this unit, you should be able to:

- Describe the basic principles of RPM
- Use the **rpm** command or available graphical interface tool to:
  - Install software packages on the system
  - Remove software packages on the system
  - Update software packages on the system
  - Query software packages on the system
- Keep your system up to date

## How you will check your progress

Accountability:

- Checkpoint questions
- Machine exercises

## Unit objectives

---

After completing this unit, you should be able to:

- Describe the basic principles of RPM
- Use the rpm command or available graphical interface tool to:
  - Install software packages on the system
  - Remove software packages on the system
  - Update software packages on the system
  - Query software packages on the system
- Keep your system up to date

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 6-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

# Software management

- Historically difficult task when there are:
  - Numerous software vendors
  - Different types of archive format
  - Dependency issues
  - Numerous tools (or lack thereof)
- In the Linux community:
  - Simplified by the wide acceptance of the RPM Package Manager (RPM)



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-2. Software management

LX158.0

## Notes:

### Introduction

Software maintenance on a system has historically been a difficult task due to:

- Numerous software vendors
- Different types of archive format
- Dependency issues
- Numerous tools (or the lack thereof)

Over time software vendors (including those that provide operating systems) have used a number of open and proprietary archive distribution types (**cpio**, **tar**, **bff**, **cab**, and so forth) to distribute their products. The ability to check for software dependencies was either non-existent or limited in features. Finally, each archive would be installed using a variety of commands or techniques, and few, if any, were consistent with each other.

The task of software maintenance on Linux based systems has been simplified by the widespread use of the RPM Package Manager (RPM) system.

**Instructor notes:**

**Purpose** — Review with students that software maintenance on any system can be a difficult task, especially considering the incredible number of people and organizations that have developed software. The Linux community has simplified this by adopting the RPM Package Manager as the accepted method of choice.

**Details** — Prior to the release of RPM, system administrators would have to deal with a number of archive formats and maintenance procedures. It was seldom, if ever, possible through software to check to see if a third-party database product required a operating system component to be installed or at a certain level.

**Additional information** — bff is backup file format, and cab is used by Microsoft for its Windows operating system.

Debian and Debian-derived distributions, such as Ubuntu, use a different package management system. But a lot of the principles are still the same.

**Transition statement** — Okay, let's look at what the Linux community came up with.

# RPM Package Manager

- Command line software management system
- Allows for the following software operations:
  - Installation
  - Removal
  - Updating
  - Querying
  - Validation
  - Building
- Components:
  - Software archives
  - RPM-related commands
  - Database files: /var/lib/rpm



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-3. RPM Package Manager

LX158.0

## Notes:

### Introduction

The RPM Package Manager<sup>1</sup> or RPM, is a robust command line software management system that solves a lot of problems that a system administrator or distributor of software typically faces, such as:

- Management of source files
- Management of the build process
- A distribution method and format for binary files, including pre- and post-install scripts.

### RPM features

The RPM Package Manager provides the following features:

- Installation
- Removal

<sup>1</sup> This tool used to be called the Red Hat Package Manager, but Red Hat changed its name to emphasize that other distributions use it too. The new official name is RPM Package Manager, and yes, that is a self-referencing acronym (SRA), just like GNU.

- Updating
- Querying
- Validation

## RPM components

RPM is comprised of three separate components:

- Software archives or packages
- RPM-related commands
- Database files

**Instructor notes:**

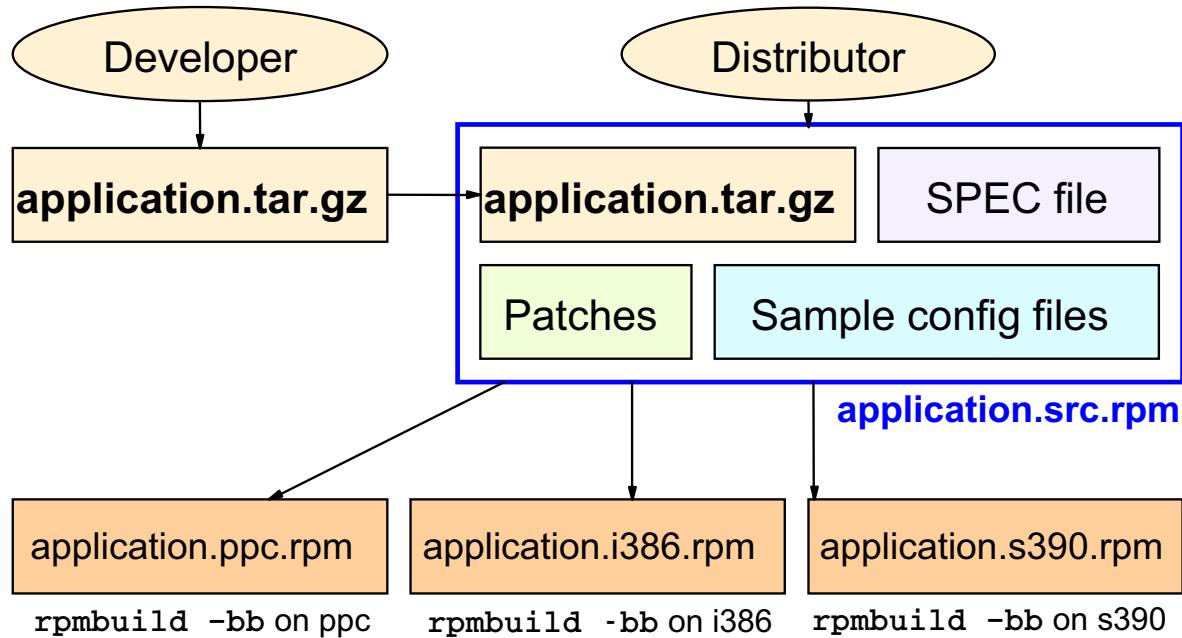
**Purpose** — Introduce students to the fact that RPM is a command line software management system.

**Details** — The RPM Package Manager is a command line software management system. It is full featured with the ability to install, remove, update, query, and validate software on a system. It is comprised of three parts: local database, command executables, and software archives or packages. We will look at each of these over the next series of visuals.

**Additional information** — Each Linux distribution has additional RPM tools. In the case of SUSE Linux, YaST has the ability to work with RPM archives.

**Transition statement** — Let's look at the philosophy behind RPM.

# RPM philosophy



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-4. RPM philosophy

LX158.0

## Notes:

### Introduction

The creators of RPM made an important observation: *In the Linux world, the person or organization writing the software would in most cases not be the person or organization that would distribute the software.* Because of this, RPM uses the philosophy of “pristine sources”. This means that the software that was developed is contained into a “Source RPM” file in a pristine state, exactly as it came from the developer. In this source RPM file (normally identified with the extension .src.rpm), you can also typically find patches and sample configuration files from the distributor, and, most importantly, a specification (SPEC) file.

The SPEC file contains all the information to unpack the pristine source, to patch it, and to compile it on any architecture. It also contains information on what files are included in a binary RPM.

With a correctly configured SPEC file, the only thing required to compile a package is the **rpmbuild -bb** (build binary) command on the target architecture. The binary RPM can then be distributed to all users of the distribution on that architecture.

When a developer develops a new version of its software, the only thing the distributor theoretically needs to do is to replace the pristine source with the new source package, rerun the **rpmbuild -bb** command, and a new version can be distributed.

In other words, thanks to this principle of "pristine sources", all the knowledge of the package builder is contained in the SPEC file. This allows others to take the source RPM and build a new, working binary RPM without having to talk to the original packager. In fact, you could take all the source RPMs that comprise a distribution and compile them yourself to get a new distribution. This can be handy if you want to port a distribution to a non-supported architecture, for instance.

***Instructor notes:***

**Purpose** — Introduce the RPM philosophy.

**Details** — Discuss the RPM philosophy.

**Additional information —**

**Transition statement** — Okay, let's look at some of the details. First, the file naming convention.

## RPM Files

---

- Software archives known as RPMs
- RPMs contain:
  - Program files
  - Configuration data
  - Dependency information
  - Data files
  - Documentation
- Naming convention
  - `name - version - release . architecture . rpm`
  - Example: `grub-0.95-3.5.i386.rpm`

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-5. RPM Files

LX158.0

### Notes:

#### Introduction

RPM Package Manager software archives or packages (also known as RPMs) are compressed archives that contain the following:

- Program files
- Configuration data
- Dependency information
- Data files
- Documentation

#### Naming convention

The naming convention for software packages is comprised of four fields:

- Software Name: The name of the software package.

This may be a single name, such as "xeyes", but complex products typically are packaged into multiple components. In that case, the different components are identified in the name as well: "glibc", "glibc-common", "glibc-devel", "glibc-headers" and so forth.

- Version: The original version number as set by the author of the package.
- Release: A secondary number that is used as the version number of the distributor. It typically identifies the various attempts of the distributor to create a working RPM file.

For simple packages, this release number is typically "-1", meaning that the first attempt of the distributor was the final version. But for complex products such as the kernel itself, there may be very complex numbering schemes.

- Architecture: The system architecture for which the package is intended/compiled. See below.

The filename for RPM packages end with the .rpm suffix.

### **Architecture types**

Architecture types are:

- ppc: PowerPC
- ppc64: PowerPC 64-bit
- s390: S390
- i386, i586, i686: x86 compatible (32-bit)
- x86\_64: x86 compatible (64-bit)
- alpha: Digital Alpha
- noarch: Non architectural dependent (e.g. shell scripts, text files and other generic file formats only.)
- src: Source package: The package that contains the original source files (typically .tar.gz), any patches that were applied, any files that were added by the distributor, and the SPEC file that governs the process of creating the binary RPMs.

### **Example**

For example, the file **grub-0.95-3.5.i386.rpm** breaks down to:

- Software Name: **grub**
- Version: **0.95**
- Release: **3.5**
- Architecture: **i386**

## **Instructor notes:**

**Purpose** — Introduce the students to the components of a RPM package and review the naming convention of RPM packages.

**Details** — An RPM package contains all the information needed to perform the software maintenance operations. The actual archive contains the parts shown on the visual. The white paper Maximum RPM

(<http://www.redhat.com/docs/books/max-rpm/max-rpm.pdf>) contains a good overview of RPM structure and command usage. Also the <http://www.rpm.org> site has good information as well.

### **Additional information —**

**Transition statement** — Next, let's look at the RPM-related commands found on LINUX distributions.

# RPM installing, freshening, and upgrading

- Installs, freshens, or upgrades an RPM
  - Freshen: Only install if an older RPM was installed
  - Upgrade: Always install, but uninstall older RPM first
- Basic syntax:
 

```
rpm -i package-filename.rpm (install)
rpm -U package-filename.rpm (upgrade)
rpm -F package-filename.rpm (freshen)
```
- Useful options:
  - v be verbose
  - h print 50 hash marks during installation

```
# rpm -ihv package-10.2-67.i386.rpm
package: #####
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-6. RPM installing, freshening, and upgrading

LX158.0

## Notes:

### Introduction

Installing an RPM can only be done if it was not already installed. If the RPM was already installed, you need to do an upgrade or a freshen. The difference between an upgrade and a freshen is that an upgrade always installs an RPM, even when a previous version was not installed. (It acts like a regular installation in that case.) A freshen only installs packages that actually have been installed previously. A freshen, therefore, is very handy to use if you downloaded a lot of patches from the Red Hat site, and you are not sure which patches you actually need. You can then just freshen all the packages, and only the things you need are actually installed.

### RPM syntax

The basic syntax for installing, freshening, and upgrading is respectively:

```
# rpm -i package-filename.rpm
# rpm -U package-filename.rpm
# rpm -F package-filename.rpm
```

Note that there is a difference between the package name and the package filename.

The RPM file which contains the package **foo** is generally called:

foo-version-release.architecture.rpm.

There are a number of options which make life a little easier on you:

- **-v** gives more information on what rpm is doing (verbose).
- **-h** prints 50 hash marks while installing so that you can track the progress. If you run rpm from a script, you can use these hash marks to make your own progress bar.
- **--nodeps** disables dependency checking.

Files in an RPM are marked as program, documentation or configuration files. When doing an upgrade or freshen, program and documentation files are automatically overwritten.

Configuration files are another matter altogether: Depending on the MD5 checksum of the original, actual, and new configuration file, the configuration file might be left in place, might be overwritten, might be saved with an extension **.rpmsave**, or might be saved with an extension **.rpmodif**. In fact, rpm can distinguish between six different cases. For more information, see the Maximum RPM book.

When installing, freshening, or upgrading packages, you can also specify the Web address of the package file instead of the package file itself. This allows you to do upgrades even on systems which are very tight on disk space but do have access to a network (for instance the Internet). Just ensure that the RPM files can be reached, either through FTP or HTTP, and you can do an upgrade. If you need to go through a proxy, there are options available to specify this proxy as well. Look at the **rpm** manual page for details.

**Instructor notes:**

**Purpose** — Describe installing, freshening and upgrading of an RPM.

**Details —**

**Additional information** — From the Maximum RPM book:

#### 4.1.1. Config file magic

While the **rpm -i** and **rpm -e** commands each do their part to keep config files straight, it is with **rpm -U** that the full power of RPM's config file handling shows through. There are no less than six different scenarios that RPM takes into account when handling config files.

In order to make the appropriate decisions, RPM needs information. The information used to decide how to handle config files is a set of three large numbers known as MD5 checksums. An MD5 checksum is produced when a file is used as the input to a complex series of mathematical operations. The resulting checksum has a unique property, in that any change to the file's contents will result in a change to the checksum of that file.

(Actually, there is a 1 in 2<sup>128</sup> chance a change will go undetected, but for all practical purposes, it is as close to perfect as we can get.) Therefore, MD5 checksums are a powerful tool for quickly determining whether two different files have the same contents or not.

In the previous paragraph, we stated that RPM uses three different MD5 checksums to determine what should be done with a config file. The three checksums are:

- The MD5 checksum of the file when it was originally installed. We will call this the original file.
- The MD5 checksum of the file as it exists at upgrade time. We will call this the current file.
- The MD5 checksum of the corresponding file in the new package. We will call this the new file.

Let's take a look at the various combinations of checksums, see what RPM will do because of them, and discuss why. In the following examples, we will use the letters X, Y, and Z in place of lengthy MD5 checksums.

##### 4.1.1.1. Original file = X, Current file = X, New file = X

In this case, the file originally installed was never modified. The file in the new version of the package is identical to the file on disk.

In this case, RPM installs the new file, overwriting the original. You might be wondering why go to the trouble of installing the new file if it is just the same as the existing one. The reason is that aspects of the file other than its name and contents might have changed. The file's ownership, for example, might be different in the new version.

##### 4.1.1.2. Original file = X, Current file = X, New file = Y

The original file has not been modified, but the file in the new package is different. Perhaps the difference represents a bug-fix or a new feature. It makes no difference to RPM.

In this case, RPM installs the new file, overwriting the original. This makes sense. If it Uemptly did not, RPM would never permit newer, modified versions of software to be installed! The original file is not saved since it had not been changed. A lack of changes here means that no site-specific modifications were made to the file.

#### 4.1.1.3. Original file = X, Current file = Y, New file = X

Here there is a file that was changed at some point. However, the new file is identical to the existing file prior to the local modifications.

In this case, RPM takes the viewpoint that since the original file and the new file are identical, the modifications made to the original version must still be valid for the new version. It leaves the existing, modified file in place.

#### 4.1.1.4. Original file = X, Current file = Y, New file = Y

At some point the original file was modified, and those modifications happen to make the file identical to the new file. Perhaps the modification was made to fix a security problem, and the new version of the file has the same fix applied to it.

In this case, RPM installs the new version, overwriting the modified original. The same philosophy used in the first scenario applies here: although the file has not changed, perhaps some other aspect of the file has, so the new version is installed.

#### 4.1.1.5. Original file = X, Current file = Y, New file = Z

Here the original file was modified at some point. The new file is different from both the original and the modified versions of the original file.

RPM is not able to analyze the contents of the files and determine what is going on. In this instance, it takes the best possible approach. The new file is known to work properly with the rest of the software in the new package - at least the people building the new package should have insured that it does. The modified original file is an unknown: it might work with the new package, or it might not. So RPM installs the new file.

But the existing file was definitely modified. Someone made an effort to change the file, for some reason. Perhaps the information contained in the file is still of use. Therefore, RPM saves the modified file, naming it <file>.rpmsave, and prints a warning so the user knows what happened:

```
warning: /etc/skel/.bashrc saved as /etc/skel/.bashrc.rpmsave
```

#### 4.1.1.6. Original file = none, Current file = ??, New file = ??

While RPM does not use checksums in this particular case, we will describe it in those terms, for the sake of consistency. In this instance, RPM had not installed the file originally, so there is no original checksum.

Because the file had not originally been installed as part of a package, there is no way for RPM to determine if the file currently in place had been modified. Therefore, the

---

checksums for the current file and the new file are irrelevant; they cannot be used to clear up the mystery.

When this happens, RPM renames the file to <file>.rpmod, prints a warning, and installs the new file. This way, any modifications contained in the original file are saved. The system administrator can review the differences between the original and the newly installed files and determine what action should be taken.

As you can see, in the majority of cases, RPM will automatically take the proper course of action when performing an upgrade. It is only when config files have been modified and are to be overwritten that RPM leaves any post-upgrade work for the system administrator.

Even in those cases, many times the modified files are not worth saving and can be deleted.

**Transition statement** — Let's look at uninstalling RPMs.

## RPM uninstalling

- For uninstalling an RPM, use the **-e** option

```
# rpm -e kdelibs3  
error: removing these packages would break  
dependencies:  
    kdelibs3 >= 3.1 is needed by kdebase3-3.1.1-63  
    libDCOP.so.4 is needed by kdelibs3-cups-3.1.1-13  
    ...
```

- Options:  
**--nodeps** (ignore any dependency breaks)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-7. RPM uninstalling

LX158.0

### Notes:

#### Introduction

Uninstalling (removal of) a software package from the system can be accomplished by using the **-e** option of the **rpm** command and the package name (not the package filename). Before removing the package, the **rpm** command checks the RPM database files for packages that might be dependent on the package to be removed. If a dependency is found, the **rpm** command will error out with a message similar to the example shown in the visual.

#### Ignore dependencies

The **--nodeps** option of the **rpm** command will cause the removal of a software package with no dependency checking.

***Instructor notes:***

**Purpose** — Describe RPM uninstalling of a software package.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Let's look at querying.

# RPM querying

- Queries the contents of an installed RPM

- Basic syntax:

- `rpm -q package-name`

- Options:

– -a	Query all installed packages
– -f <file>	Query package which owns file
– -p <package-file>	Query package-file
– -i	Display package information
– -l	Display package files
– -s	Display state of all files
– -d	Display documentation files
– -c	Display configuration files

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-8. RPM querying

LX158.0

## Notes:

### Introduction

RPM querying is the process of retrieving information about installed packages. The basic syntax is `rpm -q package-name`, but that only displays the package name. It is the options that make querying interesting:

**-a:** Queries all packages which are installed on the system.

**-f <file>:** Queries which package contains <file>.

**-p <package-file>:** Queries the (not yet installed) <package-file>.

**-i:** Displays all package information: name, version, release, install date, group, size, summary, description, build information and so forth.

**-l:** Lists all files in the package.

**-s:** Displays the state of each file in the package.

The state is either normal, not installed, or replaced.

- d: Displays all files that are listed as documentation.
- c: Displays all files that are listed as configuration files.

### Query examples

With these options, you can do a number of great things. Below are some examples:

- Do you want to know which package the dig program is in? Try **rpm -qf `which dig`** or **rpm -qif `which dig`**
- Need to know what documentation is available for a specific command, and **man -k commandname** does not work? Try **rpm -qdf `which nslookup`**
- Need a lot of data to test a network connection? Try **rpm -qila**
- Need to know which not yet installed RPM package file contains the program "pico"? Sorry, you are out of luck here. RPM only queries one RPM package at a time, so you need to do something like this:

```
for package in *.rpm
do
    rpm -q -l -p $package | grep -q pico
    if [ $? = 0 ]
    then
        echo $package
    fi
done 2>/dev/null
```

**Instructor notes:**

**Purpose** — Introduce the query mode.

**Details** — Discuss the student notes.

**Additional information** — The **which** command, which is mentioned in the student notes, is something the students might not have seen before. What it basically does is show the location of a certain executable. It might be worth introducing it here as a sidestep.

**Transition statement** — Let's look at the last mode, verification.

## RPM verification

- Verifies the actual files with the original RPM

– Size	S
– MD5 checksum	5
– Permissions, type	M
– Owner	U
– Group	G
– Modification time	T
– Symbolic link	L
– Device	D

```
# rpm -V kdelibs3
.M..... /opt/kde3/kpac_dhcp_helper
.....T /opt/kde3/share/mimelnk/application/x-applix.desktop
```

A dot (.) means test passed.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-9. RPM verification

LX158.0

### Notes:

#### Introduction

The verify option (-V) verifies all files that are supposed to be present in the RPM against the files that are available on disk. This is a very easy way to check for any unauthorized configuration changes.

The following checks are performed on each file in an RPM:

#### Check - Description

**S** - File size. This checks whether the size of the file has changed.

**5** - MD5 checksum. This is a very hard-to-fool checksum which checks whether the contents of a file have changed.

**M** - Mode. Are permissions, switch user ID (SUID) and switch group ID (SGID) bits, and the file type still the same?

**U** - User. Is the owner of the file still the same?

**G** - Group. Is the group of the file still the same?

**T** - File modification time. This checks whether the file modification timestamp (mtime) has changed.

**L** - Symbolic link. This verifies whether a certain symlink has changed.

**D** - Device. This verifies whether the major and minor numbers of a device are still intact.

If a file checks out okay, there can be no output. If there is a discrepancy, however, the name of the involved file can be listed, prepended by the discrepancy information. The output line then looks like this:

```
# rpm -V sendmail  
SM5....T c /etc/sendmail.cf
```

This means that a discrepancy was found in the file /etc/sendmail.cf. This is to be expected, since this file is a configuration file (hence the “c” in the line. The discrepancy information in this case is SM5....T, in which each letter denotes a certain discrepancy from the list above. In this case, the following discrepancies were found: size, mode, MD5 checksum, modification time.

Note that this cannot be used in place of more advance Intrusion Detection Systems such as tripwire: the /var/lib/rpm database is not encrypted or secured in another way, and any hacker worth his salt might not only change a file on disk but can also modify the corresponding entry in /var/lib/rpm.

**Instructor notes:**

**Purpose** — To introduce students to how to validate what is installed on the system against the RPM database files.

**Details** — Discuss student notes.

**Additional information** — In the Red Hat Installation Guide, which comes with Red Hat, there is an excellent unit which has many fancy uses of the **rpm** command. Point this out to the students.

Additional information on RPM can be found at <http://www.rpm.org>.

**Transition statement** — The last thing that might be handy is that RPMs can be signed by the distributor.

# RPM signatures

- RPMs can be signed by the distributor
- To verify signature:
  - Obtain public key of distributor
    - CD-ROM
    - Internet
  - Add public key to keyring using **rpm --import**
  - Verify package with **rpm --checksig**
- List all imported signatures with **rpm -qa gpg-pubkey\***

```
# rpm --import /mnt/cdrom/RPM-GPG-KEY
```

```
# rpm --checksig passwd-0.64.1-1.i386.rpm
passwd-0.64.1-1.i386 md5 gpg OK
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-10. RPM signatures

LX158.0

## Notes:

### Introduction

The RPM Package format also features the ability to include a digital signature of a package, and most distribution builders actually make use of this feature as an effective measure against trojan horses introduced in an RPM after release by the distribution builder.

Verifying this signature is a two-step process. The first step is to obtain the public key of the distribution builder. This key is stored in a text file which can usually be found on the original CD-ROMs or on the distribution Web site. This public key needs to be added to your “keyring,” your database of public and secret keys in your home directory. This is done with the following command:

```
# rpm --import /mnt/cdrom/RPM-GPG-KEY
```

**Note:** Some distributions (for instance, SUSE), perform this step automatically while installing.

The second step is to verify each individual package. This is done with the command:

---

```
# rpm --checksig packagename
```

If the output is gpg OK, then you can be sure that it was indeed the distribution builder that built this individual package and that no one has tampered with it since.

***Instructor notes:***

**Purpose** — Introduce students to the use of RPM signatures.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Okay, that finishes our discussion on single RPM files. However, RPM files don't live on their own. A typical distribution might have 2000 RPM files, and some of these may be dependent on others.

## RPM dependencies

- Most RPMs require other RPMs in order to work properly.
- These libraries and other systems that an RPM depends on are called *dependencies*.
- These can greatly complicate your life if you are installing new software.
- **rpm -q --provides** shows what capabilities a package provides.
- **rpm -q --requires** shows what capabilities a package requires.
- Solutions for automated dependency solving are different across distributions.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-11. RPM dependencies

LX158.0

### Notes:

Most of the RPMs that are contained in a typical distribution are somehow dependent on the presence of other RPMs. For instance, most RPMs require the "glibc" library to be available to handle standard I/O tasks. RPMs containing perl scripts will require the perl interpreter to be present and so forth.

All these dependencies can greatly complicate your life if you try to install an RPM that has dependencies that are not satisfied. There are query commands that show you what capabilities a package requires, and what capabilities a package provides. But to query over 2000 RPMs to see what RPM provides your required capability quickly becomes tedious, especially considering that there may be tens or hundreds of requirements, requirements can sometimes be fulfilled by multiple RPMs, and when you have finally found the proper RPM, you may find that it requires a large number of capabilities as well. It is not uncommon to want to install a single, small RPM, but having to pull in dozens of RPMs, several 100s of MBs in size, to satisfy all dependencies.<sup>1</sup>

<sup>1</sup> A typical example is if you have the GNOME desktop environment installed, but you want to run a game or something that is written for the KDE desktop environment. This then requires the installation of the majority of KDE libraries.

The only proper solution for this is to create a distribution-wide database that lists all capabilities and requirements of all packages of that distribution. How and where this database is implemented is dependent on the distribution, and typically integrated with the tools that allow you keep your distribution up to date.

***Instructor notes:***

**Purpose —** Discuss the "dependency hell".

**Details —**

**Additional information —**

**Transition statement —** Okay, let's see what the distributions have come up with for us.

# Solutions for RPM dependencies/updates

- Red Hat:
  - Yum: Command line tool to search repositories for dependent packages and brings in all needed components
  - PackageKit: Graphical interface for Yum
  - Red Hat Network: Subscription based service for central package management
- SuSE
  - YaST: GUI tool to search repositories for dependent packages and bring in all needed components
  - Zypper: Command line tool to search repositories for dependent packages and bring in all needed components

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-12. Solutions for RPM dependencies/updates

LX158.0

## Notes:

Each distribution solves the dependency problem differently.

**Red Hat** uses the **yum** tool, in combination with online or local **yum repositories**.

A **yum repository** is a collection of RPM files which have been indexed with the **createrepo** command, so that an index file **repomd.xml** is created, plus a series of files that describe the package set. This repository can then be used locally (if it's mounted in the local file system) or via a network protocol such as HTTP or FTP.

You can use several repositories simultaneously.

**yum** accesses these various repositories, retrieves (and caches) the index files, and then uses these repositories to download and install/upgrade the required software. Implicit in this process is the resolution of dependencies.

**PackageKit** is essentially a graphical frontend for **yum**.

The **Red Hat Network** (RHN) is a website that allows you to do package management across a large number of systems. RHN requires a (paid) subscription and is included in your Red Hat support contract.

---

**SUSE** uses the **YaST Software** module for package management. This allows you to manage repositories, install/update/deinstall software, and update your SUSE system online.

SUSE also offers the **zypper** tool, which allows you to do repository and package management from the command line.

***Instructor notes:***

**Purpose** — Give an overview of tools

**Details** —

**Additional information** —

**Transition statement** — Let's look at yum first.

## Yum

- Use **yum** to keep up-to-date or install additional software from yum repositories
- **yum** performs dependency checking automatically
- Syntax:
  - `yum install package1 [package2] ...`
  - `yum update [package1] [package2] ...`
  - `yum check-update`
  - `yum remove package1 [package2]`
  - `yum list '*package*'`
  - `yum info package`

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-13. Yum

LX158.0

### Notes:

**yum** is the Red Hat command line tool for package management. The first time it is started (and any time the list of repositories has changed), it will download and cache the index files from the repositories that are defined in /etc/yum.repos.d.

You can then use **yum** to list the packages in the repositories, retrieve information about packages and install/update/remove them.

Dependencies are taken into account by default.

***Instructor notes:***

**Purpose** — Introduce yum

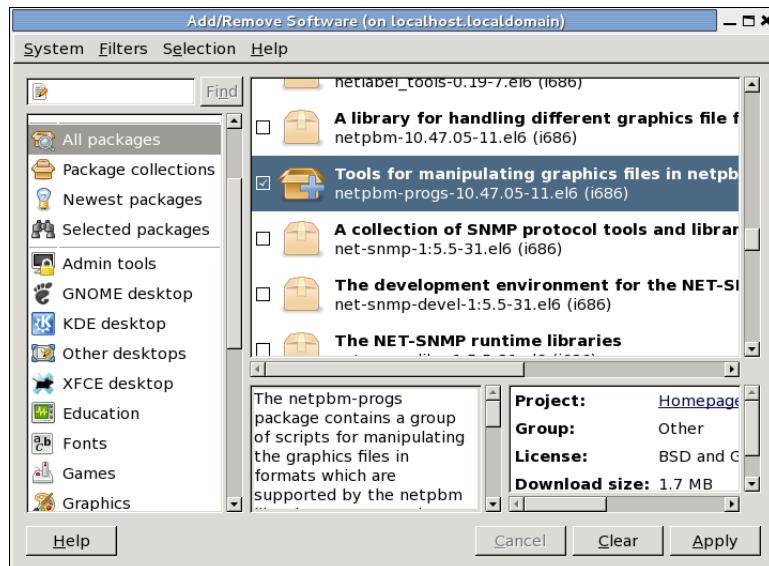
**Details** —

**Additional information** —

**Transition statement** — Let's look at an alternative.

## PackageKit

- GUI interface to yum
- Installing/deinstalling of packages only, no repository management



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-14. PackageKit

LX158.0

### Notes:

**PackageKit** is the graphical interface to **yum**. Its functionality however is limited to package management; it cannot be used to manage repositories.

The advantage of PackageKit over yum is that you can run PackageKit as a regular user. Via the D-BUS interface it communicates with a PackageKit root process, provided that you provided the root password during startup.

**PackageKit** is started from the GUI via System > Administration > Add/Remove Software.

**Instructor notes:**

**Purpose** — Discuss PackageKit

**Details** —

**Additional information** —

**Transition statement** — Let's look at yum repositories.

## Yum repositories

- Yum repositories are described in /etc/yum.repos.d
- Can be enabled/disabled by administrator
- Yum will search each enabled repository by default
  - Use **yum repolist** for a list of all available repositories
  - Use **yum --enablerepo** and **--disablerepo** to enable/disable repositories during yum operations
- Can add repositories yourself
  - Local repositories
  - Public repositories with software not included in RHEL

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-15. Yum repositories

LX158.0

### **Notes:**

As said earlier, yum can use multiple repositories simultaneously. Each repository is described with a single text file in /etc/yum.repos.d. Repositories can be enabled/disabled by the administrator, or on the fly with the **--enablerepo** and **--disablerepo** options to yum.

If you want to, you can add repositories to the list. These can be local repositories (for instance a copy of the RHEL DVD contents) or public repositories with software that is not included in RHEL.

To add a local DVD as a repository, first copy the DVD contents to disk. Then re-create the index of files.

```
# mount /dev/scd0 /media/dvd
# mkdir /export/rhel61
# cp -a /media/dvd/* /export/rhel61
# cd /export/rhel61
# createrepo -g <comps.xml file> .
```

Now create the /etc/yum.repos.d/local.repo file. The contents of this file should be as follows:

```
[local]
name=Local RHEL 6.1 Repository
baseurl=file:///export/rhel61
enabled=1
gpgcheck=0
```

**yum** will now by default include this repository when searching for software.

In addition to local repositories and the "official" Red Hat repositories on the Internet, there are additional repositories available on the Internet as well. These repositories typically contain software that is not included by Red Hat due to, for instance, license or patent issues.

An example of such a public repository is Repoforge ([www.repoforge.org](http://www.repoforge.org)). To start using it, first download the appropriate repoforge RPM and install it with **rpm -i**. This RPM will then add the Repoforge repository descriptions to /etc/yum.repos.d. The next invocation of **yum** will then download the Repoforge index, after which you can use this repository as well.

**Instructor notes:**

**Purpose** — Discuss yum repositories

**Details** —

**Additional information** —

**Transition statement** — Finally, let's look at Red Hats repository

## Red Hat Network

- Subscription based service
- Create and manage account and systems on <http://rhn.redhat.com>
- Register individual systems with **rhn\_register**, **subscription-manager** or **subscription-manager-gui**
- Use **yum** to keep systems current, or use Web interface at <http://rhn.redhat.com>  
(requires **rhnsd** daemon running on system)



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-16. Red Hat Network

LX158.0

### Notes:

The **Red Hat Network** is a subscription-based service that is normally included in your Red Hat support contract.

To use RHN you first have to register each individual system with RHN. This can be done using command-line tools such as **rhn\_register** or **subscription-manager**, or with the GUI tool **subscription-manager-gui**. The registration process is also started when you install Red Hat Enterprise Linux (but can be skipped if necessary).

Once your system is properly registered and you have the proper entitlements, it allows you to use the RHN-based yum repositories. This means that you can now use yum to install/update software.

But RHN is more. It keeps track of all the software that is installed on your systems, and will present this via an easy to use website. It will flag systems that are out of date, and allows you to schedule bulk updates for multiple systems at once.

---

Each system that is registered with RHN should run the **rhnscd** (Red Hat Network Service Daemon). This daemon regularly checks in with RHN and if any outstanding actions are found, will execute these actions.

**Instructor notes:**

**Purpose** — Discuss RHN

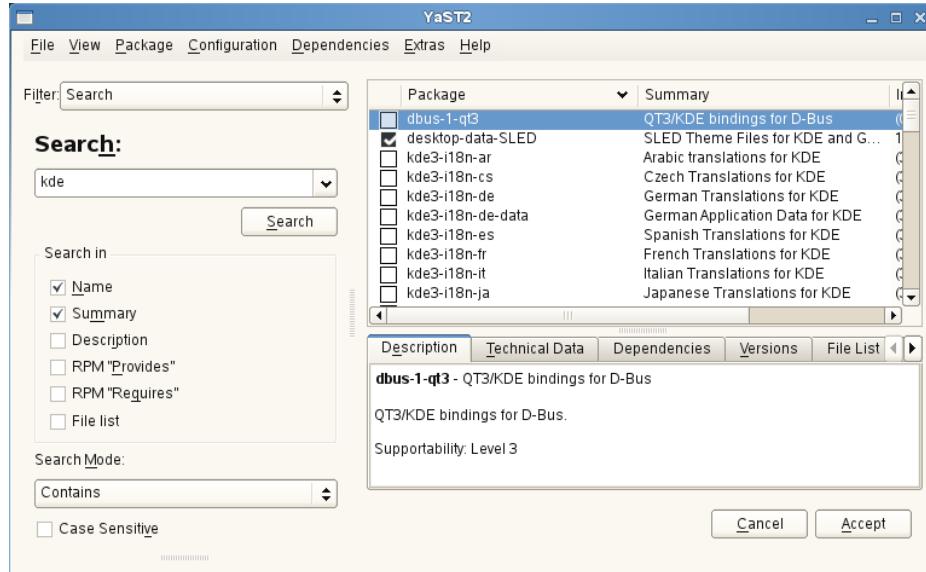
**Details** —

**Additional information** —

**Transition statement** — Okay, that's it for Red Hat. Let's look at SUSE.

## YaST software module

- Manage repositories including those requiring subscription
- Install additional software
- Update software (shortcut: **you**)



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-17. YaST software module

LX158.0

### Notes:

On a SUSE system, the main tool for managing software repositories, and to perform package management, is the YaST Software module.

Defining repositories is easy: You only need to supply the URL of the repository (either local or via some network protocol), and make sure the contents of the relevant DVDs are copied to that location.

Once repositories have been set up, you can use this tool to install/update software. To do online updates, YaST will actually use the **YOU** (YaST Online Update) module. This module can be started directly from the command line with the **you** command.

***Instructor notes:***

**Purpose** — Introduce the YaST Software module.

**Details** —

**Additional information** —

**Transition statement** — Let's also look at a command-line tool.

## Zypper

- Command-line interface to manage SUSE repositories, install/update software and so forth
- **zypper** commands:
  - repos, addrepo, removerepo
  - install, remove, verify
  - list-updates, update, list-patches, patch, dist-upgrade
  - search, info, what-provides

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-18. Zypper

LX158.0

### **Notes:**

Zypper is the SUSE command-line tool to manage SUSE repositories, install/update software and so forth. Its usage is relatively straightforward.

***Instructor notes:***

**Purpose —** Introduce Zypper

**Details —**

**Additional information —**

**Transition statement —** That's it about RPMs. Time for the checkpoint questions.

## Checkpoint

1. Which basic modes of operation does rpm have?
2. Which command can I use to verify that the permissions of /etc/sendmail.cf are still correct?
3. Which software maintenance operations does the rpm command provide?
  - a. Installation of a RPM package
  - b. Installation of a tar ball archive
  - c. Removal of seldom used packages
  - d. Updating a package
  - e. Verification of package installation

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-19. Checkpoint

LX158.0

### Notes:

**Instructor notes:**

**Purpose —**

**Details —**

## Checkpoint solutions

---

1. Which basic modes of operation does rpm have?

The answers are [install, freshen and upgrade, uninstall, query, and verify.](#)

2. Which command can I use to verify that the permissions of /etc/sendmail.cf are still correct?

The answer is [rpm -V -f /etc/sendmail.cf.](#)

3. Which software maintenance operations does the rpm command provide?

- a. [Installation of a RPM package](#)
- b. Installation of a tar ball archive
- c. Removal of seldom used packages
- d. [Updating a package](#)
- e. [Verification of package installation](#)

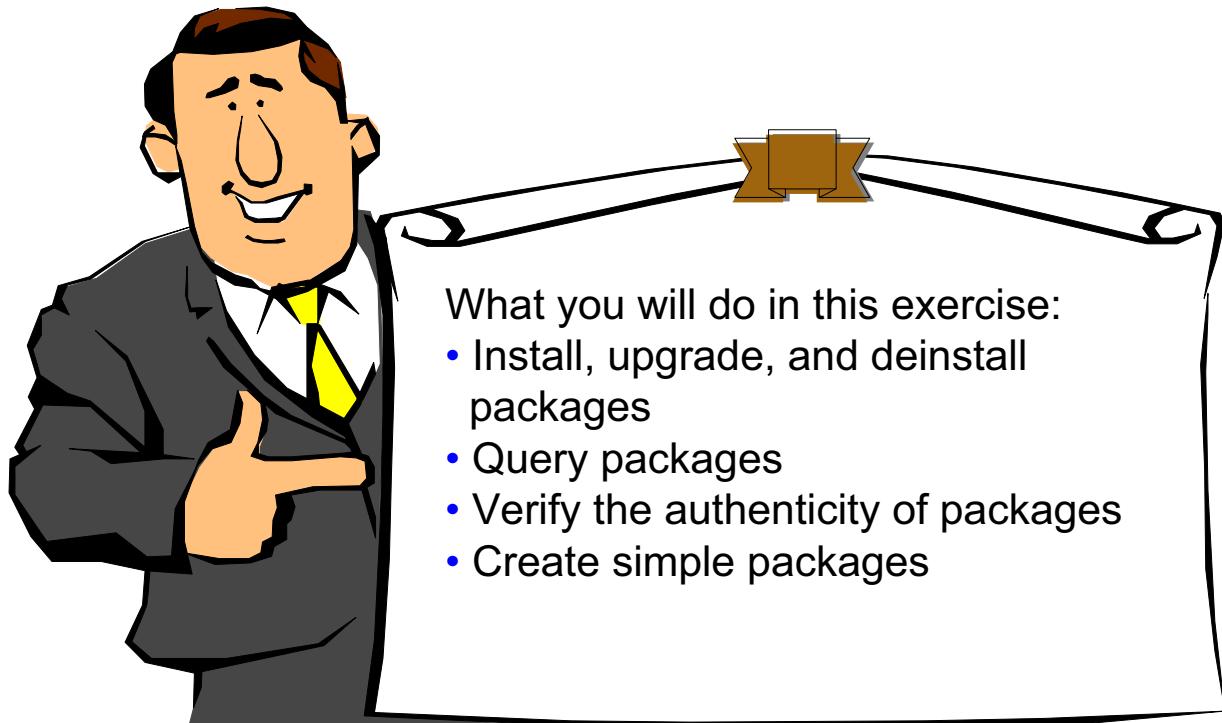
The answers are [installation of a RPM package, updating a package, and verification of package installation.](#)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —**

**Transition statement —** Let's do some exercises.

## Exercise: Package management



- What you will do in this exercise:
- Install, upgrade, and deinstall packages
  - Query packages
  - Verify the authenticity of packages
  - Create simple packages

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-20. Exercise: Package management

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Unit summary

Having completed this unit, you should be able to:

- Describe the basic principles of RPM
- Use the **rpm** command or available graphical interface tool to:
  - Install software packages on the system
  - Remove software packages on the system
  - Update software packages on the system
  - Query software packages on the system
- Keep your system up to date

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 6-21. Unit summary

LX158.0

### Notes:

Having completed this unit, you should understand:

- RPM is a versatile tool for package management
- An RPM file can be a source RPM or binary RPM
- A source RPM contains the pristine package source, patches, sample configuration files, and a SPEC file
- The SPEC file contains details about the build process
- A binary RPM contains the compiled code and is specific for an architecture
- Several integrated package management tools exist
- Each distribution has its own solution for keeping up-to-date with patches

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

# Unit 7. X Window System and VNC

## Estimated time

00:45

## What this unit is about

The unit teaches you how to use and configure the X Window System and VNC.

## What you should be able to do

After completing this unit, you should be able to:

- Describe the basic architecture of the X Window System
- Configure X.org
- Start and stop X
- Describe the function of the window manager
- Use X over a network
- Describe the VNC protocol
- Activate VNC from your Desktop Environment
- Configure and start the VNC server

## How you will check your progress

Accountability:

- Checkpoint questions
- Machine exercises

## Unit objectives

---

After completing this unit, you should be able to:

- Describe the basic architecture of the X Window System
- Configure X.org
- Start and stop X
- Describe the function of the window manager
- Use X over a network
- Describe the VNC protocol
- Activate VNC from your Desktop Environment
- Configure and start the VNC Server

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose** — Define unit objectives.

**Details** —

**Additional information** —

**Transition statement** — What is the X Window system?



## 7.1. X Window System

### Instructor topic introduction

**What students will do —**

**How students will do it —**

**What students will learn —**

**How this will help students on their job —**

## X Window System

---

- Graphical user interface of UNIX
- Initially developed at MIT
- Currently licensed by the X Consortium, Inc.
- In Linux implemented as a separate program that runs in user space
- Uses client/server architecture

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-2. X Window System

LX158.0

### Notes:

#### Introduction

The X Window System, X for short, is the GUI of Linux. It is implemented as a separate program that runs in user space, and it uses a client/server architecture.

The X Window System, more commonly referred to as simply X (but never as Windows), is the set of device drivers and libraries that puts a graphical interface on most UNIX/UNIX-like systems. It was developed during the 1980s primarily for high-end, research-oriented hardware running in networked environments - but times have changed.

X Window system servers run on computers with bitmap displays. The server distributes user input to and accepts output requests from various client programs through a variety of different interprocess communication channels. Although the most common case is for the client programs to be running on the same machine as the server, clients can be run transparently from other machines (including machines with different architectures and operating systems) as well.

---

What X.org does is provide a client/server interface between the display hardware (those physical things like the mouse, keyboard, and video displays) and the desktop environment (this is typically called a window manager as it deals with how X is displayed that is, the overall appearance).

All of this, makes X.org platform-independent, network-transparent, and extensible. In short, X.org is an open source X11-based desktop infrastructure.

**Instructor notes:**

**Purpose** — Introduce the X Window System.

**Details —**

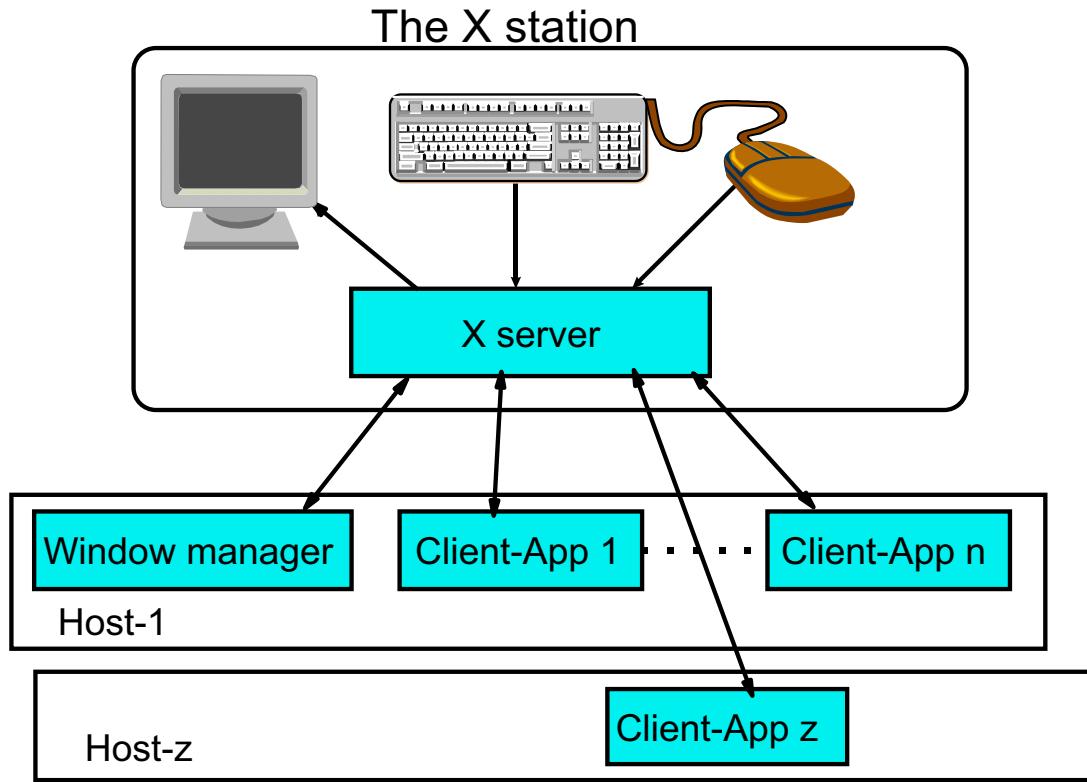
**Additional information** — IN THE BEGINNING... In its original incarnation, UNIX, child of the console generation, lacked anything remotely resembling a graphical user interface. When personal computers arrived on the scene, they too followed the text-oriented approach with products like the Apple II. In the 1980s, the introduction of the Apple Macintosh made everyone aware of the need for graphical interfaces on desktop computers. Around the same time, Microsoft began marketing its GUI-based OS, Windows. Both Microsoft Windows and the Macintosh failed to separate the duties of the OS and the windowing environment - the two were molded together.

In 1984, not long after the introduction of the Macintosh, the X Window System was born, and UNIX got its GUI. X took a fundamentally different approach to GUI design and implementation. From the beginning, X was designed to be used in a networked environment, and as such, was designed with a client/server model in mind. As a result, an X server makes no assumptions about its client's rendering hardware. This created obvious advantages (making remote computing feasible, for instance), some difficulties (putting security issues on the front burner), and some not-so-obvious drawbacks that would become more important as hardware capable of rendering 3D graphics became widespread. And of course, the networked computers that X was originally designed to run on in 1984 were high-end (and extremely expensive) scientific workstations - definitely not the kind of machines the average user was likely to have lying around the house.

Sometime in 1989 or 1990, a German student named Thomas Roell began porting the source code for the X server provided in the X Version 11 Release 4 (X11R4) distribution to work with a graphics card he had installed in a 33 MHz Intel 386-based PC (with no floating point unit, mind you - this is old and slow hardware). He eventually released his X server, which he called X386.1.1. It caught the eye of some X developers at MIT, the X Consortium, and the Dell UNIX team in Austin, Texas. Dell brought Roell over to work on drivers for some graphics cards on a multiprocessor system that was to run a licensed version of UNIX System V Release 4 (SVR4) for Intel systems. While he was at Dell, Roell worked with Stephen Gildea of the X Consortium and Mark Snitily of Snitily Graphics Consulting Service (SGCS). Together, they worked to take Roell's next X server and make it the reference implementation for PC-based X Window systems. When X11R5 was released on August 29, 1991, Roell and the X Consortium gave PC-based UNIX its first official X implementation. And just in time too, as Linux had been born a few weeks earlier.

**Transition statement** — Let's look at history of getting access to a UNIX system to see how and why the X Window system was developed.

# X client/server architecture



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-3. X client/server architecture

LX158.0

## Notes:

### Introduction

The X Window system uses a client/server architecture, which makes it very flexible. The central piece of software is the X server, which runs on the X station. This server traps all keyboard and mouse events and sends them to the appropriate application. If an application wants to put something on the screen, it sends that data to the server, which then performs the necessary hardware calls to the graphical adapter.

Any application can connect to the X server, but there should always be one special application active: the window manager. This window manager basically puts a border around each application window and allows you, for instance, to drag windows around the screen. There are numerous window managers available, each with their own style.

Other applications also connect to the X server and have their data displayed through it. Common examples are:

- **xterm**, which emulates a terminal screen, allowing you to enter Linux commands

- **xeyes**, which displays a pair of eyes on your screen, looking at the mouse pointer
  - **xbanner**, which displays a background image
  - **xcalc**, a mathematical calculator
  - **xedit**, a GUI-based editor
- and many, many more.

The connection between the X server and the X clients (including the Window manager) is a TCP/IP connection. It is therefore possible to run the X client on another system.

***Instructor notes:***

**Purpose** — Introduce the client/server architecture of X.

**Details** —

**Additional information** —

**Transition statement** — Let's talk about the X station a little more.

## Examples of X stations

- Hardware X stations
  - X server program stored in ROM chip
- UNIX/Linux
  - X server implemented as a separate program that uses the entire graphical screen to display X clients
  - UNIX/Linux can run the X clients and X server program on the same system (stand-alone solution)
- MS Windows
  - X server implemented as a separate program that uses the Windows GUI to display X clients
    - For example, Hummingbird eXceed, and others
- Xnest
  - X server implemented as an X client

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-4. Examples of X stations

LX158.0

### Notes:

### Introduction

There are several X stations possible:

- Real X stations are hardware devices which consist of a monitor, a keyboard, a mouse, and a ROM chip containing the X server program. These devices cannot do any local processing and thus need to be connected to a network at all times.
- UNIX/Linux stations with a graphical display can run an X server as a separate program. In most cases, the X server will grab the entire graphical screen.
- On most UNIX/Linux systems, the X clients and X server run on the same system, communicating with each other through the TCP/IP loopback interface or through a UNIX socket<sup>1</sup>. This makes it possible to use X as a stand-alone solution.

<sup>1</sup> A special file (type s) in a UNIX/Linux file system which makes TCP/IP-like communications between two processes possible. Because these sockets are limited to the local file system, they are generally more secure than TCP/IP connections. Furthermore, their overhead is slightly less, thus increasing performance.

- Several X servers exist that run under MS-Windows: Hummingbird eXceed, WRQ Reflection X and many others. These programs typically open an MS-Windows window and run the X server inside it.
- **Xnest** is an X client that implements an X server. In other words: it is an X server in a window. This is useful for testing.

***Instructor notes:***

**Purpose** — Discuss X stations.

**Details** —

**Additional information** — A nearly-free X server for Windows that I use personally is Xmanager from <http://www.netsarang.com>.

**Transition statement** — Now, let's look at how Linux implements the X server.

# X servers in Linux

- Open source
  - X.org (<http://www.x.org>)
  - Included in virtually all Linux distributions
- Commercial
  - Xi graphics: <http://www.xig.com>
- VNC Server
  - Does not manage physical hardware, but allows VNC clients to connect via the network

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-5. X servers in Linux

LX158.0

## Notes:

### Introduction

The X server that is most often used with Linux is X.org, an open source server which is, just like Linux, developed as a joint effort of various programmers on the Internet. Their Web page is <http://www.x.org>.

Commercial X servers are also available for Linux. One example is Accelerated-X from Xi Graphics Inc. The advantage of these X servers is that they might have better support for certain specialty graphics adapters.

In addition to this, the VNC server is also an X server. The difference between it and, for instance, X.org, is that the VNC server doesn't manage any physical hardware. Instead, it allows VNC clients to connect to it via a network connection so that it can be accessed remotely.

## **Instructor notes:**

**Purpose** — Introduce various X servers.

**Details** — XFree86 was the default X server for many years. Version 3 used multiple binary files to support many adapters. Version 4 used one binary which was able to detect the adapter this is being used.

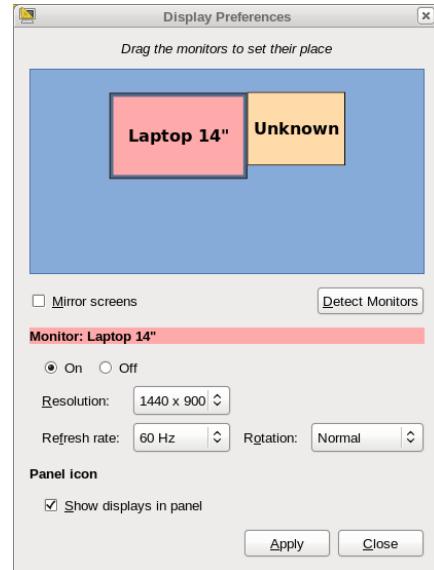
XFree86 4.3 was used to create Xorg 6.7. Xorg is now the main default X server in most major versions of Linux.

### **Additional information —**

**Transition statement** — The default X server on Linux is Xorg. Let's look at configuring Xorg.

## X.org configuration

- Traditionally done as root, stored in /etc/X11/xorg.conf
- Recent versions of Xorg & kernel allow for dynamic (re)configuration, even as regular user
  - gnome-display-properties



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-6. X.org configuration

LX158.0

### Notes:

#### Introduction

On every system which runs the X.org X server, a configuration file has to be created. This file contains the hardware characteristics of the system running the server: graphical adapter type, monitor type, mouse type, and keyboard type and language.

#### Configuration

In the early days you had to configure X.org yourself. This was a very tricky process which could, if done improperly, actually burn out your monitor. Later on, tools were created to configure X.org for you.

Since a few years X.org has become much more integrated with the Linux kernel and the result of that is that configuration of X.org is almost completely trivial. In fact, in most distributions you can reconfigure X.org on the fly through an icon or menu item in your

Desktop Environment. This then starts **gnome-display-properties** which allows you to set resolutions, color depth, refresh rates and so forth for all attached monitors.

If you have an exotic hardware configuration you still might need to edit /etc/X11/xorg.conf by hand, but the far majority of hardware allows you to run X.org without an /etc/X11/xorg.conf file.

***Instructor notes:***

**Purpose** — Discuss Xorg configuration.

**Details** —

**Additional information** —

**Transition statement** — Once our X configuration file is configured, you can start X.

# Starting X

- To start X.org and your favorite window manager, use **startx**
  - Starts X.org on a free virtual display (usually number 1 or 7)
  - Starts your favorite window manager
  - To start a second X session, use `startx -- :1`

```
$ startx  
...
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-7. Starting X

LX158.0

## Notes:

### Introduction

X itself is started with the **x** command. This starts an X server on the first free virtual terminal. Depending on the configuration, this will normally be VT 1 or VT 7.<sup>1</sup>

However, with only an X server running, you will not get anywhere: you will just get an empty, gray or black screen with an X-shaped mouse pointer. This is useful for debugging your X configuration file, but in order to do anything useful, you need to start a window manager too.

With the **startx** command, this is exactly what is accomplished. First, Xorg is started, and a few seconds later, your favorite window manager is started.

What your favorite window manager is, is determined by reading the configuration files in your home directory. Each distribution has a different setup for determining the favorite window manager, but fortunately it is not really relevant how each distribution

<sup>1</sup> SLES starts six gettys on VTs 1-6, so the X server ends up on VT 7. RHEL and variants will not start a getty on VT 1 in runlevel 5, leaving VT 1 free for the X server. However, in runlevel 3 a getty is started on VT 1, so the X server ends up on VT 7.

---

does this. Most Linux systems will employ a display manager (covered in the next visual), which allows you to choose your window manager.

Since Linux has a large number of virtual terminals, there is nothing keeping you from starting a second X session on another virtual terminal. This is accomplished by starting an X server on display :1. When you start X through `startx`, you need to make sure that startx understands that this is an option not for itself, but for X, so the full startup line becomes `startx -- :1`.

Once you have started multiple X sessions, you can toggle between them with `Ctrl+Alt+F7` and `Ctrl+Alt+F8`.

***Instructor notes:***

**Purpose —** Discuss how to start X.

**Details —**

**Additional information —**

**Transition statement —** Let's also see how you can stop X.

## Stopping X

- Use **menu screens** from your window manager
  - Stops processes then stops X server
  - Usually saves current desktop layout
- If nothing works, use **Ctrl+Alt+Backspace**
  - Stops X server directly, and other processes lose connection and die
  - Can be disabled in X server configuration file

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-8. Stopping X

LX158.0

### Notes:

### Introduction

X can be stopped in two ways:

- The proper way, by using the appropriate button from your window manager. This gracefully stops all applications, and exits X.
- The quick and dirty way, by pressing Ctrl+Alt+Backspace. This first stops the X server, and then all applications ungracefully die because their connection is lost.  
Ctrl+Alt+Backspace can be disabled in /etc/X11/xorg.conf.

**Instructor notes:**

**Purpose** — Discuss how to stop X.

**Details** —

**Additional information** —

**Transition statement** — In the previous few visuals, you have seen how to start X and our window manager manually from a text screen. However, you can also have this done automatically through a session manager.

# Session managers

- Manage X-sessions
  - Start an X server
  - Offer a graphical login
  - Authenticate a user
  - Start the user's window manager
  - Wait until user logs out
  - Restart X server
  - Offer a graphical login screen for the next user
  - And so forth
- Different session managers:
  - **xdm**
  - **kdm**
  - **gdm**
- Started from **init** in runlevel 5 (RHEL) or as a regular System V service (SLES)

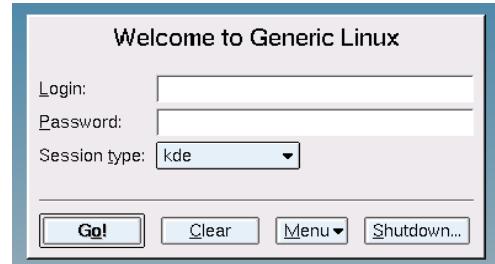


Figure 7-9. Session managers

LX158.0

## Notes:

### Introduction

A *session manager* is a program that manages X sessions. This means that it starts Xorg and displays a graphical login prompt. If a user tries to log in, the session manager authenticates this user and starts the user's favorite window manager. When the user logs out, the session manager restarts Xorg and displays a login prompt for the next user, and so forth.

On a Linux system, there are several different session managers available because nearly each desktop environment comes with its own session manager. The most common are **xdm**, **kdm**, and **gdm**.

The session manager is started from **init** in runlevel 5 (Fedora/RHEL) or with a regular System V service script in /etc/init.d (SLES).

***Instructor notes:***

**Purpose —** Discuss the session manager

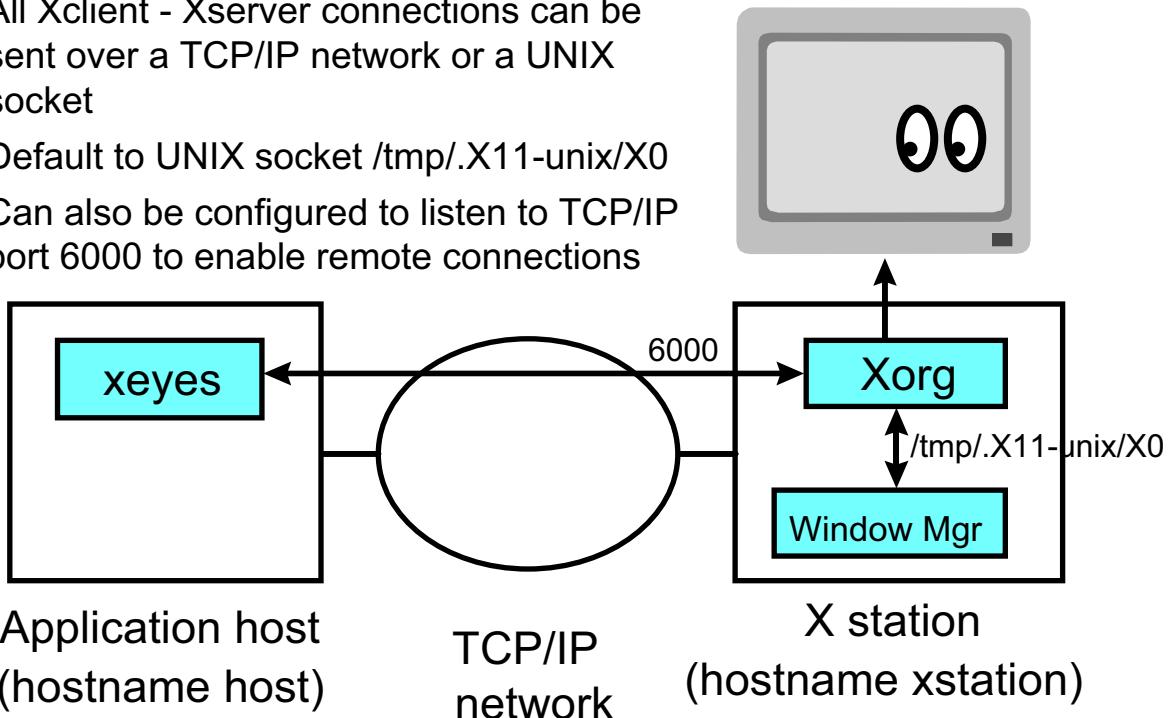
**Details —**

**Additional information —**

**Transition statement —** Because of the modular design, you can also run all of this over the network.

## X applications networked

- All Xclient - Xserver connections can be sent over a TCP/IP network or a UNIX socket
- Default to UNIX socket /tmp/.X11-unix/X0
- Can also be configured to listen to TCP/IP port 6000 to enable remote connections



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-10. X applications networked

LX158.0

### Notes:

#### Introduction

The visual shows the first level of networking X-applications. Both the Xorg server and the window manager (and possibly other applications as well) are running on the local system. Only a single application is running on the remote host (the application server).

Local connections between X clients and the X server are done through a UNIX socket, typically /tmp/.X11-unix/X0, while remote connections are done via a TCP/IP socket, typically on port 6000.

***Instructor notes:***

**Purpose** — Show how applications can be networked using X.

**Details** —

**Additional information** —

**Transition statement** — Let's see how you do this.

# Applications over TCP/IP

- On the host (where X clients run):

```
host$ xterm -display xstation:0.0
or:
host$ export DISPLAY=xstation:0.0
host$ xterm
```

- Displaying applications on a remote host is by default disabled for security reasons.
- To enable this, two methods possible: **xauth** and **xhost**
  - **xauth**: Uses cryptographic authentication method

```
xstation$ xauth extract xauthfile xstation:0.0
host$ xauth merge xauthfile
```

– **xhost**: Allows all connections from a given host

```
xstation$ xhost +host
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-11. Applications over TCP/IP

LX158.0

## Notes:

### Introduction

If you want to run an application from another server, then the only thing you basically need to do is start the application with a special option telling the application what X server to use.

This can be done using two methods:

- First, every X application will accept the **-display** option.
- Second, every X application will look at the **\$DISPLAY** environment variable to determine the X server to contact if no **-display** option is given.

The X server to contact is written as *hostname:servernumber[.displaynumber]*, with *hostname* being the IP address or hostname of the system where the X server is running, *servernumber* the instance of the X server to contact<sup>1</sup>, and *displaynumber* the screen to use.<sup>2</sup>

<sup>1</sup> One system might be running multiple servers, although this is rare.

<sup>2</sup> One X server can handle multiple screens simultaneously on so-called dual-headed systems.

You can imagine that it is not desirable that the whole Internet can redirect the graphical output of their commands to your screen. Therefore, doing this is by default disabled but can be enabled.

The first, safest method is by using the **xauth** mechanism. This works roughly as follows:

- When your X server is started, the startup scripts ensure that a random number, called the *authorization record* is generated. These records are stored in the \$HOME/.Xauthority file.
- Any client who wants to connect to the X server needs to present this authorization record. If no or an invalid authorization is presented, then access is disabled.
- Since normally all applications are started by the same person who started the X server, they all use the same .Xauthority file and present the right record.
- A client on a remote host obviously cannot access the .Xauthority file directly, so the authorization record needs to be transferred manually to that other host. This is a two-part process.

First, on the host where the X server is running, you need to extract the correct record from the .Xauthority file and store it in a file. This is done with the following command:

```
xauth extract xauthfile client:0.0
```

This means that the authorization record to connect to client:0.0 needs to be stored in the file xauthfile.

You then transfer the file to the other system (using FTP, secure copy (SCP), remote file copy (RCP) or any other means), and add it to the .Xauthority file there, with the following command:

```
xauth merge xauthfile
```

Any application started on this host, with the correct **-display** option or \$DISPLAY environment variable set now uses this authorization record to connect to the X server.

Of course, smarter ways of doing this are also possible. How about, for instance:

```
xauth extract - client:0.0 | rsh host xauth merge -  
rsh host xeyes -display client:0.0
```

The second method is less safe but more convenient. In this case, the user who has already started the X server issues the **xhost +hostname** command. This command allows all connections originating from *hostname* to succeed. This is obviously less secure, since every user on that particular host is now able to make a connection, not just the intended user. And this method is vulnerable to IP address spoofing and DNS poisoning.

---

In addition to **xauth** and/or **xhost** authentication, most distributions will also enable a firewall that blocks traffic to port 6000. You will need to open up this port, or disable the firewall altogether, to allow traffic.

Lastly, the **gdm** display manager by default starts **X** with a setting that disables TCP communications. To change this behavior, and thus enable TCP communications, add the following to the [security] section of /etc/gdm/custom.conf:

```
DisallowTCP=false
```

The **kdm** display manager has the same default setting. To enable TCP communications for the **X** server started from **kdm**, modify /etc/kdm/kdmrc and comment out the following line:

```
ServerArgsLocal=-nolisten tcp
```

***Instructor notes:***

**Purpose** — Show how applications can be used over TCP/IP.

**Details** —

**Additional information** —

**Transition statement** — Sounds extremely complicated, doesn't it? Fortunately, there's an easier and far more secure way to do this.

## SSH tunneling of X

- Secure shell (SSH) is the descendant of **rsh** and **rlogin**, which are non-encrypted programs for remote shell logins.
- OpenSSH is the most common free version of SSH and is available for virtually all UNIX-like operating systems.
- Because the SSH protocol encrypts everything it sends and receives, and allows tunnels to be setup, it can be used to secure otherwise insecure protocols.
- From the command line, use the **-X** option with **ssh** to setup a tunnel for X.

```
xstation$ ssh -X host
host$ echo $DISPLAY
localhost:10.0
host$ xeyes
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-12. SSH tunneling of X

LX158.0

### Notes:

#### Introduction

Secure Shell (SSH) is a secure implementation of the old **rsh** and **rlogin** programs. In its most basic usage, it allows you to connect to a remote system and start a shell on that system. You can then type commands.

All communication between the SSH client and server is encrypted, and authenticated using either the users password or a users public/private key pair. To prevent against man-in-the-middle attacks, all communication is also authenticated using a unique server public key pair.

The SSH protocol is far more advanced and complicated than just passing the encrypted stdin/stdout of the shell program to your local system. Among other things, it allows multiple protocols to be tunneled simultaneously over this single connection, and it is able to forward X authentication data. This makes it very convenient, and secure, to tunnel X applications over SSH.

## Setting up the connection

Tunneling X applications over SSH is called X Forwarding. You can enable it with the "-X" option when starting **ssh**, or make it default by modifying your \$HOME/.ssh/ssh\_config file, and adding the following line:

```
ForwardX11 yes
```

When X Forwarding is requested, **ssh** will do two important things:

- The **ssh** client on your local system will read the \$HOME/.Xauthority file, extract the proper keys, transfer these to the remote system, and add these to the .Xauthority file there.

This means that if any X client on the remote system connects to the local X server, it will be authenticated automatically, without the need for **xhost** settings or manually copying the .Xauthority data with **xauth**.

- The **sshd** daemon on the remote server will open a network port, typically 6010 or above, and will listen for any incoming traffic on this port. If data arrives, it is tunneled back via the SSH connection to the **ssh** client. This client will then forward it to the UNIX or TCP/IP socket identified in the **\$DISPLAY** variable.

On the remote end, the **sshd** daemon will ensure that the **\$DISPLAY** variable is set so that all X clients will try to connect to this port. A typical setting is "localhost:10.0".

**Instructor notes:**

**Purpose** — Show how ssh authentication can be used for security.

**Details** —

**Additional information** —

**Transition statement** — Okay, that completes our coverage of X. Let's also look at something called VNC, and how this integrates with X.



## 7.2. Virtual Network Computing

### Instructor topic introduction

**What students will do —** Learn how to setup VNC.

**How students will do it —**

**What students will learn —**

**How this will help students on their job —**

# Virtual Network Computing

- Method of allowing access to a remote graphical environment
  - Authentication
  - Efficient transfer of screen data
  - No strong encryption by default
- Typically used for cross-platform remote management
- Within Linux, two possibilities:
  - VNC access to Xorg-managed desktop: Integrated in Desktop Environment (KDE/GNOME)
  - Run separate VNCserver for each user (as :1, :2 and so forth)
- **vncviewer** connects to VNC server

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-13. Virtual Network Computing

LX158.0

## Notes:

Virtual Network Computing (VNC) is a standard method of getting access to a remote graphical environment. The standard protocol allows for authentication (using a password) and the efficient transfer of screen data (only updates are transmitted, for instance, not the whole screen). Commercial extensions do allow for stronger authentication and encryption, but these implementations may break compatibility with other implementations.

VNC is typically used for cross-platform remote management. For instance remote management of Windows servers or remote assistance of desktop users.

In a Windows environment, it allows access to the (single) desktop of the remote machine. However, since Linux is capable of running multiple X servers simultaneously, there are several possibilities:

- You can enable your current Desktop Environment to enable VNC communications. This allows you, or someone else, to connect to your local desktop via the network.
- You can run the VNC server. This program implements a full X server, but only supports the VNC protocol. It does not have direct access to hardware such as a graphical

---

adapter, mouse and keyboard, so you cannot use the VNC server locally (other than through a VNC client). The advantage is that you can run dozens, or even hundreds of VNC servers on a single machine. This essentially gives all your users access to a Linux desktop from their local (Windows) workstation, which might be a solution if you have a graphical application that needs to run on Linux.

Linux also has a VNC client, called **vncviewer**, which allows you to connect to any VNC-capable server.

***Instructor notes:***

**Purpose** — Introduce VNC

**Details** —

**Additional information** —

**Transition statement** — Okay, let's first see how we can enable our local desktop for VNC.

## Configure KDE/GNOME desktop for VNC

- Made possible by **vino** (GNOME), **krfb** (KDE) or **x11vnc** (X11)
- Once enabled, connect using web browser (port 5800) or VNC client (port 5900)



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-14. Configure KDE/GNOME desktop for VNC

LX158.0

### Notes:

The first method of supporting VNC is by enabling your local desktop for VNC. Depending on your desktop environment, this can be done with **vino** (GNOME) or **krfb** (KDE). If your desktop environment does not support VNC, you can also use the generic **x11vnc**, but this is harder to configure.

Configuration of **vino** and **krfb** is integrated in the setup menus of your desktop environment, typically called "Remote Desktop" or "Desktop Sharing".

Once VNC access is enabled, you typically get two access methods:

- You can connect the **vncviewer** to port 5900 (assuming this is the first desktop, ":0").
- You can point your web browser to port 5800. The web server that's integrated in the VNC code will then serve up a web page containing a Java-based VNC client.

Configuration of **vino** and **krfb** also allows you to set security settings, such as the password, and whether the desktop user should give permission every time someone connects via VNC.

**Instructor notes:**

**Purpose** — Discuss VNC enablement of the existing desktop.

**Details** —

**Additional information** —

**Transition statement** — Let's also look at the second method, VNCserver.

# Configure VNCserver

- Individual user:
  - Set VNC password in `~/.vnc/passwd` with **vncpasswd**
  - Start **vncserver**
- System-wide
  - Set all VNC passwords for all users
  - **RHEL:** Create `/etc/sysconfig/vncservers` listing all desktops; **service vncserver start**
  - **SLES:** Configure `/etc/xinetd.d/vnc` with all desktops; **service xinetd restart**
- In both cases, the VNCserver is a separate X server (`:1`, `:2`, ...), not connected to the main desktop (`:0`)
- Use **vncviewer host:1** to connect to VNC Server :1
- Use **http://host:5801** to connect to built-in webserver w/Java

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-15. Configure VNCserver

LX158.0

## Notes:

In addition to giving VNC access to your local, existing desktop, you can also start up a separate VNC server program which will act as a different X server. This X server will have its own state, resolution, window manager and so forth, and is completely independent of your default desktop.

In fact, you can even run the VNC server program on a system that doesn't have any graphical hardware at all. This is not very common in the Intel world, but very normal in embedded systems, or IBM Power and System z systems.

The easiest way of running **vncserver** is by simply starting it as a local user. For security, it is recommended to first set a VNC password in `$HOME/.vnc/passwd` with the **vncpasswd** command. You then start **vncserver**, which will look for the first free port above 5900, start the embedded X server, and start your desktop environment. What desktop environment is used is typically determined by the `/etc/X11/xinit/xinitrc` shell script. This script is distribution-dependent, but will typically honor the **\$DESKTOP**, **\$WINDOWMANAGER** or **\$DISPLAYMANAGER** shell variables.

For more extensive configurations, where you will want to support the desktops for multiple users on a single system simultaneously, most distributions have created scripts to start multiple **vncserver** sessions simultaneously.

On a **Red Hat** system, this works as follows:

- Set the password for all users that need access to VNC with the **vncpasswd \$USER/.vnc/passwd** command.
- Modify the file **/etc/sysconfig/vncservers** to include all users that need a VNC session, with their X display number (:1, :2 and so forth), plus set any VNC options. An example file would look like this:

```
VNCSEVERES="1:root 2:tux"  
VNCSEVERARGS [1]="-geometry 800x600"  
VNCSEVERARGS [2]="-geometry 1024x768"
```

This means that two instances of the VNC server will be started. Instance ":1" will be for user "root" and use a resolution of 800x600, while instance ":2" will be for user "tux" and use a resolution 1024x768.

There is a direct relation between the instance number (such as ":1") and the port numbers that are opened: The VNC interface for instance ":1" will always be at port 5901, while the web server interface (serving up a web page containing a Java VNC client) will always be at port 5801.

- Start the VNC server with the command **service vncserver start**

This will start a VNC server instance for the users identified in the **/etc/sysconfig/vncservers** file. After this, it will start the desktop environment of that user, by running the **\$HOME/.vnc/xstartup** shell script. This script, by default, will not start a complex desktop environment such as KDE or GNOME, but will use a relatively simple window manager such as **twm**. However, since this script is user-editable, you can configure your desktop environment and preferences yourself.

- Depending on your local settings and wishes, you may want to enable this at system restart with **chkconfig vncserver on**. You might also need to enable your firewall to allow incoming connections at TCP/IP ports 5801 and up, and 5901 and up.

On a **SUSE** system, the default setup is slightly more complicated. A SUSE system doesn't start a series of VNC server processes by default, but configures the **xinetd** daemon to listen on ports 5901 and up. When a VNC client connects to these ports, it will start **Xvnc** (the main component of **vncserver**) and will offer a login prompt (session manager) on this display. To enable this, do the following:

- Modify the file **/etc/xinetd.d/vnc** and enable (set "disabled" to "no") all relevant stanzas.
- Modify the file **/etc/sysconfig/displaymanager** and set **DISPLAYMANAGER\_REMOTE\_ACCESS** to "yes". You might also want to set **DISPLAYMANAGER\_ROOT\_LOGIN\_REMOTE** to "yes" if you want the root user to be able to login via VNC.

- Restart the display manager: **service xdm restart**

**Note:** xdm is only started in runlevel 5. On a system with no physical graphical hardware you might want to run VNC in runlevel 3. In that case, enable xdm to run in runlevel 3, but disable it to start on actual graphical hardware (the :0 instance).

- Restart the xinetd manager: **service xinetd restart**

You can now connect a VNC client.

## **Instructor notes:**

**Purpose —** Discuss **vncserver**

**Details —** This might be a good time to recap the :0, :1, :2 instance naming, which we've seen before when covering the startx -- :1 command.

The port numbers related to this are:

6000+n: Communication between X clients and X server (X11 protocol)

5900+n: Communication between VNC client and VNC server (VNC protocol)

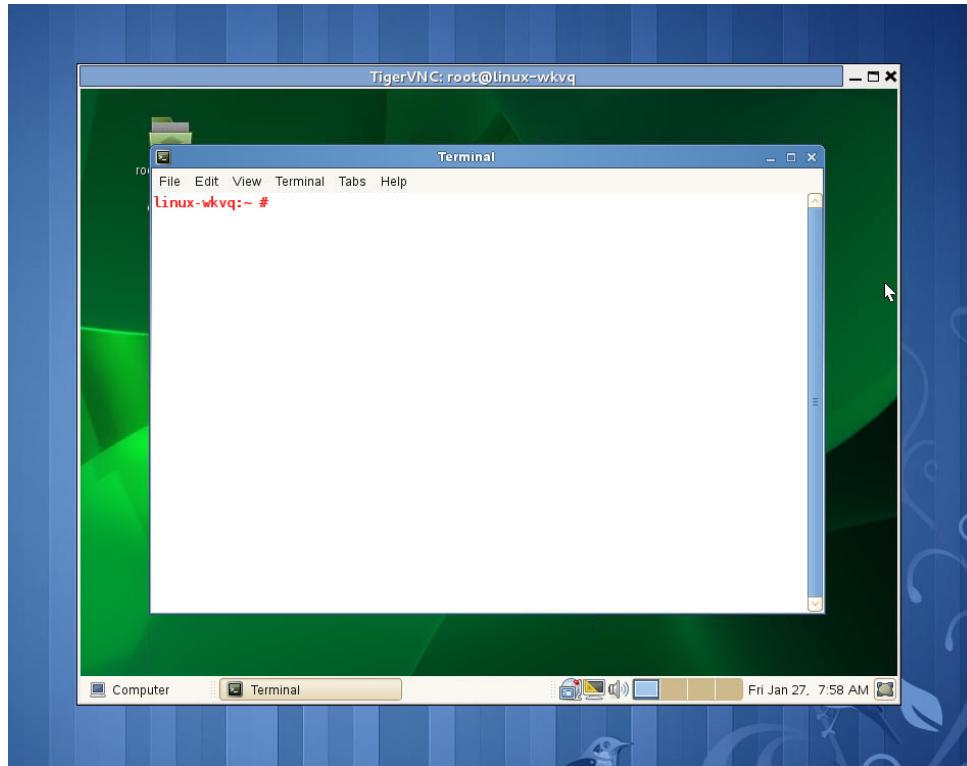
5800+n: Communication between a web browser and a built-in VNC server that serves up an HTML page that refers to a Java VNC client (HTTP protocol)

:0 is the default desktop, while :1 and further are additional desktops, for instance delivered by VNC or by starting a second X server on a free virtual terminal (Ctrl-Alt-F8 typically).

**Additional information —**

**Transition statement —** Lastly, let's look at the VNC client.

## VNC client



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-16. VNC client

LX158.0

### Notes:

Once VNC has been configured, you can access it with a VNC client. Under Linux, this is called **vncviewer**. It connects to a VNC server and will simply display whatever the VNC server wishes to display.

When required, **vncviewer** will ask for a password.

Once started, **vncviewer** will forward any mouse and keyboard events, and will receive any screen updates from the server to display.

***Instructor notes:***

**Purpose** — Discuss the VNC client

**Details** —

**Additional information** —

**Transition statement** — That's it. Checkpoint time.

## Checkpoint

1. What is the function of X.org?
2. What is the function of a window manager?
3. How do you run an individual X application over a network?
4. What port number do you need to connect to VNC server instance :4 via VNC?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-17. Checkpoint

LX158.0

### Notes:

### **Instructor notes:**

**Purpose** — Check student's knowledge of presented materials.

**Details** —

## **Checkpoint solutions**

---

1. What is the function of X.org?

The answer is it is the graphical user interface for UNIX/Linux.

2. What is the function of a window manager?

The answer is it allows more control for your windows space.

3. How do you run an individual X application over a network?

The answer is set the DISPLAY variable (or option) and enable authentication with xauth or ssh.

4. What port number do you need to connect to VNC server instance :4 via VNC?

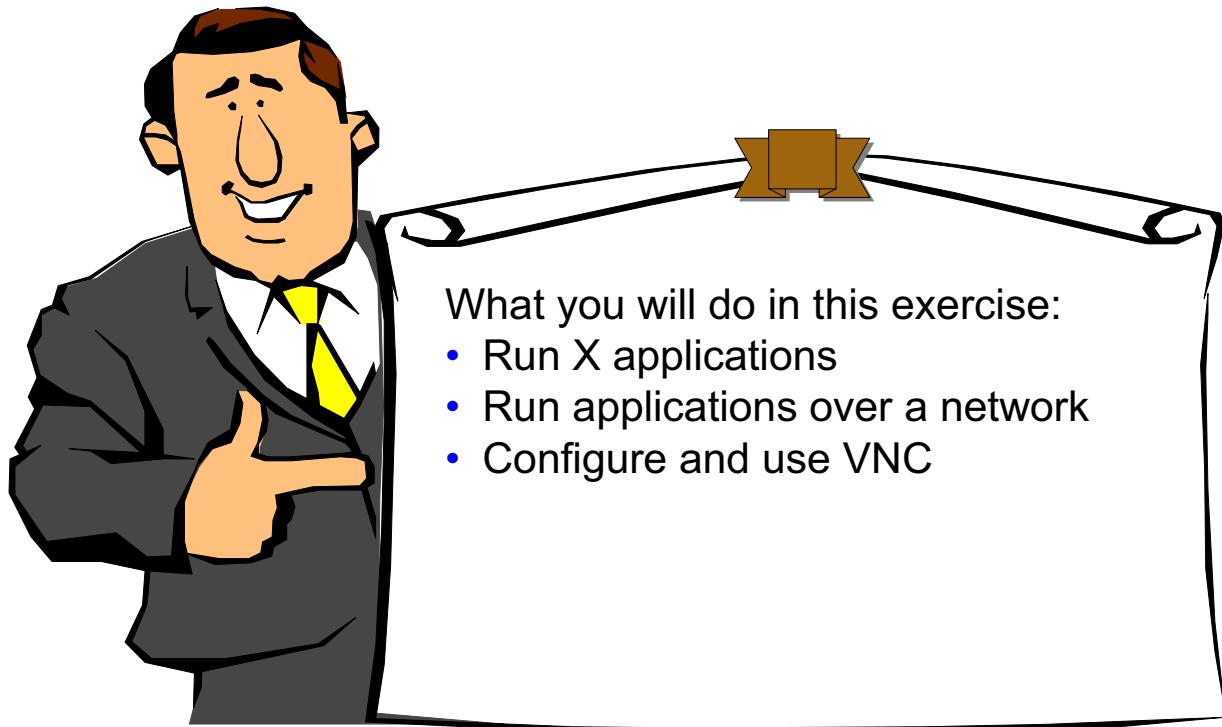
The answer is 5904.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

### **Additional information** —

**Transition statement** — Let's use what we learned in a class exercise.

## Exercise: X Window System and VNC



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-18. Exercise: X Window System and VNC

LX158.0

### Notes:

***Instructor notes:***

**Purpose —** Hand off students to exercise.

**Details —**

**Additional information —**

**Transition statement —** Now, let's summarize what we covered in this unit.

## Unit summary

Having completed this unit, you should be able to:

- Describe the basic architecture of the X Window System
- Configure X.org
- Start and stop X
- Describe the function of the window manager
- Use X over a network
- Describe the VNC protocol
- Activate VNC from your Desktop Environment
- Configure and start the VNC Server

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 7-19. Unit summary

LX158.0

### Notes:

Having completed this unit, you should understand:

- The X Window System is the graphical user interface for Linux (and other UNIX-based systems)
- You can start and stop X with startx and **Crtl+Alt+BackSpace**
- Window managers make the GUI user friendly
- You can use X over a network safely with ssh
- Remote access/management is made easy using VNC

***Instructor notes:***

**Purpose —** Summarize unit material.

**Details —**

**Additional information —**

**Transition statement —** Now, we can move on to the next unit.

# Unit 8. User administration and security

## Estimated time

00:45

## What this unit is about

This unit describes how users and groups can be managed on the system, and how to secure your user/groups setup.

## What you should be able to do

After completing this unit, you should be able to:

- Add, change and delete user accounts
- Add, change and delete user groups
- Manage user passwords
- Communicate with the user community
- Describe the concepts of PAM
- Define ways of controlling root access to the system
- Define the use of SUID, SGID, and sticky bit permissions bits
- Configure Access Control Lists
- Identify the data files associated with users

## How you will check your progress

Accountability:

- Checkpoint questions
- Machine exercises

## Unit objectives

---

After completing this unit, you should be able to:

- Add, change and delete user accounts
- Add, change and delete user groups
- Manage user passwords
- Communicate with the user community
- Describe the concepts of PAM
- Define ways of controlling root access to the system
- Define the use of SUID, SGID, and sticky bit permissions bits
- Configure Access Control Lists
- Identify the data files associated with users

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose —** Define unit objectives.

**Details —**

**Additional information —**

**Transition statement —** Let's begin with a discussion of basic security concepts.



## 8.1. User administration

### Instructor topic introduction

**What students will do —**

**How students will do it —**

**What students will learn —**

**How this will help students on their job —**

# Security concepts



- Unique name
- Unique ID (UID)
- Password
- File ownership is determined by user ID
- Unique name
- Unique ID (GID)
- Users who need access to the same files

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-2. Security concepts

LX158.0

## Notes:

### Introduction

The security of a Linux system is based on a user being assigned a unique name, user ID (UID), and password. When a user logs in, the UID is used to validate all requests for file access. When a file is created, the UID associated with the process that created the file is assigned to the file. Only the owner or root can change the access permissions. Users that require access to a set of files are placed in groups. A user can belong to multiple groups. Each group has a unique name and Group ID (GID). Every user will always be member of at least one group. This is called the *primary group* and usually has the same name as the user name. In addition to that, users can also be members of other groups. These are called *secondary groups*.

**Instructor notes:**

**Purpose** — Highlight some basic concepts concerning Linux security.

**Details** — Linux security can be defined by two basic principles:

1. Ownership of data controls access
2. Permissions or access to the data is granted by the owner to other users

Rather than have individually nominated users that can access a piece of information, Linux provides the concept of a group of users.

All members of a group have something they share in common. In a human sense, they need to access/use a set of data. In a machine sense, they belong to a group.

**Additional information —**

**Transition statement** — Since the root user is such a powerful user account, it is a good idea to close up this account as much as possible.

# User hierarchy

- **root**
  - Super user
  - File permissions do not apply for root
  - Can do anything except the obvious
  - Account for the system administrator
- **bin, daemon, lp, sync, news, ftp ...**
  - User accounts used by different applications and daemons
  - Cannot (and should not) be used to log in
- Ordinary user accounts

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-3. User hierarchy

LX158.0

## Notes:

### Introduction

The most important user (from a system administrative point of view) is the **root** user. The file permissions do not apply to root, so he or she can read, change, and delete any file he wants to. In fact, root can do just about anything, except for obvious things like writing to read-only mounted file systems (CD-ROM), unmount busy file systems, and so on. Furthermore, most system administration tasks can only be executed by the root user.

### Special user accounts

Besides the root user, Linux has a number of other default users. These users should not be used to login but are there for the convenience of some applications and daemons.

These users should not be used to carry out any administration task; use the root user for this.

---

Some examples of special user accounts include:

- daemon** – Used to execute system server processes.
- bin** – Owns the executable files for most user commands.
- sys** – Owns system files.
- adm** – Owns accounting files.
- news** – Pseudo-user for news service.
- lpd** – Pseudo-user for print subsystem.
- mail** – Pseudo-user for mail service.
- cron** – Pseudo-user for job scheduler.
- auth** – Pseudo-user associated with system audit facility.
- nobody** – Used by NFS server

### Standard users

The last type of user account is the normal user account. The purpose of these accounts is to give ordinary users the opportunity to login to a Linux system and carry out tasks.

**Instructor notes:**

**Purpose** — Describe the purpose of the different user accounts.

**Details** — In discussions concerning security, the root user is the exception to most, if not all, restrictions that are applied to the system.

Introduce the concept of having your own user account, even as a system administrator. A system administrator should never log in as root directly, but always as himself (or herself).

**Additional information —**

**Transition statement** — Let's take a look at groups.

# Groups

- A group is a set of users, all of whom need access to a given set of files.
- Every user is a member of at least one group and can be a member of several groups.
- *Primary group*: Used for file/directory creation.
- *Group set*: Used to determine access permissions.
- The user has access to files in all of the groups in its groupset. The **groups** command shows all the groups a user is member of.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-4. Groups

LX158.0

## Notes:

### Introduction

The creation of groups to organize and differentiate the users of a system or network is part of system administration. The guidelines for forming groups should be part of the security policy. Defining groups for large systems can be quite complex, and once a system is operational, it is very hard to change the group structure. Investing time and effort in devising group definitions before your system arrives is recommended.

### Group definitions

#### User groups

User groups should be made for people who need to share files on the system, such as people who work in the same department or people who work on the same project.

#### System-defined groups

The system-defined groups are used to control certain subsystems. There are two different kinds of groups available to users. The first group is the primary group. The primary group is used by the system when you create a file (and directory).

Every file created is assigned a group and this is the primary group of the user creating the file. The group set is the set of groups determining the permissions you have on a given file or directory. The group set is used by the system when you want to work with a file or directory.

**Instructor notes:**

**Purpose** — Describe in more detail the concept of a group.

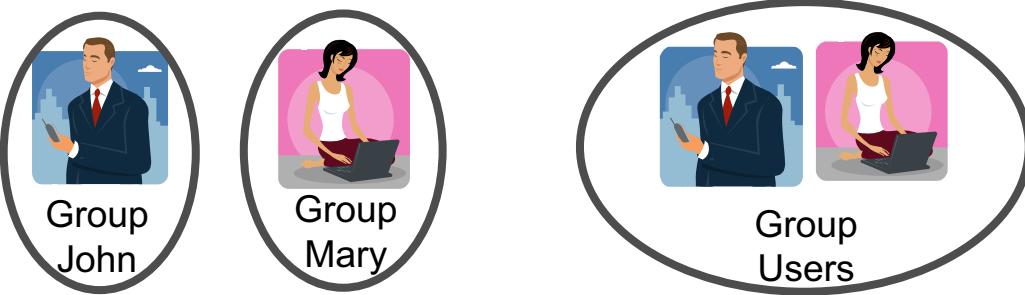
**Details** — The permissions in each group in the groupset are available to the user. If you belong to the groups project and admin, you can work with files belonging to the project group and also to the admin group according to the permissions specified for the files in the group field.

**Additional information —**

**Transition statement** — Every user is a member of at least one group. Let's see what this group actually is.

## User Private Groups

- User Private Groups: Scheme where every user has its own private group as primary group, instead of one big, generic group of which everyone is member
- Advantages:
  - Easier to give users access to home directories of other users (for example, secretary to boss' home directory)
- Disadvantages:
  - Requires changes to authorization subsystem (for example, **umask**, **useradd**, ...)
- RHEL/Fedora uses User Private Groups, SLES does not.



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-5. User Private Groups

LX158.0

### Notes:

#### Introduction

In the previous visual we have seen that every user is a member of at least one group. By default, in most UNIX and Linux systems, this is a generic group called *users* or *staff*. However, some distributions, including RHEL/Fedora, have introduced something called *User Private Groups*.

#### Group scheme

With this scheme, a group is created for each and every user account. This account is made a member of that group. The user name and group name are the same, as are the UID and GID numbers. This has an advantage over the traditional scheme in that it is easier to give someone (for instance, a secretary) access to someone else's home directory (for instance, the boss' directory): Simply make the secretary member of the boss group. With the traditional scheme, still used by SLES, this is virtually impossible. Also, note that this scheme still allows all the things the traditional scheme allows as well: groups related to a project, where every project member is member of the group and can access the files of that group.

***Instructor notes:***

**Purpose** — Introduce the concept of User Private Groups.

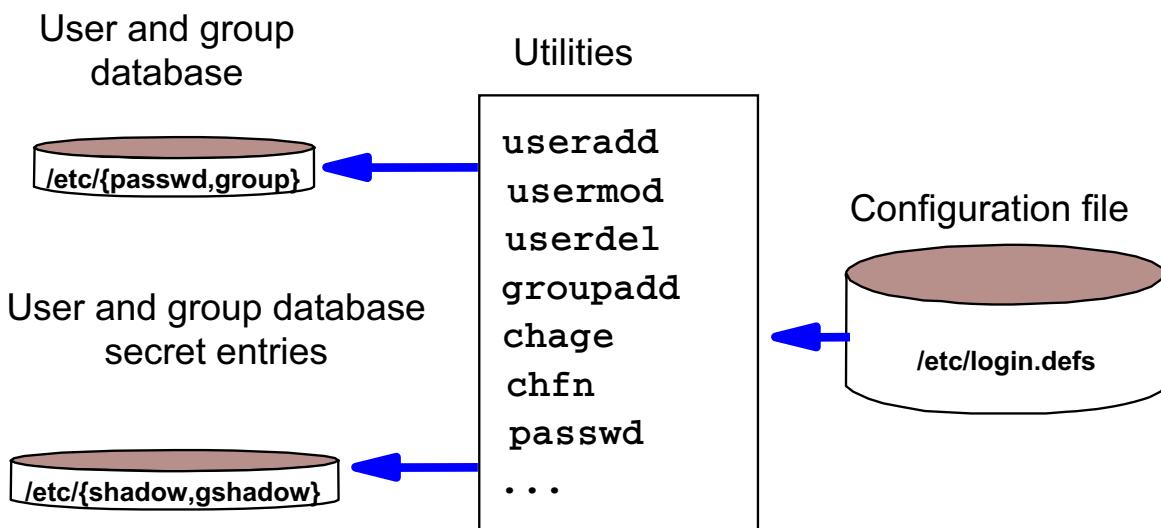
**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Another conceptual thing we need to cover here is the Shadow Password Suite.

## Shadow password suite

- Local users and groups secret information (passwords, ...) are managed by the shadow password suite.



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-6. Shadow password suite

LX158.0

### Notes:

#### Introduction

In the early days of UNIX, all user information, including the encrypted password, was stored in `/etc/passwd`. This file needs to be readable for the whole world: programs such as `ls`, for instance, need to be able to perform `UID <-> username` mapping.

#### Security concerns

This meant that every user on the system could get a list of all the encrypted passwords of all users, which he or she could then subject to a dictionary attack. When CPUs were comparatively slow by today's standards, this was a lot of work and not really practical.

Today, however, dictionary attacks take mere seconds, and with hardware which is currently available to wealthy hackers, a brute force attack which tries out every possible password is becoming feasible. It is therefore of paramount important that also the encrypted passwords of the users are shielded from ordinary users. This is performed by the shadow password suite. This suite of programs and libraries adds two additional files to the system: `/etc/shadow` and `/etc/gshadow`. These files are read-write only for root, so

---

ordinary users cannot get access to them, except for a few carefully written SUID programs that are part of the shadow password suite. The shadow password suite also implements password aging: a mechanism that forces the user to change his/her password every now and then. These parameters are stored in /etc/login.defs.

**Instructor notes:**

**Purpose** — Discuss the shadow password suite.

**Details** — Remind students of the permission structure of the /etc directory and that traditional security settings are compromised by this. Shadowing is used in a number of UNIX-based operating systems, so your students probably have some experience with this concept.

**Additional information —**

**Transition statement** — Okay, now let's see how to create these user accounts.

## Command line user tools

- Add a user account:

```
# useradd -m -g staff -G audio,uucp tux20
# passwd tux20
```

- Delete a user account:

```
# userdel -r tux20
```

- Change a user account:

```
# usermod -g users -G video tux20
```

- Locking and unlocking a user account:

```
# usermod -L tux20
# usermod -U tux20
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-7. Command line user tools

LX158.0

### Notes:

**adduser or useradd:** The adduser and useradd commands only create the user account. You have to set the password manually afterwards. Depending on the configuration in /etc/login.defs, useradd creates the home directory of the user as well. To always create the home directory, regardless of these settings, use the **-m** option.

**userdel:** Removes users from your system. The **-r** option also removes the contents of the user's home directory and the directory itself.

**usermod:** Changes settings of a user. This command can also be used to lock and unlock a user account. This is done by putting an exclamation point in front of the password in /etc/shadow.

***Instructor notes:***

**Purpose** — Show some other tools for user administration.

**Details** — Point out the visual is an example only. The groups staff and audio do not exist on Linux by default, so the first command example would fail on the student's lab system.

**Additional information —**

**Transition statement** — After the users, let's take a look at groups.

## /etc/skel

- Directory with skeleton files that users will receive in their home directory upon creation of their account
- **useradd -m** creates the home directory with files from /etc/skel
- **useradd -m -k** allows you to specify a different skeleton directory

```
# useradd -m tux25
# useradd -m -k /etc/my_own_skel tux30
```



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-8. /etc/skel

LX158.0

### Notes:

#### Introduction

When a user logs in, the shell will try to read some configuration files from its home directory. These files can be made manually by the root user or by the user itself, but they can also be copied automatically to the home directory of the user. The /etc/skel directory is the directory that contains a number of skeleton files. These files are copied to the home directory of a user when this user account is first created.

**Instructor notes:**

**Purpose** — Explain the /etc/skel directory.

**Details** — The /etc/skel directory contains skeleton files for the user. These files are copied to the user's home directory when the user account is created. If you want every user you create to have a special file in their home directory, place this file in the /etc/skel directory. This file will get copied to the home directory when you create the user.

**Additional information —**

**Transition statement** — Let's take a look at the information that is stored for a user.

## Command line group tools (1 of 2)

- List available groups:

```
# groups  
# groups tux1
```

- Add, delete, or change groups:

```
# groupadd penguins  
# groupmod -n oldname newname  
# groupdel penguins
```

- Add or delete users to/from groups:

```
# usermod -G penguins tux1 tux2  
# gpasswd -a tux1 penguins
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-9. Command line group tools (1 of 2)

LX158.0

### Notes:

#### Introduction

You could also use command line tools to manage your groups. An interesting feature of Linux is that you, as the superuser, can assign group administration rights to other users. This allows group administrators to add users to their group and remove them from their group. Remember, the user accounts themselves still need to be created by the superuser.

***Instructor notes:***

**Purpose** — Discuss the command line tools for administering groups.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — There are a few more useful command line tools.

## Command line group tools (2 of 2)

- Defer administration of a group to a user:

```
# gpasswd -A linus penguins
linus$ gpasswd -a tux1 penguin
linus$ gpasswd -d tux2 penguin
```

- Switch the default group used in file creation:

```
# newgrp penguin
```

- Edit the etc/group file:

```
# vigr
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-10. Command line group tools (2 of 2)

LX158.0

### Notes:

More command line tools for group management and administration. The /etc/group file will be discussed shortly.

***Instructor notes:***

**Purpose** — Discuss the command line tools for administering groups.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Let's take a look at passwords.

# Passwords

- Change a user's password with **passwd**
- Generate a random password for a user with **mkpasswd**
- Change a user's password expiry information with **chage**

```
# passwd tux30
New password:
...
# mkpasswd tux30
VjOmnoYXyPP4U
# chage -l tux30
Minimum:           14
Maximum:          186
Warning:            21
Inactive:           7
...
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-11. Passwords

LX158.0

## Notes:

### Introduction

Users can change their passwords by using the **passwd** command. Root can also use this command to reset passwords of other users. A useful tool is **mkpasswd**. This generates a random password and, optionally, assigns this password to a user. Note that the **mkpasswd** command is not installed by default. On a RHEL/Fedora system, it is part of the **expect** RPM, while on a SLES system, it is part of the **whois** RPM. Another useful tool is **chage**. This allows you to view and change the password aging information.

### **Instructor notes:**

**Purpose** — Explain the **passwd** command.

**Details** — The **passwd** command is used to reset the passwords of users. Ordinary users can use the **passwd** command to reset their own password. Root can use this command to reset the passwords of all users.

**Additional information** —

**Transition statement** — Let's see where the user and group information is stored.

## /etc/passwd

- The /etc/passwd file contains non-secret information about the users.

```
# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
...
postfix:x:51:51:Postfix Daemon:/var/spool/postfix:/bin/false
...
tux30:x:537:100::/home/tux30:/bin/bash
```

Fields are separated by ":"

- 1) User name
- 2) Password (x means encrypted password available)
- 3) UID
- 4) GID
- 5) GECOS (user information)
- 6) Home directory
- 7) Login shell

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-12. /etc/passwd

LX158.0

### Notes:

#### Introduction

Most user information is stored in /etc/passwd. It contains a line for each user, and values on the line are separated by colons.

From left to right, each line consists of:

1. The user name.
2. An “x”, meaning that the encrypted password is stored in /etc/shadow.
3. The user ID (UID) of the user.
4. The primary group ID (GID) of the user.
5. The full name of the user. Some system administrators also choose to include location, room number, telephone numbers and so forth in this field.
6. The home directory of the user.
7. The preferred shell of the user.

This file is world readable, meaning that everyone can read (but not write) to this file.

***Instructor notes:***

**Purpose** — Show /etc/passwd.

**Details** — Step through each value with class.

**Additional information —**

**Transition statement** — Passwords are stored in /etc/shadow.

## /etc/shadow

- Credentials of any user account are stored in the /etc/shadow file.

```
# cat /etc/shadow
...
bin:*:10787:0:99999:7:-1:-1:
...
tux1:$1$VOHuuCQM$Kqc9m7wS1QnRtqANtZCba/:10792:0:99999:0:0:13453
tux2:$1$BgSP6XLW$/tDKJTmLZzqh9372X7U7o:10791:-1:99999:-1:-1:13544
```

- 1) Login name
- 2) Encrypted password (MD5)
- 3) Last change of credentials (days since Jan 1, 1970)
- 4) Days before password can be changed
- 5) Days after which password must be changed
- 6) Days before password is to expire that user is warned
- 7) Days after password expires that account is disabled
- 8) Days since Jan 1, 1970, that account is disabled

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-13. /etc/shadow

LX158.0

### Notes:

#### Introduction

The passwords of the users are stored in /etc/shadow. This file contains, from left to right:

1. The user name.
2. The MD5 encrypted password of the user. MD5 encryption is a one-way encryption, meaning that once encrypted, a password can never be decrypted. To test whether an entered password is correct, the entered password is encrypted too and compared to the encrypted password in /etc/shadow. MD5 encryption is rather new. Older UNIXes, and other Linux distributions might still be using the old **crypt** algorithm. The real advantage of MD5 is that the allowed password length is increased from 8 to 256 characters.

Red Hat uses SHA by default since RHEL6. This generates a 100+ character encrypted password.

**Note**

A “\*” means that this user does not have a password. That user account can therefore not be used to log in.

3. The day the password was last changed (number of days since Jan. 1st, 1970).
4. Number of days before the password can be changed again.
5. Number of days after which the password has to be changed again.
6. Number of days the user will be warned of a password expiry.
7. Number of days after expiry, after which the account is disabled.
8. The day the account was disabled.

The /etc/shadow password file should be read/writable by root only. Other users should not be able to read this file at all.

***Instructor notes:***

**Purpose** — Discuss the contents for the /etc/shadow file.

**Details** — Go through each of the items on the list.

**Additional information —**

**Transition statement** — Let's view the last file, /etc/group.

## /etc/group and /etc/gshadow

```
# cat /etc/group
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
...
penguins:x:500:linus,tux1,tux2
tux1:x:501:
tux2:x:502:

# cat /etc/gshadow
...
penguins:!::linus:tux1,tux2
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-14. /etc/group and /etc/gshadow

LX158.0

### Notes:

#### Introduction

The /etc/group file contains group information. The file contains, from left to right:

- The group name.
- The group password. This is set to “x” if the group password is in /etc/gshadow.
- The group ID (GID).
- The list of users that have this group as their secondary group.

The /etc/gshadow file contains extended group information. From left to right:

- The group name.
- The group password. Note that the group password is an old UNIX concept which is seldom used today. For backwards compatibility, this field is kept alive.
- The name of the group administrator.
- The list of users that have this group as their secondary group.

## **Instructor notes:**

**Purpose —** Show the contents of **/etc/group**.

**Details —** The reason to have passwords on groups is to give users the opportunity to become temporarily members of another group. After issuing the **newgrp** command and entering the correct password, the user is member of the new group. He stays a member of this group until the user runs the **newgrp** command or until the user logs out.

## **Additional information —**

**Transition statement —** Now that we have set up the users and groups, let's look at the initialization process.

## /etc/issue and /etc/issue.net

- Contain the login message for mingetty and telnetd

```
# cat /etc/issue
```

```
Welcome to Generic Linux 1.0
Kernel \r on an \m
```

- Several backslash escaped sequences are supported by **mingetty**:
  - \r - kernel release
  - \m - machine type
  - \o - domain name
- See the mingetty manual page for a complete list.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-15. /etc/issue and /etc/issue.net

LX158.0

### Notes:

#### Introduction

The /etc/issue and /etc/issue.net (not available in SLES) files contain the login message shown at login time. The /etc/issue file is shown by the **mingetty** process, and /etc/issue.net is shown by the telnet server when a client logs in over the network. The /etc/issue file only appears on an ASCII screen, not on a GUI window. The /etc/issue and /etc/issue.net files can contain escape sequences: a backslash followed by a single character. These escape sequences are then replaced with dynamic information such as the date, the architecture, and the kernel version when the file is displayed. For a list of these escape codes, see **man mingetty**.

**Instructor notes:**

**Purpose** — Show the contents of these files.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — After login, the message of the day is displayed.

## Message of the day

- /etc/motd
- Should only contain information necessary for the users to see

```
# cat /etc/motd

***** SYSTEM OUTAGE *****
Due to a hardware upgrade this system will not
be available between 10pm and 11pm tonight.
*****
```

- If \$HOME/.hushlogin exists, /etc/motd will not be shown when the user logs in.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-16. Message of the day

LX158.0

### Notes:

#### Introduction

The message of the day (**motd**) is stored in /etc/motd. Under normal conditions, users do see the contents of this file on their screen when they login. Users who login graphically do not see the message of the day. The .hushlogin file is used to disable the **motd** facility. When you create this file in your home directory (it might be an empty file), you do not see the motd at login times anymore.

### **Instructor notes:**

**Purpose** — Explain the /etc/motd file.

**Details** — The /etc/motd file is a way to communicate information to users. For instance, a message like Next Friday, the system will be down from 3 p.m. to 6 p.m. for maintenance could be put in the motd file.

**Additional information** — The .hushlogin file is very useful for people using uucp.

**Transition statement** — Let's talk about user security next.

## 8.2. User-level security

### Instructor topic introduction

**What students will do —**

**How students will do it —**

**What students will learn —**

**How this will help students on their job —**

## User-level security overview

- *Authentication*: Verifying that you are who you say you are
- Can be based on:
  - Something you only know (for example, password, PIN)
  - Something you only have (for example, smartcard, token, key)
  - Something you only are (for example, fingerprints, retina scan)
- *Authorization*: Determining your level of access
  - File permissions, ACLs
  - Account restrictions (login times, login tty, and so forth)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-17. User-level security overview

LX158.0

### Notes:

#### Introduction

Security issues that surround users that log into a system are handled under the area of user-level security. Two steps are required to secure system resources.

- Authentication
- Authorization

#### Authentication

The first step is *authentication*. Authentication means: verifying that you indeed are who you say that you are. In theory, there are several methods of achieving this:

- By showing that you know something, such as a password or PIN code
- By showing that you have something, like a smart card, ATM card, key or token
- By showing that you are something, for instance, by using biometric data such as finger prints, retina scans and so forth

## Authorization

The second step is *authorization*. Authorization means that we have established that you are who you say that you are, but need to determine what you're allowed to do on the system. This is implemented in Linux using file permissions.

***Instructor notes:***

**Purpose** — Define user-level security for students.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — First, let's look at authentication.

# Pluggable Authentication Modules

- Authentication system of Linux
- Implemented as a suite of shared libraries
- Enables the system administrator to choose how applications authenticate users
- Provides flexibility to applications that need authentication
- Initially developed by Sun Microsystems
  - Adapted for Linux

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-18. Pluggable Authentication Modules

LX158.0

## Notes:

### Introduction

Linux uses a subsystem called Pluggable Authentication Modules (PAM) to authenticate users. It is very flexible due to its modular design, allowing the system administrator to customize how applications authenticate users. It also allows for a standard way for applications to authenticate users without needing application changes for new authentication methods.

### Shared libraries

PAM is implemented as a suite of shared libraries that are used by the different programs that need authentication services.

### Sun Microsystems

PAM was initially developed by Sun Microsystems but later adapted for Linux.

***Instructor notes:***

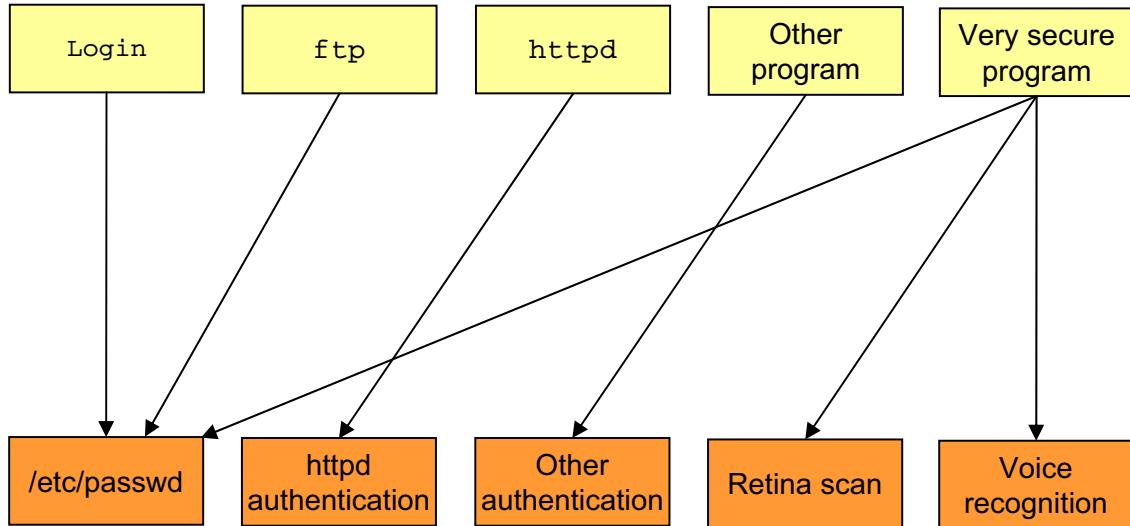
**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

# Authentication before PAM



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
 US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-19. Authentication before PAM

LX158.0

## Notes:

### Introduction

For a system administrator, the situation before PAM was far from ideal. Every application that ran on a system required its own security and authentication mechanism. Some of them were based on `/etc/passwd`, `/etc/group`, and `/etc/shadow`, like **login** and **ftp** (although **ftp** also knew the “anonymous” login possibility), and others used their own authentication mechanisms. A program which was supposed to be very secure might actually employ a layered approach, maybe incorporating biometric authentication techniques like retina scans or voice recognition. All these different authentication mechanisms are a nightmare for system administrators, because if the administrator wants to add a user, he has to do that in multiple places. Plus, the system administrator was not free to choose his own method. Suppose, for instance, that a university decides to supply all students with a chipcard which is used for the restaurant, the library and the computer facilities as the authentication device. With a scheme like this, even if you managed to implement it correctly, adding another form of authentication somewhere would cause another large piece of work.

***Instructor notes:***

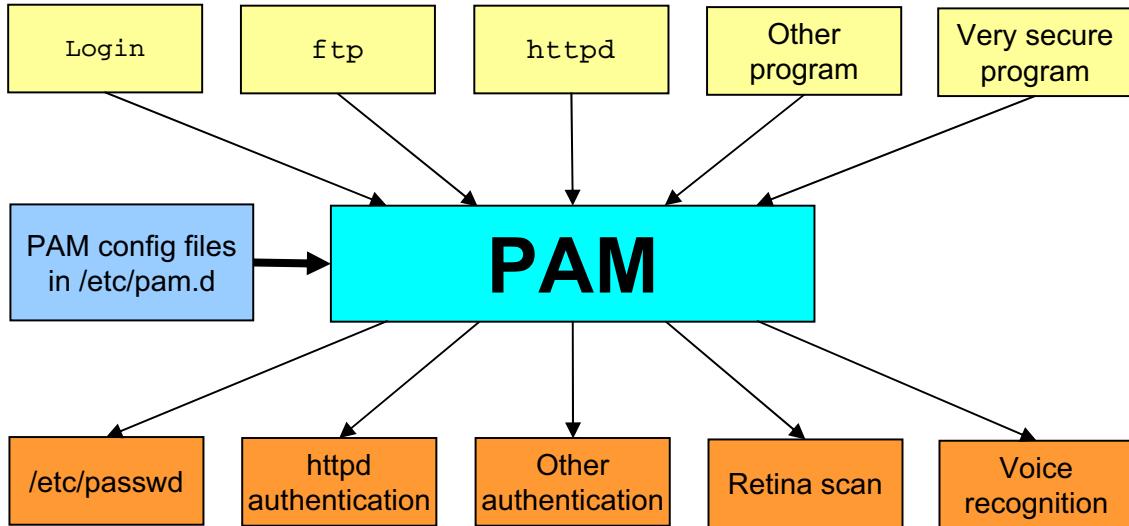
**Purpose** — To show students how authentication was done prior to the use of PAM.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Now, let's look at how authentication is done with PAM.

# Authentication with PAM



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-20. Authentication with PAM

LX158.0

## Notes:

### Introduction

With PAM, every application that needs some kind of authentication, needs to be rewritten to use the PAM authentication mechanisms. Then, the only thing that program has to do is ask PAM, "Is this user authorized to use me?" PAM will tell the program yes or no. To authenticate that user, the system administrator can set up different authentication mechanisms and specify which program should use which kind of authentication mechanism. There are a number of authentication mechanisms currently available. Some of the more important are:

- User ID/password checking
- Anonymous login (for example, for anonymous ftp)
- Deny, for services that cannot be used
- Secure TTY, meaning that logging in is only allowed from a secure terminal

Of course, PAM allows the system administrator to add its own mechanisms, like retina scans, voice recognition, fingerprint readers, chipcard readers, time-driven mechanisms

(only allowed to login during office hours) and so forth. Which service uses which authentication mechanism is specified in configuration files in /etc/pam.d. There is one configuration file for each service, and there is a default configuration file, called other, which is used when a specific configuration file is not available.

***Instructor notes:***

**Purpose** — To shows students how PAM is used to do authentication.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Next, let's look at a PAM configuration file.

# PAM configuration file example

```
# cat /etc/pam.d/login
#%PAM-1.0
auth      required  /lib/security/pam_securetty.so
auth      required  /lib/security/pam_unix.so likeauth
auth      required  /lib/security/pam_nologin.so
auth      required  /lib/security/pam_env.so
account   required  /lib/security/pam_unix.so
password  required  /lib/security/pam_cracklib.so retry=3
password  required  /lib/security/pam_unix.so nullok
session   required  /lib/security/pam_limits.so
session   required  /lib/security/pam_unix.so
session   optional   /lib/security/pam_console.so
```

**Note:** PAM configuration is different from distribution to distribution.

Please see the Notes area, below:

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-21. PAM configuration file example

LX158.0

## Notes:

### Introduction

The visual above shows an example PAM configuration file. Every file you will encounter within PAM is split up in four sections, which apply to the four phases of the login process:

1. **auth:** Verify the authentication of the user, usually by checking the password.
2. **account:** Manage the account. For instance, force a user to change its password if the password used is expired.
3. **password:** Change the password itself. This phase can also be called from the passwd program.
4. **session:** Manage the session where the user logged in.

### Example

Using the example shown on the visual, from top to bottom, the lines mean roughly:

- When the user tries to authenticate, perform the following checks:

- Require that root only logs in from a TTY listed in /etc/security.
- Check that the UNIX password is correct.
- Do not allow a regular user in if the file /etc/nologin exists. Instead, print the contents of the file /etc/nologin.
- Set a number of environment variables used in the login process.
- For the account management, only perform the regular UNIX checks of password expiration.
- When a user sets a password, perform a dictionary attack first. Then, set the password using the regular UNIX files.
- When the users session is set up, apply a number of limits (CPU, memory, ...), and perform standard UNIX login tasks, such as switching to the appropriate user ID. If the user logs in on the console, make the user owner of certain console devices such as /dev/cdrom.

More information on PAM can be found in /usr/share/doc/pam-version. This includes a description of every function of every PAM module.

**Note:** The file in the visual is not an actual file, but merely an example.

### RHEL/Fedora

Some actual examples are:

```
# cat /etc/pam.d/login:  
#%PAM-1.0  
auth      [user_unkown=ignore success=ok default=bad pam_security.so  
auth include system-auth  
auth      account required pam_nologin.so  
account   required system-auth  
password  include system-auth  
# pam_selinux.so close should be the first session rule  
session   required  pam_selinux.so close  
session   include system-auth  
session   required  pam_loginuid.so include system-auth  
session   optional   pam_console.so  
# pam_selinux.so open should be the last session rule  
session   required  pam_selinux.so open  
session optional pam_keyinit.so force revoke
```

As you can see, RHEL/Fedora uses the pam\_stack.so module to refer to a generic system-auth file. This file is modified by authconfig and used in virtually any PAM authentication configuration file.

The contents of the system-auth file:

```
# cat /etc/pam.d/system-auth:
```

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.

auth    required  pam_env.so
auth    sufficient pam_unix.so nullok try_first_pass
auth    requisite  pam_succeed_if.so uid >= 500 quiet
auth    required  pam_deny.so
account required  pam_unix.so
account sufficient pam_succeed_if.so uid < 500 quiet
account required  pam_permit.so
password requisite pam_cracklib.so try_first_pass retry=3
password sufficient pam_unix.so md5 shadow nullok
password required  pam_deny.so
session optional  pam_keyinit.so revoke
session [success=1 default=ignore] pam_succeed_if.so service in
crond   quiet      use_nid
session required  pam_unix.so
```

## SLES

An actual example from a SLES system is:

```
# cat /etc/pam.d/login:
```

```
auth    required  pam_securetty.so
auth    include   common-auth
auth    required  pam_nologin.so
account include  common-account
password include common-password
session include  common-session
session required  pam_lastlog.so nowtmp
```

```
session    required      pam_resmgr.so
session    optional      pam_mail.so standard
```

As you can see, SLES uses the include option to direct PAM to use other PAM stacks, such as common-auth, for example.

***Instructor notes:***

**Purpose** — Show students an example PAM configuration file.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Next, let's look at some commonly used PAM modules.

## Common PAM modules

- Some commonly used PAM modules are:
  - pam\_unix.so: Regular UNIX authentication (passwords)
  - pam\_env.so: Set environment variables
  - pam\_cracklib.so: Check passwords for strength
  - pam\_pwdb.so: Enforce password aging rules
  - pam\_pwcheck.so: Check passwords (SLES only)
  - pam\_nologin.so: Deny login if /etc/nologin exists
  - pam\_listfile.so: Allow/deny login if user listed in file
  - pam\_securetty.so: Allow login for root only from secure ttys
  - pam\_time.so: Allow/deny login based on time of day
  - pam\_stack.so: Include another PAM config file (RHEL/FC only)
  - pam\_limits.so: Set limits on CPU and memory usage
  - pam\_console.so: Set permissions for console users
  - pam\_deny.so: Always gives an error
  - pam\_selinux.so: Sets up the default security context for the next execed shell
- Several PAM modules have additional configuration files in /etc/security

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-22. Common PAM modules

LX158.0

### Notes:

#### Introduction

Various modules exist as part of the PAM library and can be used by applications. Also, obviously, you can write your own modules, for instance if you actually decide to use biometric authentication mechanisms.

Some PAM modules require configuration files. Typically, these files are stored in /etc/security.

***Instructor notes:***

**Purpose** — List some PAM modules.

**Details** — There are far more details on user-level security and PAM in the LX41 course.

**Additional information —**

**Transition statement** — That covers the authentication phase. Now, let's look at authorization.

## Advanced authentication options

- **Network Information Service (NIS/NIS+)**: NIS provides a means to distribute password files, group files, and configuration information across a set of managed machines. Server-Client model. Must be securely deployed.
- **Secure Remote Password (SRP)**: Provides secure remote authorization of passwords using one-way hashing of login.
- **Kerberos**: Modern network authentication protocol. Provides strong authentication security using secret-key cryptography. Server-Client model.
- **Lightweight Directory Access Protocol (LDAP)**: LDAP is a powerful protocol that can be used to access information from an object-orientated database. It allows centralization of user information (including passwords, home dirs, and so on), encryption of data, all in single place on the network.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-23. Advanced authentication options

LX158.0

### Notes:

#### Introduction

The industry has required increasingly more flexible and distributed methods of user authentication. These methods have increasingly moved onto the network. Here are a few options for this more advanced way of authenticating a user:

**Network Information Service (NIS/NIS+)** – distributes password/group files across the network providing account and password synchronization. NIS+ adds network security features.

NIS is an older solution (used to be called “yellow pages”) originally developed by Sun Microsystems.

NIS+ can be difficult to set up properly and securely.

**Secure Remote Password (SRP)** – is a protocol to remotely authenticate a password. Passwords are sent hashed (not strictly encrypted – which is important for U.S. export laws). It was developed at Stanford University. It is popular and easy to use.

**Kerberos** – is a network authentication protocol. Uses cryptography to secure users. Very good option for network based authentication. Does not interact with the Name Service Switch (NSS).

Only store user authentication information, not details such as home directory or default shell.

**Lightweight Directory Access Protocol (LDAP)** – solves the problem of network authentication in a much more fundamental way. LDAP basically instantiates a object orientated database on a server that can be used to access information about an object, say a user. The database can hold passwords, home directories, preferred shells, even phone numbers or office numbers associated with the user. Can interact with the NSS. Encryption can be used.

LDAP takes a larger amount of time to set up, but is very robust and valuable once deployed.

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Principles of authorization

- Authorization in Linux based on file permissions
  - Exception: root is allowed to do everything
- Once logged in, users cannot change their identity except through a SUID program, which allows them to run a command as someone else (most often root).
- Examples of SUID programs:
  - **passwd**: Allows users to update the /etc/shadow file
  - **mount**: Allows users to mount a floppy or CD
  - **su**: Runs a shell as another user, after supplying the password
  - **sudo**: Runs a particular command as another user
  - Various games (to track high scores)
- All SUID programs should be known to the administrator and checked/updated for security problems.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-24. Principles of authorization

LX158.0

### Notes:

#### Introduction

Authorization is generally based on file permissions. These permissions tell you what files to read and write, what directories to go to, and what programs to execute. File permissions apply to all users except root.

#### SUID programs

It is impossible for users to upgrade their own security level (in other words, become root), unless the program that is being executed has a special SUID bit set. We talk about this later. Some programs that have this bit set and thus allow you to perform an action which would otherwise not be allowed are:

- *passwd*: When you change your password, the file /etc/shadow needs to be updated. For this, you need root permissions.
- *mount*: To be able to mount a floppy or CD requires access to the /dev/fd0 and /dev/cdrom devices. This is usually reserved for root.

- *su*: This stands for “switch user.” It allows you to run a shell as another user. It is most often used to start a shell as root.
- *sudo*: This was invented when people started noticing that sometimes users need to execute scripts or complicated commands as root without actually becoming root. Traditional methods would either mean giving these users the root password or setting the SUID bit on that particular command. The first is not desirable for obvious reasons, but the second can be too permissive too; The user would be able to run the command with any arguments that he would choose.

**sudo** only allows specific users to run specific commands with specific options as specific users and nothing more. The list of users and commands that they are allowed to run is in /etc/sudoers.

Make sure that you always use absolute paths to programs when creating a sudoers file, since otherwise, users might change their **\$PATH** variable and use **sudo** to start arbitrary scripts in their own **\$HOME/bin** directory.

- Various games might have their SUID bit set. This is usually needed to implement some sort of high score tracking.

## Locating SUID programs

Apart from kernel bugs, SUID programs are the only means for a hacker to gain root privileges when he or she is logged in as a regular user. This means that all SUID programs on the system should be known to the system administrator and checked/updated regularly for security problems.

The following command will list all SUID programs on your system:

```
# find / -perm +4000 -ls
30292    32 -rwsr-xr-x 1 root root 32108 Jan 26 04:21 /bin/su
26339    36 -rwsr-xr-x 1 root root 35692 Jan 29 08:23 /bin/ping
66295    20 -rwsr-xr-x 1 root audio 20404 Jan 29 09:58 /bin/eject
66739   112 -rwsr-xr-x 1 root root 114448 Feb 1 01:49 /bin/mount
...
```

***Instructor notes:***

**Purpose** — Introduces students to the principles of authorization.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Let's look at the file permissions in depth.

# File permissions

Permission	File	Directory
r	User can see contents of File.	User can list the contents of directory.
w	User can change contents of file.	User can change the contents of the directory.
x	User can execute file as a command.	User can cd to directory and can use it in PATH.
SUID	Program runs with effective user ID of owner.	No effect.
SGID	Program runs with effective group ID of Owner.	Files created in directory inherit the same group ID as the directory.
Sticky bit	No effect.	Only the owner of the file and the owner of the directory can delete files in this directory.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-25. File permissions

LX158.0

## Notes:

### Introduction

There are a number of permission bits associated with files and directories. These permissions are:

- **r (see):** User can see the contents of the file or directory.
  - File: **less file**
  - Directory: **ls**
- **w (write):** User can modify the contents of a file or create and delete files in a directory.
  - File: **vi file** (and make some adjustments)
  - Directory: **rm file**
- **x (execute):** User can execute the file or enter a directory.
  - File: **file**
  - Directory: **cd directory**

- **Switch UID (SUID):** If the file gets executed, it runs with an effective UID of the owner of the file. This permission is not supported on shell scripts. This permission has no meaning on directories.
- **Switch GID (SGID):** On an executable file, this means that when the file runs, the process runs with an effective GID of the group owner of the file. On a directory, it means that any file/directory made within the directory will have the same group ownership as the directory rather than the primary group of the user. SUID and SGID programs are hackers' favorites. When a hacker has entered your system, he or she usually leaves some SUID /SGID programs ("trojan horses") around. With these programs, he is then able to gain root access anytime he is logged on as a regular user, even without knowing the root password. It is therefore important that the system administrator knows which SUID and SGID programs are installed on the system. They can be listed with the following command:

```
# find / -perm +6000 -ls
```

- **Sticky bit:** On an executable file (thus, a program), this bit used to mean that the program should not be removed from memory after it was executed. The next time the program were to be executed, the program would start significantly quicker. With modern memory management, this usage is no longer implemented. On a directory, it means that even if the directory has global write permissions, users cannot delete a file in that directory unless they either own the file or the directory.

**Instructor notes:**

**Purpose** — Examine the effect that permissions have on files and directories.

**Details** — The effect of the basic permissions is often forgotten or misunderstood.

For example:

The read (r) permission on a file can be likened to placing a photocopier next to a stack of documents, saying in effect, “Copy me.” The write (w) permission allows users to remove the original contents, replacing it with whatever they like.

On directories, the misunderstandings are greater. Remember, the directory is just a list of file names and i-node numbers. *The write (w) permission on a directory allows users to remove (!) files from that directory, regardless of the file permissions.*

The worst bugbear of the system administrator is, however, the SUID program. These programs can provide the so-called “back doors” to Linux. Understand what all your SUID programs do and monitor the appearance of new ones.

**Additional information** — If the SGID bit is set but not the group execute bit (for instance in mode 2644), it indicates mandatory locking. This means that while the file is opened read-write, no other process can open the file, and while the file is opened read-only, no other process can open the file read-write. There is virtually no documentation on this, except for some comments in the kernel.

**Transition statement** — Changing the permissions on files and directories is done with the **chmod** command.

## Changing permissions

- Setting file permissions is done with the **chmod** command.

```
# chmod 1755 (or o+t) commandir
# ls -ld commandir
drwxr-xr-t 2 root proj1 4096 May 19 09:00 commandir/
# chmod 2755 (or g+s) myprog
# ls -l myprog
-rwxr-sr-x 1 root root 729402 May 19 09:02 myprog
# chmod 4755 (or u+s) passwd
# ls -l /etc/passwd
-rwsr-xr-x 1 root root 2721 Mar 15 10:58 /etc/passwd
```

- Changing file owner and group:

```
# chown john finance
# chgrp staff finance
# chown john.staff finance
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-26. Changing permissions

LX158.0

### Notes:

#### Introduction

File permissions are changed with the **chmod** command. There are special flags which can be used to change to the SUID, SGID, and sticky bits.

**chmod {[ugoa]{+-=}[rwx]}|[ug]{+-=}s|[0]{+-=}t} file**

The octal method can also be used:

**chmod <octal> file**

The owner of a file can be changed using the **chown** command. Only root can execute this command.

**chown user[.group] file ...**

The owner or root can change the group ownership of a file with the **chgrp** command. The owner can only change the group to another group in his group set.

**chgrp group file ...**

**Instructor notes:**

**Purpose** — Demonstrate how to set or change extended permissions on files or directories.

**Details** — Only the root user can change the owner of a file. You can change the group of a file only if you are root or if you own the file. If you own the file but are not a root user, you can change the group only to a group of which you are a member.

**Additional information —**

**Transition statement** — We will now see how files and directories get their permissions.

## Access Control Lists (1 of 2)

- Access Control Lists (ACLs) provide additional user access controls.
- More flexibility and fine-grained control to files.
- Enabled at the file system level; in /etc/fstab
- ACL for a file displayed using **getfacl** command

```
# getfacl myfile.txt
# file: myfile.txt
# owner: tux1
# group: tux1
user::rw-
group::r--
other::r--
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-27. Access Control Lists (1 of 2)

LX158.0

### Notes:

Access Control Lists extend the user based control in Linux. The Linux implementation is based on the IEEE POSIX.1e standard and are commonly used in commercial Unix systems. ACLs are enabled at the file system level and must be enabled before the file system is mounted (you can always remount after enabled, of course).

To view the current access list for a file, use the **getfacl** command.

**Instructor notes:**

**Purpose** — Explain Access Control Lists.

**Details** — ACLs are a more involved way of user control, but you gain a finer-grain control of permission and files. They are especially beneficial when avoiding having many groups for many things.

**Additional information** —

**Transition statement** — Continued look at ACLs

## Access Control Lists (2 of 2)

- ACLs for a file are set using the **setfacl** command,
- The **ls -l** command indicates ACLs marked by a “+”.

```
# setfacl -m u:tux2:rw- myfile.txt
# file: myfile.txt
# owner: tux1
# group: tux1
user::rw-
user:tux2:rw-
group::r--
mask::rw-
other::r--
# ls -l myfile.txt
-rw-rw-r--+ 1 tux1 tux1 1 Jan 1 16:59 myfile.txt
```

- The mask entry indicates the *effective rights mask*.
- The mask will take precedence if it is more restrictive than the ACL permissions.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-28. Access Control Lists (2 of 2)

LX158.0

### Notes:

Setting control lists for a file is done using the **setfacl** command. There are also masks that can be used for a base level of permissions as well as a default control list that can be set for new files.

The command **setfacl -m mask** will allow you to set the mask.

The command **setfacl -d** will allow you to set the default control list.

***Instructor notes:***

**Purpose** — Explain Access Control Lists.

**Details** —

**Additional information** — There are more advanced options for the setfacl command.

**Transition statement** — Let's look at the default permissions.

## umask

---

- Sets the default permissions on new files
- System-wide umask for all users in /etc/profile
- Individual umask in \$HOME/.bash\_profile or \$HOME/.profile
- Default value of **umask** is:
  - For root 022
  - For user 002 (if user private groups are used) or 022 (otherwise)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-29. umask

LX158.0

### Notes:

#### Introduction

The **umask** specifies what permission bits will be set on a new file when it is created. The umask is an octal number that specifies which of the permission bits will not be set. On a file, the execute permissions can never be set automatically. The root user might have a different umask than normal users. For root, the default umask is 022, and for normal users this will be 002 (when User Private Groups are used) or 022 (otherwise). For example, a umask of 022 specifies that the permissions on a new file will be 644 and on a new directory will be 755. A umask of 000 would give 666 permissions on a file and 777 on a directory. To view the current umask value, just run the **umask** command. The default umask for all users is specified in the /etc/profile file. For specific users, it could be set in the \$HOME/.bash\_profile or \$HOME/.profile file.

**Instructor notes:**

**Purpose** — Show how the file and directory permissions can be set by default.

**Details** — To understand the working of umask, we first need to know that the permissions on files are nothing more than a number of bits in the i-node. Thus, permissions like `rwxr--r-` would be written in binary as `111100100`. A umask of `022` could also be written in binary, giving us `000010010`. The last thing we have to know is that the system will always assume that the permissions on a new file will be `666`, giving us in binary `110110110`. Now we can compute the default permissions for a file.

**Additional information —**

**Transition statement** — Let's pull all this together and create a team directory.

## Example: Creating a team directory

- Create a group:

```
# groupadd penguins
```

- Add users to the group:

```
# usermod -G penguins tux1
```

or:

```
# gpasswd -a tux1 penguins
```

- Create a directory and set group permissions:

```
# mkdir /groups/penguins
# chgrp penguins /groups/penguins
# chmod 2770 /groups/penguins
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-30. Example: Creating a team directory

LX158.0

### Notes:

#### Introduction

The visual shows an example of the steps that you need to undertake to create a team directory: A directory which allows multiple people in the same group to share files.

**Instructor notes:**

**Purpose** — As an example, show how to setup a team directory.

**Details** —

**Additional information** —

**Transition statement** — That's it for the regular users. Let's look at some special security considerations for root.

## Root access

- *Dangerous*
- Root's password should be changed on an unannounced schedule by the system administrator.
- Assign different root passwords to different machines.
- Always log in as yourself, not as root.
- Remote login as root by default disabled.
- Some Linux distributions (such as Ubuntu) disable local login as root altogether, opting to force users down a more secure path of using sudo for admin commands.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-31. Root access

LX158.0

### Notes:

#### Introduction

If the root password is known by too many people, no one can be held accountable for changes in the system. The root password should be limited to the lowest number of users possible. The fewer people who know the root password, the better. However, do not make the mistake of keeping the root password as your personal secret. A good method to achieve this is to put the root password in a sealed envelope and store it in a safe somewhere. The system administrator should ensure that distinct root passwords are assigned to different machines. You might allow normal users to have the same passwords on different machines, but never do this for root. Attempts to become root through su can be investigated. Successful and unsuccessful attempts might be logged by the audit system. Most Linux systems have remote login (through **telnet**) for root disabled by default: root is only able to log in on consoles that are listed in /etc/securetty.

***Instructor notes:***

**Purpose** — Discuss root usage

**Details** —

**Additional information** —

**Transition statement** — So let's see how we handle the root account.

## su command

- Switch to another user ID:

```
$ whoami  
tux1  
$ su  
Password:  
# whoami  
root
```

- Using **su - <user>** changes to the environment of that user.
- Execute a command as another user:

```
$ su - root -c /sbin/poweroff  
Password:  
$  
Broadcast message from root (tty1):  
The system is going down for system halt NOW!
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-32. su command

LX158.0

### Notes:

#### Introduction

The **su** command runs in a subshell with the effective user ID and root privileges (if no username is specified). You are asked for root's password before you gain root permissions. To end the session, type **exit** or **Ctrl+D**, which returns you to the original shell session and privileges. For example, **su terry** gives you the privileges of Terry, but you can still be in the environment of the user issuing su. **su - terry** sets up the environment as if you had logged in as Terry.

**Instructor notes:**

**Purpose** — Discuss su.

**Details** —

**Additional information** —

**Transition statement** — **su** is an all-or-nothing solution. If you want more fine-grained control, you can also use **sudo**.

## sudo command

- Allows users to execute specific commands as another user without requiring that user's password
  - *Do not use sudo for interactive commands!*
- /etc/sudoers file lists which users are allowed to execute which commands on which host as which user.
  - *Edit this file with visudo only*
- Macros can be defined to reduce complexity.
- Syntax:

– `user host = [(newuser)] command`

```
# cat /etc/sudoers
User_Alias OPERATORS = tux1, tux2, tux3
Host_Alias WEBSERVERS = www, www-1, www-2
Cmnd_Alias PRINTCMDS = /usr/bin/printtool, /usr/bin/klpq
tux1 WEBSERVERS = (root) /sbin/service httpd restart
OPERATORS printsrv = (root) PRINTCMDS
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-33. sudo command

LX158.0

### Notes:

#### Introduction

The **sudo** command, as mentioned, allows users to execute specific commands with the authentication of another user, on specific hosts. Which combination is possible is configured in the /etc/sudoers file.

The basic syntax of this file is easy:

**user host = [(newuser)] command**

This means that *user* is allowed to execute *command* as *newuser* on *host*. If no *newuser* is specified, it is assumed that the command is executed as *root*.

What makes this complicated, but also terribly flexible, is that for all four elements, macro definitions can be added. These macros are typically written in capital letters, and there is a special *ALL* macro defined as well. See the visual for an example of this.

The /etc/sudoers file supports a large number of options as well, which govern, for instance, whether a user is allowed to add any options to the command or not. For examples of this, see the sudoers manual page.

---

Because of security and locking issues, only edit this file with the **visudo** command, not with a regular editor.

Note that the intention of **sudo** is to allow users to execute a specific command as another user, most often root, without having to supply that user's password. This also leads to a security risk if the command that is allowed can be used for something unintended. As an example, if you let a user start **vi**, through **sudo**, then that user is able to edit that particular file. But by using the **:r** and **:w** commands in **vi**, the user is also able to edit other files owned by root. And by using **!:!** in **vi**, the user is able to execute any command as root. You should therefore be really careful in configuring your /etc/sudoers file so that it cannot be used to edit arbitrary files or execute arbitrary programs.

***Instructor notes:***

**Purpose** — Discuss sudo.

**Details** —

**Additional information** —

**Transition statement** — Okay, let's last look at a few log files and other things that can tell you what's going on.

# Security logs

- /var/log/lastlog - Last successful login
  - /var/log/messages - General log file
  - /var/log/secure - Failed logins
  - /var/log/wtmp - Successful logins
  - /var/run/utmp - Currently logged in users

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-34. Security logs

LX158.0

## **Notes:**

## Introduction

There are several files that keep track of failed, successful, and current logins. These files are:

- `/var/log/lastlog`: Records the last time a user logged in. This file can be examined with the **lastlog** command.
  - `/var/log/messages`: This is the general log file. Most applications and daemons will write log information to this file.
  - `/var/log/secure`: Keeps track of the failed login attempts. Use **more /var/log/secure** to view the contents of this file.
  - `/var/log/wtmp`: All successful logins are saved in this file. This file can also be examined with the **who** command. Another tool for viewing this file is the **last** command.
  - `/var/run/utmp`: Logs the users currently logged in the system. The default output of the **who** command is the contents of this file.

***Instructor notes:***

**Purpose** — Introduce security log files.

**Details** —

**Additional information** —

**Transition statement** — Let's also look at some commands.

## Useful commands

- Who is logged in and doing what?

```
# w
09:43:46 up 17 days, 20:50, 6 users, load average: 0.00, 0.02, 0.00
USER     TTY      FROM           LOGIN@    IDLE   JCPU   PCPU WHAT
root     pts/3    sig-9-49-140-187 09:40    2:09   0.03s  0.01s ssh 10.0.0.2
root     pts/4    sig-9-49-140-187 09:40    2:13   0.02s  0.01s ssh 10.0.0.3
root     pts/5    sig-9-49-140-187 09:41   49.00s  0.02s  0.01s ssh 10.0.0.4
root     pts/6    sig-9-49-140-187 09:43    0.00s  0.01s  0.00s w
```

- Who is logged in and examine the contents of /var/log/wtmp and /var/log/utmp?

```
# who
root     pts/3      May 19 09:40 (sig-9-49-140-187.mts.ibm.com)
root     pts/2      May  8 07:43 (:1.0)
root     pts/1      May  8 10:22 (:1.0)
root     pts/4      May 19 09:40 (sig-9-49-140-187.mts.ibm.com)
root     pts/5      May 19 09:41 (sig-9-49-140-187.mts.ibm.com)
root     pts/6      May 19 09:43 (sig-9-49-140-187.mts.ibm.com)
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-35. Useful commands

LX158.0

### Notes:

#### Introduction

The visual shows example command output.

***Instructor notes:***

**Purpose** — Introduce commands.

**Details** —

**Additional information** —

**Transition statement** — There are a few more.

## Additional commands

- Show information about a user:

```
# id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
context=root:system_r:unconfined_t
```

- Show the last time a user logged in or the last time a tty was used to log in:

```
# last
root      pts/6          sig-9-49-140-187 Fri May 19 09:43 still logged in
...
reboot    system boot   2.6.9-27.ELsmp Mon May  1 12:53      (17+20:53)
root      pts/1          sig-9-65-56-32.m Mon May  1 11:40 - down (00:03)
wtmp begins Mon May  1 11:40:45 2006
```

- Show the last login time of all users:

```
# lastlog
Username        Port      From           Latest
root            pts/6      sig-9-49-140-187 Fri May 19 09:43:15 -0700 2006
bin
...
guest          pts/7      sig-9-48-37-17.m Mon Apr 24 10:52:59 -0700 2006
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-36. Additional commands

LX158.0

### Notes:

#### Introduction

The visual shows more example command output.

***Instructor notes:***

**Purpose —** More commands.

**Details —**

**Additional information —**

**Transition statement —** That's it. Checkpoint time.

## Checkpoint (1 of 2)

1. What is a User Private Group?
  - a. A group for users who need privacy.
  - b. A group which has the same name as the user; this user has this group as its primary group.
  - c. A group which is used for sharing files between the members of this group.
  - d. The "staff" group.
2. Where are the passwords of users stored?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-37. Checkpoint (1 of 2)

LX158.0

### Notes:

**Instructor notes:**

Purpose —

Details —

## **Checkpoint solutions (1 of 2)**

---

1. What is a User Private Group?
  - a. A group for users who need privacy.
  - b. A group which has the same name as the user; this user has this group as its primary group.
  - c. A group which is used for sharing files between the members of this group.
  - d. The "staff" group.

The answer is a group which has the same name as the user; this user has this group as its primary group.

2. Where are the passwords of users stored?

The answer is /etc/shadow.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —**

**Transition statement —**

## Checkpoint (2 of 2)

3. What is the difference between authentication and authorization?
4. True or false: The user root can log in anywhere, anytime.
5. True or false: PAM is the subsystem responsible for user authentication.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-38. Checkpoint (2 of 2)

LX158.0

### Notes:

**Instructor notes:**

Purpose —

Details —

## Checkpoint solutions (2 of 2)

---

3. What is the difference between authentication and authorization?

The answer is authentication is how you identify yourself to the system, and authorization specifies what you can do once logged in.

4. True or false: The user root can log in anywhere, anytime.

The answer is false.

5. True or false: PAM is the subsystem responsible for user authentication.

The answer is true.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —**

**Transition statement —**

## Exercise: User administration and security



What you will do in this exercise:

- Add and delete user accounts
- Manipulate files involved in user administration
- Perform various activities related to user-level security

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-39. Exercise: User administration and security

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Unit summary

Having completed this unit, you should be able to:

- Add, change and delete user accounts
- Add, change and delete user groups
- Manage user passwords
- Communicate with the user community
- Describe the concepts of PAM
- Define ways of controlling root access to the system
- Define the use of SUID, SGID, and sticky bit permissions bits
- Configure Access Control Lists
- Identify the data files associated with users

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 8-40. Unit summary

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

# Unit 9. Disk management, LVM, and RAID

## Estimated time

01:00

## What this unit is about

This unit covers disk management, Logical Volume Management (LVM) and Redundant Array of Independent Disks (RAID).

## What you should be able to do

After completing this unit, you should be able to:

- List the device naming scheme for IDE and SCSI hard disks
- Partition a hard disk and list the device naming for partitions
- Configure and use LVM
- Configure and use RAID

## How you will check your progress

Accountability:

- Checkpoint questions
- Machine exercises

## Unit objectives

---

After completing this unit, you should be able to:

- List the device naming scheme for IDE and SCSI hard disks
- Partition a hard disk and list the device naming for partitions
- Configure and use LVM
- Configure and use RAID

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 9-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose** — Define unit objectives.

**Details** —

**Discussion Items** —

**Additional information** —

**Transition statement** — Let's take a look at hard disk types first.



## 9.1. Disk management

### Instructor topic introduction

**What students will do** — Learn about disk management and partitioning

**How students will do it** — Lecture and exercises

**What students will learn** — Disk management

**How this will help students on their job** — Without disk management students will not be able to store any data.

# Internal hard disk type overview

- Most common device for persistent storage
- Three common types: IDE, SATA, and SCSI
- Integrated Drive Electronics (IDE) (also known as PATA)
  - Some cabling limitations
  - Device naming /dev/hda, hdb, ..., hdh
  - Version 2 is called Enhanced IDE (EIDE)
- Serial Advanced Technology Attachment (SATA)
  - Offers some advantages over ATA/EIDE:
    - Hot swapable, faster transfer speed, smaller cabling
- Small Computer System Interface (SCSI)
  - Different subtypes: fast, wide, fast wide, ultra-wide, ...
  - Max seven or 15 disks on one bus (depends on subtype)
  - Needs correct termination at both ends of bus
  - Generally more expensive
  - Device naming /dev/sda, ..., sdz, sdaa, ..., sddx

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-2. Internal hard disk type overview

LX158.0

## Notes:

### Introduction

Hard disks are the most common form of persistent storage on a typical Linux system.

Two types are most common on the Intel (and other) architectures: IDE and SCSI.

*IDE* and the newer variant, *E-IDE*, allow a maximum of two disks to be attached to one “bus” (ribbon cable). Only one of these disks can have its controller active and is then said to be *master* of the bus. The controller of the master controls the operation of the slave too.

A typical E-IDE adapter supports two buses, and there is a maximum of two E-IDE adapters per system, yielding a total of eight E-IDE devices per system. IDE device numbering is based on how the device is connected:

- The master on the first bus on the first adapter is **hda**
- The slave on the first bus on the first adapter is **hdb**
- The master on the second bus on the first adapter is **hdc**
- and so forth.

Most CD-ROM, CD-RW, and DVD players for the home market are attached as if they were IDE devices too. This is governed by the AT Attachment Packet Interface (ATAPI) standard.

## SATA

Recently, parallel ATA/EIDE interfaces have begun to be replaced by a new technology called Serial ATA (Advanced Technology Attachment). SATA offers more speed and ease of use over older ATA/EIDE as well as hot swapping easier cable management.

## SCSI devices

*SCSI* is a technology which is technically superior to IDE but generally more expensive. It has various subtypes, each of which has its own performance characteristics and physical connector size and types. Depending on the subtype, there is a maximum of 8 or 16 devices on each bus, one of which is the SCSI controller itself. This leads to a maximum of 7 or 15 disks on each bus.

However, an adapter typically supports multiple buses, and multiple SCSI adapters might be used simultaneously.

SCSI device naming is largely based on the SCSI ID: The SCSI drivers will detect and activate each and every adapter and SCSI bus in turn and subsequently activate each device on that bus starting with the lowest SCSI ID. This number can be manipulated, typically by setting a jumper combination, dip switch combination, or rotary dial. All devices are assigned a device entry in the order in which they were detected. Thus, the first drive detected becomes **sda**, the second device becomes **sdb**, and so forth<sup>1</sup>. In most cases, the first drive detected is also used as the boot device.

Obviously, it is of the utmost importance that no two SCSI devices on the same bus have the same ID number. This can be checked by looking at the output of the **dmesg** command or by entering the SCSI BIOS when the system boots. Note that the SCSI adapter always needs one ID for itself too.

Information on your SCSI devices can be obtained from the /proc/scsi directory, and with the command **scsi\_info device**.

<sup>1</sup> This causes a problem if new disks are inserted with IDs lower than the existing drives. Suppose you have one SCSI bus with two disks connected to it. The disks use ID 3 and 6, respectively, and are named sda (device with ID 3) and sdb (device with ID 6). If you were to add a disk with ID 5, then this new disk becomes sdb, and the disk with ID 6 becomes hdc. This might lead to boot problems, particularly when the disk partitions need to be fsck-ed and mounted, and their names (/dev/hdb1, for instance) are hard-coded in /etc/fstab, instead of using ext2 file system labels.

## **Instructor notes:**

**Purpose** — Discuss common hard disk standards.

**Details** — Point out that changes to the 2.6 kernel (udev, which was discussed earlier in this unit) should help avoid naming problems on SCSI devices. However, “name slippage” might still occur. This will not be a problem on the student lab system but could be on larger servers with multiple storage devices.

## **Additional information —**

**Transition statement** — Disks are incredibly complicated devices: their platters turn around with speeds up to 10,000 RPM, and the disk arms are typically less than a micron away from the magnetic surface. It is no wonder that disk drives, together with case fans, are the first components to fail in a modern computer. Let’s see how you monitor these devices.

# Monitoring hard disk health

- Self Monitoring And Reporting Technology (SMART)
  - Technology included in most modern IDE/SCSI disks
  - Reports various disk parameters (errors, temperature, various counts, ...) to OS
- smartctl: Tool for accessing SMART data
  - smartctl -a /dev/sda  
reports all /dev/sda parameters
  - smartctl -x /dev/sda initializes long self-test
  - man smartctl for more options
- smartd: Monitoring daemon
  - Monitors attributes every 30 minutes
  - Reports changes to syslog
  - Newer versions can send mail too if attributes change (use /etc/smartd.conf for configuration)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-3. Monitoring hard disk health

LX158.0

## Notes:

### Introduction

Disk drives, together with case fans, are typically the only moving parts in a system, and thus the first devices that fail. Because of this, most modern IDE and SCSI disks are equipped with Self Monitoring and Reporting Technology (SMART).

A disk equipped with SMART continuously collects data about its own performance and environment: number of power cycles, temperature, error rates, and so forth. The operating system can then collect these parameters through a standardized interface.

The **smartmontools** use this interface to collect these parameters and report them back to the user.

There are two tools available:

- **smartctl** is a tool which collects the parameters from the drive and reports them back to the user. It can also be used to initiate self-tests if the drive supports that.
  - The two most common options are **-a**, which reports all parameters, and **-x**, which initiates a long self-test. For more options, see the manual page.

- **smartd** is a daemon which monitors all parameters of all disks every 30 minutes. It then reports any changes through the system logger. Newer versions can also report changes through mail. You will need an /etc/smartd.conf file to configure this.

**Instructor notes:**

**Purpose** — Discuss SMART.

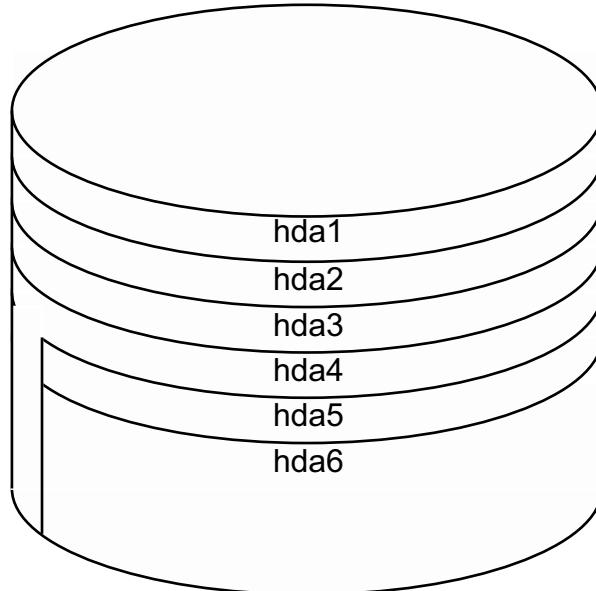
**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Disks come in various shapes and sizes, but the size is not always optimal for what we want to do with it. We therefore are able to partition these disks into smaller chunks and use them separately.

## Disk partitioning principles

- Four primary partitions, one of which may be an "extended" partition
  - `/dev/hda1` - `/dev/hda4`
- Extended partition: large number of "logical" partitions
  - `/dev/hda5` and up
- Each partition is an independent block device
  - filesystem
  - swap space
  - raw volume



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-4. Disk partitioning principles

LX158.0

### Notes:

Disk partitioning was already present in the first PCs, where hard disks did not exceed 10 Megabytes. Even though hard disks have grown considerably, the partition scheme used in the PC (Intel) world, still has not changed.

As we've seen earlier, the first sector of the disk contains the Master Boot Record, the Partition Table and a Magic Number. It is the Partition Table that concerns us in this unit.

This partition table is 64 bytes. As each partition description requires 16 bytes for start cylinder, end cylinder, type and a few other parameters, you can see that only four partitions can be described in this partition table. These four partitions are called "primary partitions" and are named `/dev/hda1` through `/dev/hda4` (assuming we're talking about hard disk `/dev/hda`).

Four partitions were not a problem in the early days of the PC. When hard disks grew however, this did become a problem. But there was no space in the first sector to increase the size of the primary partition table.

---

The solution was to designate one of the primary partitions as an "extended partition". This partition, typically /dev/hda4, was then split up in a number of "logical partitions". The administration for these logical partitions is kept in "Extended Boot Records" (EBRs), which reside in the extended partition itself.

As the EBRs form a linked list, there is no inherent limitation on the number of logical partitions that can be described this way. In practice, since each logical partition still needs to occupy at least one full cylinder, the number of cylinders on disk provides a limitation. Also, the operating system may pose a limit on the number of logical partitions that are accessible.

Each partition (primary or logical) is a separate block device and can hold different content. This may be a file system, a swap space or a raw volume to be used for a database for instance. In the rest of this unit, we will also see that these partitions can be used as Physical Volumes in an LVM Volume Group, or as RAID volumes in a RAID array.

Note that the extended partition itself cannot be used to hold data, as that will overwrite the logical volumes contained in the extended partition, plus the EBRs.

***Instructor notes:***

**Purpose** — Discuss partitioning theory

**Details** —

**Additional information** —

**Transition statement** — Let's look at some of the tools that are available for partitioning.

# Partitioning tools

- **fdisk**
  - Virtually every PC OS comes with a tool fdisk to create partitions for that OS
    - Linux, Windows, and so forth
- **parted**
  - GPLed Linux program, available at [www.gnu.org](http://www.gnu.org)
  - Can create/resize/move/delete partitions
- GParted, QTParted
  - GUI utilities that use GNU Parted
  - Can create/resize/move/delete partitions
- Disk Druid and others
  - Partitioning program integrated in Linux install program

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-5. Partitioning tools

LX158.0

## Notes:

### Introduction

A large number of tools exist for partitioning your hard disk. The most important thing to consider when choosing a tool is not whether it is able to generate a partition table (which is only 64 bytes after all), but what it can do with the content of your partitions if you decide to move or resize a partition.

***Instructor notes:***

**Purpose** — List some partitioning tools.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — OK. That is enough about hard disks themselves. Let's look at some of the things that Linux allows us to do with these disks.

## 9.2. Logical volume management (LVM)

### Instructor topic introduction

**What students will do** — Learn about LVM

**How students will do it** — Lectures, exercise

**What students will learn** — How to implement LVM and some of its features

**How this will help students on their job** — LVM makes disk management a lot more flexible.

## Logical volume management (1 of 2)

- Traditional disk partitioning scheme has several disadvantages:
  - Virtually impossible to resize or move a partition.
  - Partition size is limited by disk size.
- Logical volume management solves these disadvantages:
  - One or more physical volumes (hard disks, partitions) are assigned to a volume group (VG).
  - All physical volumes (PV) are split into physical extents (PE) of identical size (default 4 MB).
  - PEs in a VG can be combined into logical volumes (LV), which can be used like any block device.
- An LV can span multiple disks
- To increase the size of an LV, add PEs
- To increase the size of a VG, add PVs

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-6. Logical volume management (1 of 2)

LX158.0

### Notes:

Logical volume management (LVM) is a technique to overcome some limitations that are imposed on the system with the traditional partitioning scheme:

- It is virtually impossible to re-size or move a partitions since other partitions are always in the way
- The largest partition you can create is one that spans your whole disk, and thus the size of any partition is limited by your disk size

To overcome these limitations, LVM introduces some extra abstraction layers in this scheme:

1. Every hard disk or hard disk partition is assigned to a volume group (VG). Each hard disk or hard disk partition is then called a physical volume (PV).
2. Each physical volume is split into physical extents (PEs) of identical size. The default size of a PE is 4 MB, but this can be changed when the VG is defined.

3. PEs in a VG are then combined into logical volumes. Each logical volume is a block device and can be used to hold a file system, for instance. Since an LV always consists of one or more PEs, its size will always be a multiple of 4 MB.

The physical extents (PEs) that are part of an LV do not have to be on the same physical disk or disk partition, as long as they are all part of the same volume group. That means that a logical volume can be larger than your physical disk size. Furthermore, the PEs that are part of an LV do not have to be sequentially located on disk. This means that it is easy to extend an LV. If a volume group becomes full, it can be extended by adding another PV (a hard disk or hard disk partition).

***Instructor notes:***

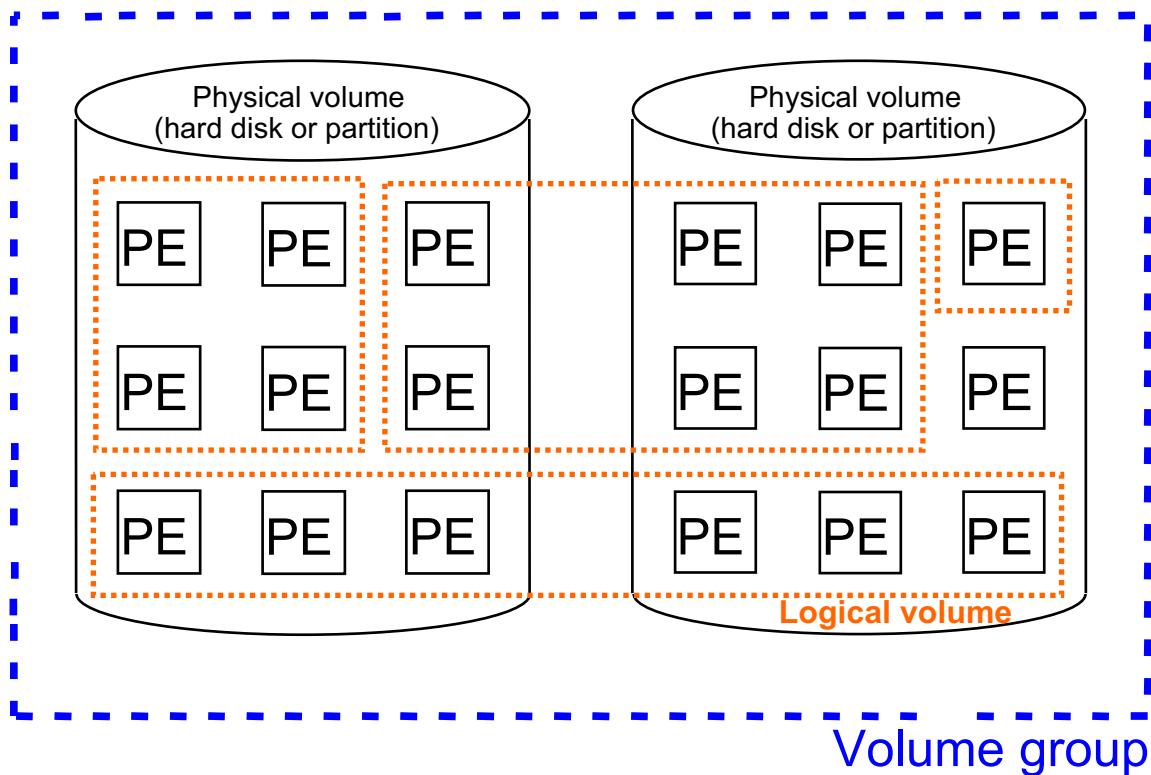
**Purpose** — Introduce LVM.

**Details** — Point out to students that LVM is automatically implemented on RHEL, while it is an option on SLES.

**Additional information** —

**Transition statement** — This might be a little complicated. Let's look at a graphical example.

## Logical volume management (2 of 2)



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-7. Logical volume management (2 of 2)

LX158.0

### Notes:

#### Introduction

The visual shows a volume group that consists of two physical volumes. In this case, whole disks are used as physical volumes, but you can use disk partitions too. Each PV is split into a number of PEs (nine in this case), which are our building blocks for building LVs. Four LVs have been created, with two spanning two PVs. One PE is still unallocated and can be used to extend an already existing LV or can be used to create a new LV.

***Instructor notes:***

**Purpose** — Give a more graphical example of LVM.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Well, that's the principle behind LVM. Now, let's see how you set it up.

## LVM implementation overview

- Add hard disks and/or create partitions (type 0x8e) on existing hard disks
- Initialize physical volumes (disks or partitions)

```
# pvcreate /dev/sda3
# pvcreate /dev/sdb
```

- Create volume group **vg00** with physical volumes

```
# vgcreate vg00 /dev/sda3 /dev/sdb
```

- Create logical volume **lv00** in volume group

```
# lvcreate -L 50M -n lv00 vg00
```

- Can now use /dev/vg00/lv00 as block device

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-8. LVM implementation overview

LX158.0

### Notes:

#### Introduction

Implementing LVM comes down to three tasks:

- First, you need to identify which physical volumes you are going to use, and format them accordingly. This is done with the **pvcreate** command.
- Second, you need to create the volume group which is going to exist of the physical volumes you created in the first step. This is done with the **vgcreate** command.
- Last, you need to create the logical volumes in the volume group. This is done with the **lvcreate** command.

After this, you can use your logical volumes, now called /dev/VGname/LVname as regular block devices.

***Instructor notes:***

**Purpose** — Give an overview on implementing LVM.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — There is of course a large number of other commands that can be used for LVM.

Let's first look at two commands that allow us to manage our physical volumes.

# Physical volume commands

- **pvccreate <pv>**
  - Initializes a physical volume by putting an (empty) volume group descriptor area at the start of the PV



- **pvmmove [-n <lv>] <source pv> [<destination pv>]**
  - Move PEs from one PV to another PV in the volume group
- **pvdisplay <pv>**
  - List information about a PV

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-9. Physical volume commands

LX158.0

## Notes:

### Introduction

When you implement a volume management strategy on your system, you will have a number of commands available to manage the disc devices. As an example, here are some commands you will use:

### Command descriptions

- **pvccreate** Initializes a physical volume. Among other things, this means that a volume group descriptor area (VGDA) is added at the start of the PV. This VGDA later contains LVM information, such as the size of the physical extents.
- **pvmmove** Allows you to move all PEs on a PV to another PV within the same volume group. This is useful if you want to take that PV out of the volume group.
- **pvdisplay** Allows you to view information about a PV.

**Instructor notes:**

**Purpose** — Discuss PV commands.

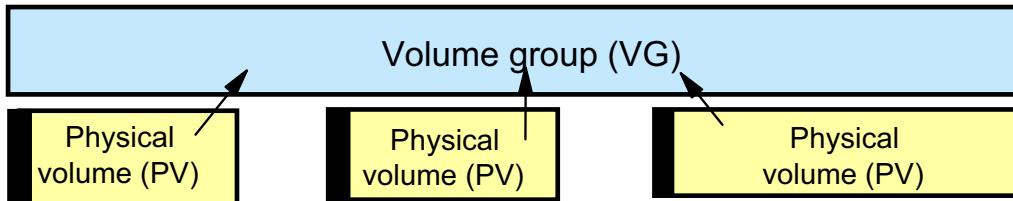
**Details** — We are only touching on a few commands in this example, so do not spend too much time on the details. Use this time as an introduction.

**Additional information** —

**Transition statement** — Let's look at VG commands next.

## Volume group commands

- `vgcreate [-s <pe size>] <vg name> <pv> [<pv>...]`
  - Create a volume group



- `vgdisplay [<vg>]`
  - Display information about a volume group
- `vgremove <vg>`
  - Delete a volume group

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-10. Volume group commands

LX158.0

### Notes:

#### Introduction

Several commands are available to let you work with volume groups. The following are a sample of some of these commands:

- **vgcreate** Allows you to create a new volume group. As part of the command, you need to specify the PE size that is going to be used in this volume group.
- **vgdisplay** Displays information about a volume group. **vgscan** can also be used to display volume group information.
- **vgchange** Changes attributes of a volume group. The most important change is to deactivate a volume group with the **vgchange -a n vg\_name** command. This needs to be done before either **vgexport** or **vgremove** can be executed.
- **vgremove** Deletes a volume group.

**Instructor notes:**

**Purpose** — Discuss VG commands.

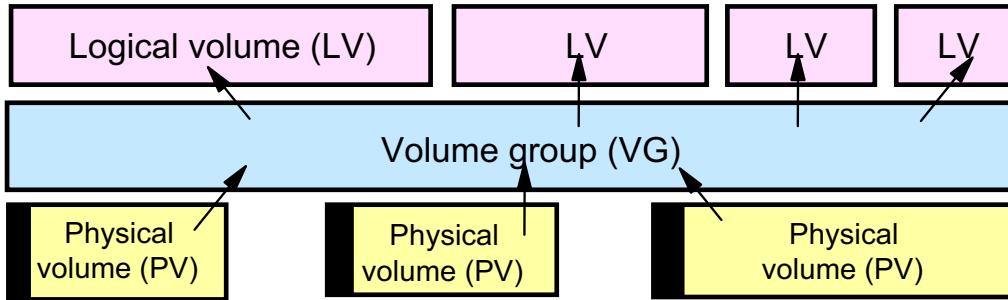
**Details** — As with the previous visual, we are simply introducing some commands dealing with the subject of volume group management. This is not intended to be an all-inclusive list. Spend this time in general discussion.

**Additional information** —

**Transition statement** — Let's also look at LV commands.

# Logical volume commands

- `lvcreate -L <size> [-n <lv name>] <vg> [<pv>...]`
  - Create a logical volume in a volume group
  - Use `-m` option to enable mirroring



- `lvdisplay <lv> [<lv>...]`
  - Display information about a logical volume
- `lvremove <lv> [<lv>...]`
  - Remove a logical volume

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-11. Logical volume commands

LX158.0

## Notes:

### Introduction

There are several commands that let you manage logical volumes. The following are some examples of these commands:

### Important logical volume commands

- **lvcreate** Creates a logical volume of the specified size, with an optional name, in a certain volume group. You can also specify the physical volumes to be used.
- **lvdisplay** Displays information about a logical volume. **lvscan** can also be used to display logical volume information.
- **lvremove** Removes a logical volume.

### Mirroring

Mirroring is done at the logical volume level. By specifying the `-m` option to `lvcreate` you can identify the number of mirrors that need to be created. Mirror copies will always be created on separate physical volumes, if possible.

**Instructor notes:**

**Purpose** — Discuss LV commands.

**Details** — This concludes our group of sample commands. Remind the students we have only touched on general concepts on these past three visuals.

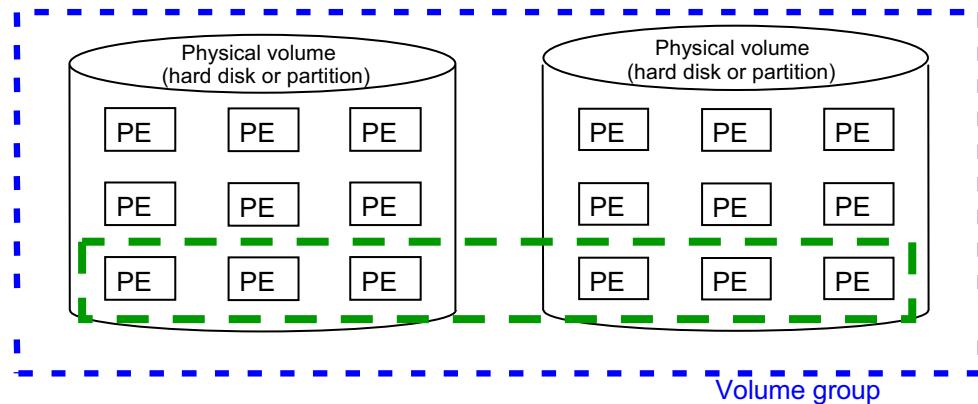
**Additional information —**

**Transition statement** — Okay, those are the basics of LVM. Now let's look at some things that you can do with LVM, but not with regular partitions. The first thing is striping.

## Striping logical volumes

- A logical volume can be striped across two or more physical volumes during creation.
- For large data transfers, this increases performance.

```
# lvcreate -L 300M -i 2 -I 8 -n mystripedlv vg00
```



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-12. Striping logical volumes

LX158.0

### Notes:

#### Introduction

Logical volumes can be striped. This means that the logical volume is spread out over several disks. This greatly increases performance for large data transfers, especially if the disks are attached to separate controllers as well. The disadvantage of striping is that if any of the disks involved fails, then your LV is lost.

**Instructor notes:**

**Purpose** — Discuss striping.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Another thing we can do is creating snapshots.

## Logical volume snapshots

- A snapshot is a frozen version of the LV.
  - Useful when live system is constantly changing, but you want to do a back up of a single point in time
- Snapshots are R/W in LVM2.
  - Allows mounting the snapshot for testing
  - Useful for creating volumes for Xen as well
- Use **lvcreate -s** to create snapshot LV.
- Snapshot LVs that become full are automatically disabled (unusable).
- Backup the snapshot!
  - Mount snapshot
    - mount /dev/mybackup /mnt/backup
  - tar can backup
    - tar -cf /dev/storage /mnt/backup

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-13. Logical volume snapshots

LX158.0

### Notes:

A "snapshot" is a frozen-in-time version of the logical volume. It allows you to access the LV in the state it was when the snapshot was taken. But at the same time the original LV is still available for normal read/write operations.

This is very useful when, for instance, you have a large database which requires near-24 hour uptime, but cannot be backed up while it is online. You stop the database, unmount the logical volume, make a snapshot of the logical volume, mount the logical volume and start the database again. This process will take less than a minute, usually. But at the end of the process you end up with two different "views" of the logical volume:

- The logical volume itself, which is opened for reading and writing, and will continue to support the database
- The snapshot, which is the frozen-in-time (unmounted) view of the logical volume at the time when the snapshot was made.

You typically mount the snapshot at a different location, normally read-only, and can now create the offline backup of your precious database at your leisure.

To understand how a snapshot works, you first need to consider how logical volumes are administered. A logical volume essentially consists of two parts:

- A Logical Volume Control Block (LVCB), which contains some generic information, and a list of pointers that point to the PEs that make up the LV.
- The PEs themselves.

When a snapshot is created, the following things happen:

- A new LVCB is created for the snapshot LV. The pointers in this LVCB point to the original PEs. So no data is copied.
- In the original LVCB, all pointers are marked so that the LVM code knows that a snapshot exists.
- A number of empty PEs are reserved and added to the "empty" list of the snapshot LVCB.

Once the original LV is mounted, and applications start writing to this LV again, the system needs to preserve the state of the original PE for the snapshot. It does this by copying the original PE contents to one of the empty PEs, once an application starts writing to this particular PE. It then updates the pointers of the snapshot LVCB, so that it now points to this "original" PE.

It is important to realize that the empty PEs that were assigned to the snapshot LV will eventually fill up. How fast this happens obviously depends on how often the original LV is written to. But because of this, snapshots are intended to be short-lived. As listed before, you can for instance use them to create a backup at your leisure. But after having created the off-line backup, you should delete the snapshot. Another usage example is in system administration: Make a snapshot of your original LVs, then upgrade the system or application, and test the upgrade. If the upgrade was successful, delete the snapshot. If the upgrade was not successful, simply copy the contents of the snapshot back into the logical volume and you are back to the pre-upgrade state.

LVM2 is the default LVM implementation in Linux today. LVM2, amongst other things, allows read/write snapshots. Obviously you need to be very careful with this, as all writes to the snapshots are lost when the snapshot is eventually deleted.

**Instructor notes:**

**Purpose** — Cover snapshots

**Details** —

**Additional information** —

**Transition statement** — Another advantage of LVM is that volume groups and logical volumes can be extended and reduced on the fly.

## Extending or reducing a volume group

- To add or remove a physical volume to or from a volume group, use the **vgextend** and **vgreduce** commands.
- To move physical extents from one physical volume to another, use **pvmove**.

```
# vgextend vg00 /dev/sdb6
# vgreduce vg00 /dev/sda5
ERROR: can't reduce volume group "vg00" by used
physical volume "/dev/sda5"
# pvmove /dev/sda5 /dev/sdb6
# vgreduce vg00 /dev/sda5
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-14. Extending or reducing a volume group

LX158.0

### Notes:

#### Introduction

After a while, you might find that the original LVM scheme that you created when you installed the system is not suitable for your needs anymore. With a traditional partitioning scheme, you need considerable downtime of your system to rearrange your partitions.

With LVM, you can add disks to the system and then, while the system is running, add these disks to volume groups. You can then migrate physical extents to this new disk and take the old disk out of the volume group. All this can take place while the system is running: You do not even have to unmount the logical volumes involved.

Extending a volume group with a new physical volume is done with **vgextend**. Moving physical extents from one physical volume to another is done with **pvmove**, and reducing the volume group is done with **vgreduce**.

**Instructor notes:**

**Purpose** — Discuss extending and reducing a volume group.

**Details** — This reviews the command discussion from a previous visual. This should reinforce these concepts to the student.

**Additional information —**

**Transition statement** — However, LVM allows an even neater thing: extending and reducing logical volumes.

## Extending or reducing a logical volume

- To extend/shrink a logical volume use the **lvextend/lvreduce** commands.
    - Use **-L** option to specify size in bytes
    - Use **-I** option to specify size in PEs
  - **lvextend/lvreduce** do not extend/shrink a file system in the LV automatically!
- (Extending/shrinking a file system will be covered later.)

```
# lvextend -L +300M /dev/vg00/mylv
lvextend -- rounding relative size up to physical extent boundary
lvextend -- extending logical volume "/dev/vg00/mylv" to 380 MB
lvextend -- doing automatic backup of volume group "system"
lvextend -- logical volume "/dev/vg00/mylv" successfully extended
# lvreduce -l -12 /dev/system/mystripedlv
...
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-15. Extending or reducing a logical volume

LX158.0

### Notes:

#### Introduction

Just like we could extend a volume group, we can also extend logical volumes. This is done simply by adding physical extents at the end of the logical volume or taking them away from the end. The commands for this are **lvextend** and **lvremove**.

**Important note:** If you extend or reduce a logical volume with **lvextend** and **lvremove**, then you are not automatically enlarging or shrinking the file system inside. This is a separate step and will be covered in the next unit.

**Instructor notes:**

**Purpose** — Discuss LV extending and reducing.

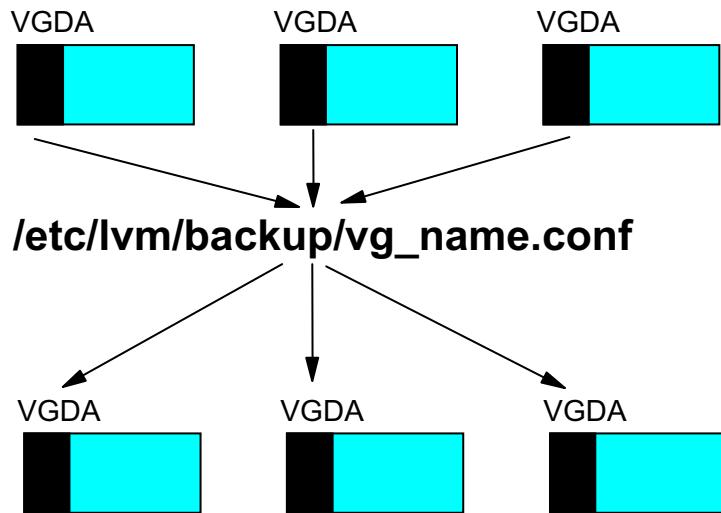
**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Okay, that is it for the neat things that LVM can do. However, this comes at a small price though: increased backup complexity.

## LVM backup and recovery

- It is very important to save the LVM metadata stored in the VGDA for recovering reasons.
  - 1. `vgcfgbackup`**
  - 2. `vgcfgrestore -n vg_name PV`**



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-16. LVM backup and recovery

LX158.0

### Notes:

#### Introduction

When backing up your system, it is vitally important to save the LVM metadata (which is stored in the VGDA) as well. (Just as it is equally important to save the partition table of a system when doing a backup.)

Creating a text file that contains the LVM metadata is done with the **vgcfgbackup** command. The resulting text file, `/etc/lvm/backup/vg_name.conf` can be archived just like any file.

In the event you need to restore your system, you can create your LVM configuration with the **vgcfgrestore** command. Obviously, you also need to restore the data stored in your logical volumes as well in that case, but that is another topic.

***Instructor notes:***

**Purpose** — Discuss backing up the VGDA.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Okay, you are nearly done with LVM. Just some loose ends.

## Additional LVM considerations

- Do not put /boot on LVM
- Advanced LVM features:
  - Splitting volume groups
  - Migrating volume groups to other system
  - Extending PVs (useful on LUNs)
- CLVM (Cluster LVM) allows for concurrent access
  - Useful in clusters with data on SAN

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-17. Additional LVM considerations

LX158.0

### Notes:

#### Boot Loader issues

Most bootloaders, including GRUB, do not support LVM. For this reason, GRUB itself (stage 1.5, stage 2, grub.conf and various related files) and the kernel files (kernel image and initramfs) that need to be loaded by GRUB, need to be kept outside of LVM. In practice, this means that you need to create a small primary partition (64-100 MB is usually sufficient) that will be mounted on /boot. All distributions are setup so that everything needed to boot a system is stored in /boot, so this neatly ensures that the system can boot despite LVM.

#### Advanced features

LVM has a few advanced features that are outside the scope of this course. We mention them here so you know they're possible, but you will have to consult the respective manual pages for their exact implementation.

- LVM allows you to split volume groups. This is done by first migrating the LVs to the proper disks, and then split a number of disks off into a new volume group.

- If you want to migrate volume groups to a different system, you need to deactivate and export the volume group, using the **vgchange -a n** and **exportvg** commands, respectively. On the new system, you first import and then activate the volume group using **importvg** and **vgchange -a y**.
- LVM also has the ability to work with physical volumes that were extended. Obviously extending physical volumes that reside on a local hard disk is very complicated, as you need to rearrange your complete partition table to do so. But this feature is particularly useful if you use a Storage Area Network (SAN) which offers Logical Units (LUNs) to your Linux partition. The SAN storage solution that you're using might be able to resize LUNs on the fly, and by using the **pvresize** command you can have Linux detect the new size, and add PEs to the volume group accordingly.

Last, LVM and LVM2 are not capable of concurrent access. If you are in a cluster situation, where multiple Linux machines have access to the same LUNs, you need something called "Cluster Logical Volume Management" (CLVM) to support LVM on these LUNs. You will also need a file system that can deal with concurrent access, such as GFS or IBMs GPFS.

**Instructor notes:**

**Purpose** — Wrap up the LVM part of the lecture.

**Details** — Discuss student notes.

**Additional information** — Mirroring means that one or more copies of each PE are made automatically and continuously on another PV which is also part of the VG. If a disk fails, then at least you still have your data.

**Transition statement** — Now let's move our discussion to RAID.

## 9.3. RAID

### Instructor topic introduction

**What students will do** — Learn about RAID

**How students will do it** — Lecture and exercises

**What students will learn** — Principles and configuration of RAID

**How this will help students on their job** — RAID may be very useful to provide mirroring capabilities to Linux.

# RAID

- Redundant Array of Independent Disks
- Typical PC hard disks, compared to expensive mainframe-quality hard disks, are:
  - Slower
  - Less reliable
  - Smaller
  - But less expensive
- RAID uses multiple hard disks in an array to create a logical device that is:
  - Faster
  - More reliable
  - Or larger
  - And still relatively inexpensive

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-18. RAID

LX158.0

## Notes:

### Introduction

*Redundant Array of Independent Disks (RAID)* was developed separate from LVM as a technique to increase the performance and reliability of hard disks by dividing or replicating data among multiple hard disks. This was done because people had observed that typical PC hard disks, especially in the early days of the PC, were slower, less reliable, and smaller than the devices used on mainframes but were also less expensive. Thus, people started packing a large number of these hard drives together with some additional control software (usually implemented on a dedicated hardware chip). Once combined, they could be used as if it were one logical device that was either faster, more reliable, or larger than the individual disks.

This new combination was a cost effective way increase the reliability, speed and size of local storage disks. It is important to note that the three features (speed, reliability, and size) are, to a certain extent, mutually exclusive. It is possible to create a RAID array that is both faster, more reliable, and larger than a single disk, but this requires a lot of hardware.

Usually, RAID arrays are only used to boost either speed, reliability or size, but not all simultaneously.

***Instructor notes:***

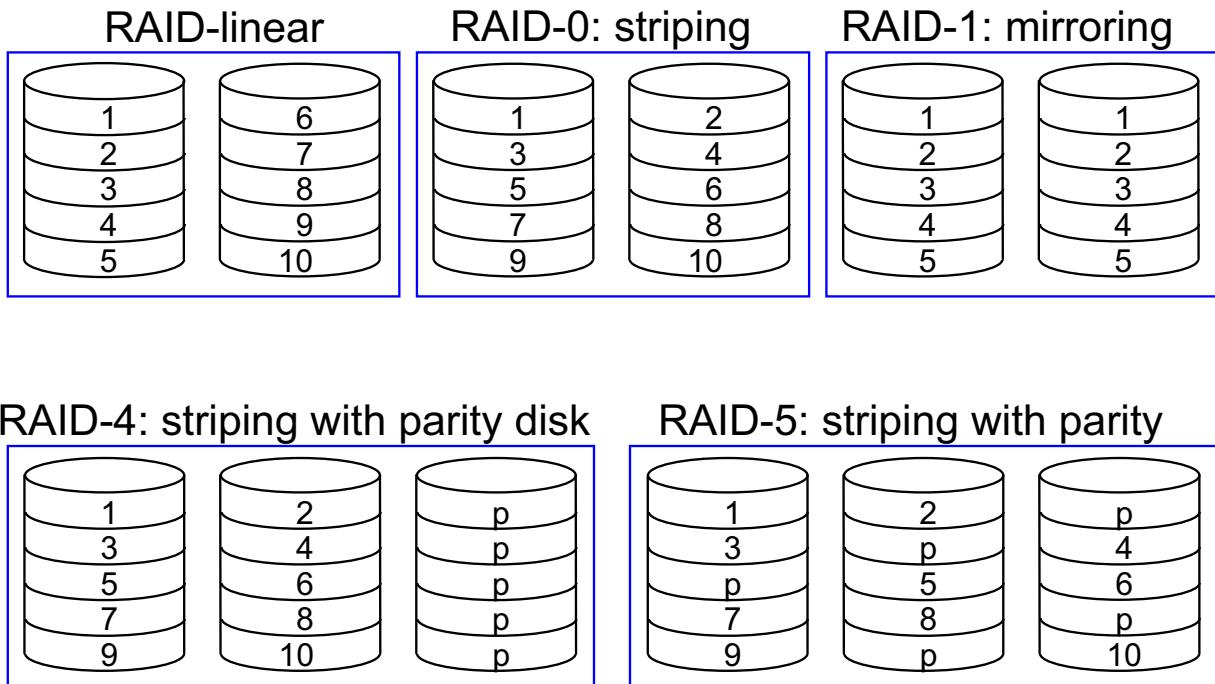
**Purpose** — Introduce RAID.

**Details** — Discuss student notes.

**Additional information** — This technique was originally called “Redundant Array of Inexpensive Disks.” “Inexpensive” is now replaced with “Independent,” but the former term is the one that was used when the term “RAID” was first coined by the researchers at the University of California at Berkeley, who first investigated the use of multiple-drive arrays in 1987.

**Transition statement** — Several RAID levels have been defined. Let's look at the most common ones.

## RAID levels (1 of 2)



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-19. RAID levels (1 of 2)

LX158.0

### Notes:

#### Introduction

In the RAID standards, several different “levels” have been defined. All these levels have different ways of storing the data on disk and thus exhibits different characteristics.

#### RAID linear

The first method, *RAID-linear* is actually not listed in the RAID standard. It is implemented in Linux as a way of simply combining two or more partitions on different disks into one, larger block device. The first partition is written until it is full, and then the second disk is used.

RAID level zero, or *RAID-0* for short, is nearly the same as RAID-Linear. With RAID-0 however, data is striped across the different disks. This means that reading or writing a large file actually puts both disks to work, which theoretically leads to a doubled throughput (that is, if your controller, bus, memory, and CPU can sustain that). If one disk is larger than the other, then the last part of the data can not be striped but just stored on the larger disk.

It would seem that RAID-0 is always preferable over RAID-linear, but in reality, it is not. Consider for instance the situation where one of your disks crashes. With RAID-linear, there is a good chance that you can retrieve at least half of your files. With RAID-0, every single file (except for the really small ones) was stored at least partly on the disk that had crashed. You should therefore use RAID-0 only for data which can be missed or easily restored.<sup>4</sup>

## RAID 1

*RAID-1* uses the second (and third disk) for mirroring: data written to the first disk is written to all other disks as well. This costs a lot of disk space, but means that you can sustain multiple disk crashes without losing your data.

## RAID 4

*RAID-4* also offers redundancy, not by mirroring, but by storing parity information<sup>1</sup> on a separate disk. Should one disk (or the parity disk) fail, then the data on this disk can be calculated from the data on the other disks. RAID-4 therefore needs at least three disks. RAID-4 uses striping to store the data blocks on disk for increased performance.

## RAID 5

*RAID-5* is similar to RAID-4 in that it calculates the parity of two disk blocks and stores this in a third disk block. It also stripes the data onto the disks. The difference between RAID-4 and RAID-5 is that RAID-4 stores all parity information on the same disk. This disk then quickly becomes a bottleneck unless this disk is significantly faster than the others. With RAID-5, the parity information is striped too, leading to better performance.

## RAID 6 (not shown in the picture)

*RAID-6* is similar to RAID-5, but uses two parity blocks instead of one. This means that RAID-6 is able to sustain a two-disk failure.

## RAID 10 (not shown in the picture)

*RAID-10* is a combination of RAID-0 (striping) and RAID-1 (mirroring).

Several other RAID levels exist, but these are not implemented in Linux, and not widely used anyway.

<sup>1</sup> The parity in this case is calculated by XORing the data on disk 1 with the data on disk 2. If one of the three elements (disk1, disk2, parity) should fail, then that element can be calculated based on the other two. This is because the parity calculation can easily be reversed. In other words,  $p = d1 \text{ XOR } d2$  also means that  $d1 = p \text{ XOR } d2$ , and  $d2 = p \text{ XOR } d1$ . This can be extended to as many disks as needed, although five or six disks is normally the limit.

**Instructor notes:**

**Purpose** — Discuss RAID levels.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — You have already seen that each level has its own characteristics. Now, let's compare them to each other.

## RAID levels (2 of 2)

- RAID levels have different characteristics.
  - RAID-5 is not better than RAID-1
- Use RAID level according to needs.

RAID level	Min # disks	Read performance	Write performance	Redundancy	Data capacity with 3x1GB disk	Other remarks
Linear	2	Equal	Equal	No	3 GB	Can be used if disks are not equal
0	2	Fast	Fast	No	3 GB	
1	2	Fast	Somewhat slower	Yes	1 GB	Can sustain N-1 disk crash(s)
4	3	Somewhat faster	Slow	Yes	2 GB	Can sustain one disk crash Parity disk is bottleneck
5	3	Somewhat faster	Somewhat faster	Yes	2 GB	Can sustain one disk crash CPU intensive

(\*) Performance compared to a single disk, for data transfers greater than block size

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-20. RAID levels (2 of 2)

LX158.0

### Notes:

#### Introduction

As seen in the visual, the different RAID levels use different ways of storing the data on disk. This leads to different characteristics. What you should note is that RAID-5 is not better than RAID-1. It is just different and might or might not be suited for your circumstances.

**Instructor notes:**

**Purpose** — List the differences between the most common RAID levels.

**Details** — Discuss student notes.

**Additional information** — There is a lot of information about RAID on

<http://www.storagereview.com/guide2000/ref/hdd/perf/raid/index.html>.

**Transition statement** — Let's see what Linux does with this concept of RAID.

# Linux RAID support

- Software RAID
  - Implemented in Linux kernel
  - Needs **mdadm** package
  - Uses disk partitions to create RAID devices
  - Logical device name: /dev/mdn
- Hardware RAID
  - Implemented in special adapter cards
  - Adapter needs to be supported by Linux kernel
  - Generally specific software needed to configure adapter correctly (might not be available under Linux)
  - RAID devices show up as regular SCSI disk

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-21. Linux RAID support

LX158.0

## Notes:

### Introduction

Linux supports both software RAID and hardware RAID.

Software RAID means that all the RAID logic is built into the Linux kernel. The user can access the partitions directly or go through the RAID layer and access the RAID volumes, which are called /dev/mdn. To implement this, you need the **mdadm** package, which is usually supplied as part of your distribution. For software RAID, the only thing you need is more than one (IDE and/or SCSI) hard disk. In fact, you can even test it by using multiple partitions on one single disk, but that negates any benefit you might want to gain from RAID.

Hardware RAID is typically implemented in special adapter cards, which look like SCSI controllers (in fact, they usually are) but contain some special RAID chipsets. Most of these controllers are supported by Linux. In fact, Linux just detects a single large disk instead of multiple, smaller ones. Configuring these adapter cards might require special software, but once the cards are configured, no additional software is needed.

**Instructor notes:**

**Purpose** — Give an overview of Linux RAID support.

**Details** — Discuss student notes.

**Additional information** — I have even seen hardware RAID adapters that make use of IDE disks, but still show up as a single SCSI disk in Linux.

**Transition statement** — We are only going to look at software RAID.

## Linux software RAID implementation: mdadm

- Linux multiple devices (MD) admin tool
- Handles all configuration, monitoring, and maintenance tasks
- Common syntax for every RAID management command
- No configuration file

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-22. Linux software RAID implementation: mdadm

LX158.0

### Notes:

#### Introduction

RAID devices are virtual devices created from two or more real block devices. This allows multiple devices (typically disk drives or partitions thereof) to be combined into a single device to hold (for example) a single file system. Some RAID levels include redundancy and so can survive some degree of device failure.

Linux software RAID devices are implemented through the multiple devices (MDs) device driver. The classic **raidtools** are the standard software RAID management tool for Linux, so using **mdadm** is not a must.

Multiple devices admin (**mdadm**) is an extremely useful tool for running RAID systems. It can be used as a replacement for the **raidtools**, or as a supplement.

The main differences between **mdadm** and **raidtools** are:

- **mdadm** can diagnose, monitor and gather detailed information about your arrays
- **mdadm** is a single centralized program and not a collection of disperse programs, so there is a common syntax for every RAID management command **mdadm** can perform

---

almost all of its functions without having a configuration file and does not use one by default. Also, if a configuration file is needed, **mdadm** helps with management of its contents.

**Instructor notes:**

**Purpose** — Introduce **mdadm** tool.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Next, let's look at the operations of **mdadm**.

## mdadm modes

- **mdadm** major modes of operation:
  - Assemble
  - Build
  - Create
  - Manage
  - Misc
  - Follow or Monitor

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-23. mdadm modes

LX158.0

### Notes:

#### Modes of operation

When using **mdadm**, you will be operating in one of the following modes:

- **Assemble:** Assemble the parts of a previously created array into an active array. Components can be explicitly given or can be searched for. **mdadm** checks that the components do form a bona fide array, and can, on request, fiddle superblock information so as to assemble a faulty array.
- **Build:** Build a legacy array without per-device superblocks.
- **Create:** Create a new array with per-device superblocks.
- **Manage:** This is for doing things to specific components of an array, such as adding new spares and removing faulty devices.
- **Misc:** This mode allows operations on independent devices such as examining MD superblocks, erasing old superblocks, and stopping active arrays.
- **Follow or Monitor:** Monitor one or more md devices and act on any state changes.

This is only meaningful for RAID 1, 4, 5, 6 or multipath arrays as only these have interesting state. RAID 0 or linear never have missing, spare, or failed drives, so there is nothing to monitor.

***Instructor notes:***

**Purpose** — Introduce **mdadm** modes of operation.

**Details** — Discuss the different modes of operation.

**Additional information** —

**Transition statement** — Let's look at how to implement **mdadm** on your system.

## mdadm implementation

- Create

```
# mdadm --create -v /dev/md0 -l \
linear -n 2 /dev/sda6 /dev/sda7
```

- Manage

```
# mdadm --detail /dev/md0
```

- Misc

```
# cat /proc/mdstat
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-24. mdadm implementation

LX158.0

### Notes:

#### Introduction

The following identifies command arguments to use within the various modes of **mdadm**.

#### mdadm - create (build) mode

- **-l, --level** - Set RAID level. When used with --create, options are: linear, raid0, 0, stripe, RAID1, 1, mirror, RAID4, 4, RAID5, 5, RAID6, 6, multipath, mp.
- **-n, --raid-devices** - Specify the number of active devices in the array. This, plus the number of spare devices (see below) must equal the number of component devices (including “missing” devices) that are listed on the command line.

Setting a value of 1 is probably a mistake and so requires that force be specified first. A value of 1 will then be allowed for linear, multipath, raid0 and raid1. It is never allowed for RAID4 or RAID5.

- **-x, --spare-devices** - Specify the number of spare (eXtra) devices in the initial array. Spares can also be added and removed later. The number of component devices listed

on the command line must equal the number of RAID devices plus the number of spare devices.

### mdadm - manage mode

- **-a, --add** - Hotadd listed devices.
- **-r, --remove** - Remove listed devices. They must not be active.
- **-f, --fail, --set-faulty** - Mark listed devices as faulty.

### mdadm - misc mode

- **-R, --run** - Start a partially built array.
- **-S, --stop** - Deactivate array, releasing all resources.



#### Note

Upon initial RAID creation, an “initial resync” activity will take place, even on empty drives. This can take some time and the array is not fully resilient in the meantime. It is, however, fully usable. Once the resync is complete, the array is clean and all disks should be active.

***Instructor notes:***

**Purpose** — Look at implementation of the **mdadm** command.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — To view the operation of a RAID device, we can also look to the /proc file system for information.

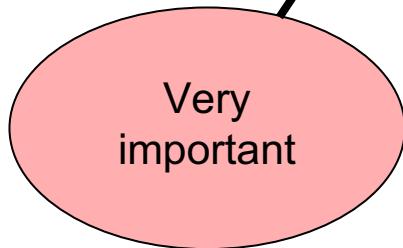
## Watching a running RAID

- To see the current state of your RAID view /proc/mdstat

```
# cat /proc/mdstat
Personalities: [linear]      [raid5]
read_ahead   1024 sectors
md0:  active raid5 sda1[2] sdb1[1] sdd1[3] sde1[0]
633849 blocks level 5, 32k chunk, algorithm 2 [4/4]  [UUUU]
unused devices: <none>
```

–U means all devices are up.

–F means a device has failed.



Very  
important

- For better understanding, use: `watch -n1 -d cat /proc/mdstat`

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-25. Watching a running RAID

LX158.0

### Notes:

#### Introduction

To check the health status of your RAID subsystem, view the contents of the /proc/mdstat file. This contains all the information you need.

Particularly important are the letters between the square brackets. These signal the health status of each of the disk or disk partitions that make up a RAID volume. A U means that the device is up and running normally, but an F means that the device has failed. You should investigate this and possibly replace the device as soon as possible.

#### Operation

If you want to simulate a failing RAID device, you can use the command **raidsetfaulty**. Do not forget, you can open a new terminal and run the following command:

```
# watch cat /proc/mdstat
```

You can also simulate a failed disk using **mdadm**:

```
# mdadm --manage --set-faulty /dev/md0 /dev/sda7
# mdadm /dev/md0 --remove /dev/sda7
# mdadm /dev/md0 --add /dev/sda7
# mdadm --stop /dev/md0 # mdadm --run /dev/md0
```

**Instructor notes:**

**Purpose** — Discuss RAID status determination.

**Details** — Discuss student notes.

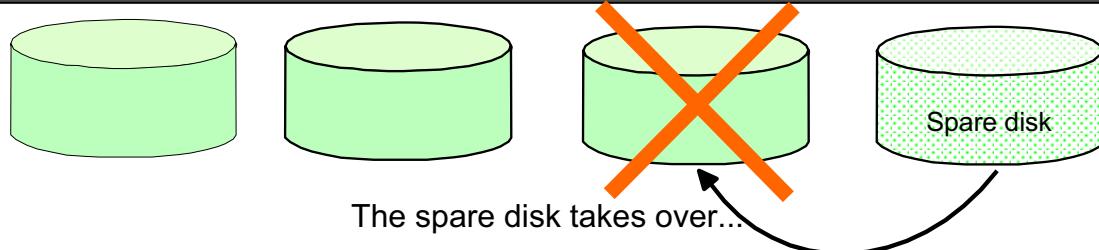
**Additional information —**

**Transition statement** — One thing you can do to prevent failed disks from having dire consequences is to add spare disks.

## Spare disks

- To make RAID1/RAID5 more failsafe in case of a disk failure, use spare disks!

```
# cat /etc/raidtab
...
nr-spare-disks 1
device /dev/sdd1
spare-disk 0
...
```



- Remove a failed disk with **raidhotremove**
- Add a new disk to the array with **raidhotadd**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-26. Spare disks

LX158.0

### Notes:

#### Introduction

To make a RAID-1 or RAID-5 configuration more failsafe, you can add spare disks. These disks sit idle until one of the active disks fails. The spare disk is then used in place of this active disk.

For a RAID-1 volume, this means that the spare disk is now being mirrored with the other disks. For a RAID-5 volume, this means that the parity information, together with the data remaining on the other disks, is used to re-create the data that is now lost.

#### Benefits of spare disks

You might wonder why a RAID-1 configuration can benefit from spare disks, while this disk can also be configured to perform continuous mirroring anyway. The reason is simply performance: keeping disks mirrored costs time. It is faster to have two mirrored disks and one spare than to have three mirrored disks.

## Array management

If a disk in a RAID array fails, you want to replace it. This can be done without taking the RAID array down. First, you execute the **raidhotremove** command to remove the failed disk. You then swap disks and execute the **raidhotadd** command to add the new disks.

Note that the new disk takes the place of the spare disk (if you configured spare disks), and the former spare disk remains production disk.

***Instructor notes:***

**Purpose** — Discuss spare disks.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Some last RAID considerations.

## Additional RAID considerations

- Put RAID partitions on different disks.
- Use different SCSI or IDE controllers if possible for different disks that are part of a RAID volume.
- Do not use RAID for /boot partition.
- If RAID-based file systems are listed in /etc/fstab, then RAID support needs to be included in the initrd.
- Software RAID4 and RAID5 need a lot of CPU time.
- Do not use RAID-linear or RAID0 for swap space.
  - The Linux kernel can stripe across swap spaces more efficiently.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-27. Additional RAID considerations

LX158.0

### Notes:

#### Introduction

There are a few things to note when using RAID:

Always put your RAID partitions on different disks, or you can nullify any advantage that RAID might try to give you. If possible, use different SCSI and/or IDE controllers for the different disks (or partitions) that make up your RAID volume. This increases your performance and reliability.

#### Boot partition

With RHEL and Fedora, never use RAID for your /boot partition, and note that if you use RAID for any other file systems listed in /etc/fstab (that are mounted automatically), you need to include the RAID modules (“linear”, “raid0”, “raid1”, “raid5” and/or “xor”) in your initial root disk (initrd).

## Performance considerations

Software RAID4 and RAID5 need a lot of CPU time to perform the parity calculations.

For maximum reliability, RAID4 and RAID5 allow you to configure spare disks. These disks (usually only one per array) are not used until one of the other disks in the array fails. If that happens the RAID software automatically starts using the spare disk instead of the disk that failed. The data on that disk is created automatically from the parity information on the other disks.

Do not use RAID-linear or RAID0 for swap space. The kernel itself can stripe swap data over multiple swap spaces if multiple swap spaces are defined and can do this faster than the RAID subsystem. On the other hand, using RAID1, RAID4, or RAID5 can be used to increase the reliability of your swap subsystem.

***Instructor notes:***

**Purpose** — Discuss some last RAID considerations.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Done. Checkpoint time!

## Checkpoint

---

1. True or false: RAID volumes can be used as physical volumes in an LVM setup.
  
2. Mirroring is offered by RAID level:
  - a. Linear
  - b. Zero
  - c. One
  - d. Four
  - e. Five

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 9-28. Checkpoint

LX158.0

### Notes:

**Instructor notes:**

**Purpose** — Check student progress.

**Details** —

## Checkpoint solutions

1. True or false: RAID volumes can be used as physical volumes in an LVM setup.

The answer is true.

2. Mirroring is offered by RAID level:

- a. Linear
- b. Zero
- c. One
- d. Four
- e. Five

The answer is one.

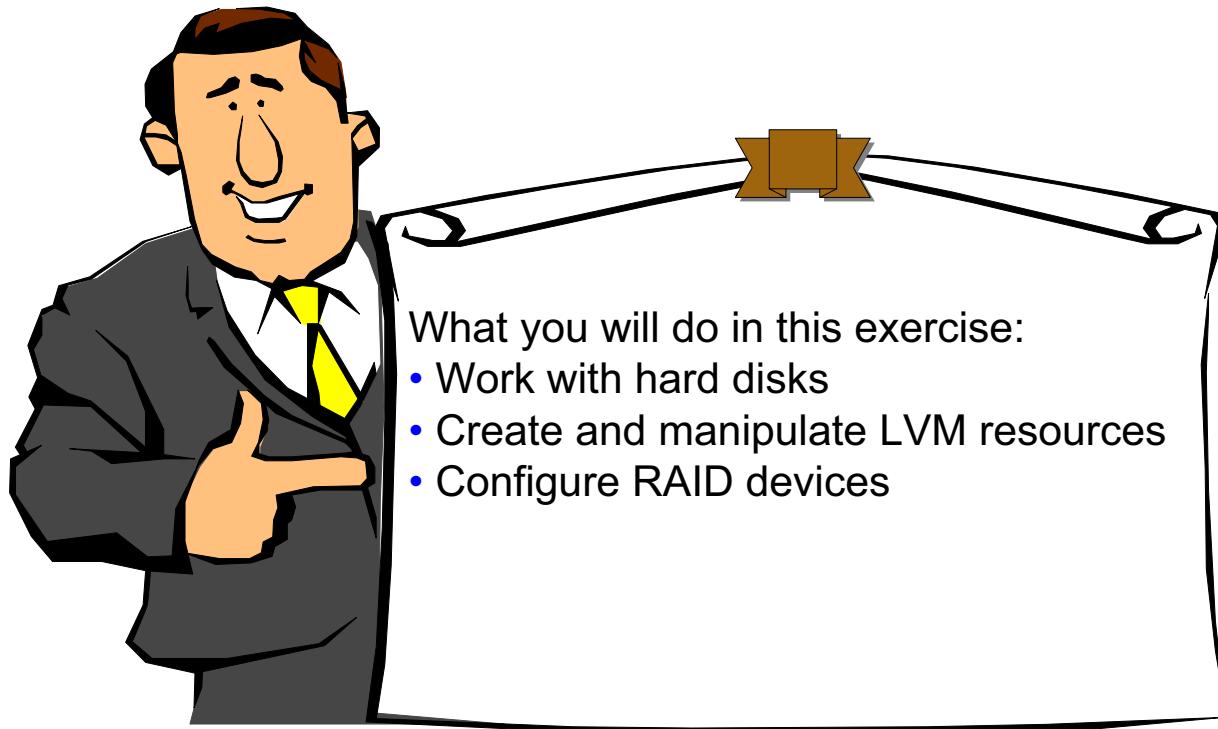
© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information** —

**Transition statement** — Next, we will proceed with the exercise.

## Exercise: Disk management, LVM, and RAID

---



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 9-29. Exercise: Disk management, LVM, and RAID

LX158.0

### Notes:

***Instructor notes:***

**Purpose** — Introduce the exercise.

**Details** —

**Additional information** —

**Transition statement** — Let's review what we have discussed.

## Unit summary

---

Having completed this unit, you should be able to:

- List the device naming scheme for IDE and SCSI hard disks
- Partition a hard disk and list the device naming for partitions
- Configure and use LVM
- Configure and use RAID

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 9-30. Unit summary

LX158.0

### Notes:

Having completed this unit, you should understand:

- Hard disks are either represented as /dev/hdx or /dev/sdx in Linux and are monitored using SMART tools
- Hard disks are normally partitioned
- Logical Volume Management allows you to go beyond the limits of regular partitioning since it allows you to create logical volumes that are larger than the disk size and which can be resized
- RAID is a technology to use inexpensive, less reliable, relatively slow and small IDE or SCSI disks in such a fashion that the virtual volume is larger, more reliable, or faster than the individual disks

***Instructor notes:***

**Purpose** — Summarize unit material.

**Details** —

**Additional information** —

**Transition statement** — Now, on to the next unit.



# Unit 10. File systems and file system quota

## Estimated time

00:45

## What this unit is about

This unit teaches you what file systems are and how to handle them.

## What you should be able to do

After completing this unit, you should be able to:

- Describe what a file is
- Describe what a file system is
- List possible file systems
- Describe i-nodes
- Create/mount/unmount file systems
- Create predefined mounts
- Set up user and group quota

## How you will check your progress

Accountability:

- Checkpoint questions
- Machine exercises

## Unit objectives

---

After completing this unit, you should be able to:

- Describe what a file is
- Describe what a file system is
- List possible file systems
- Describe i-nodes
- Create/mount/unmount file systems
- Create predefined mounts
- Set up user and group quota

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 10-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose** — Define unit objectives.

**Details** —

**Additional information** —

**Transition statement** — Let's begin with a simple question: what is a file?



## 10.1.File systems

### Instructor topic introduction

**What students will do** — Learn what a file system is, how it is built up, and how to manage it.

**How students will do it** — Lecture and Exercises

**What students will learn** — What a file system is, how it is built up, and how to manage it.

**How this will help students on their job** — Knowing how file systems are stored in a file system will help them to optimize their file systems.

## What is a file?

- Consecutive number of bytes
  - No internal structure by default (applications define their own structure)
- Stored and referenced in a file system
  - Can have multiple references (names)
- Special files exist
  - Block, Character > Device
  - Pipes, Sockets > Interprocess communication

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-2. What is a file?

LX158.0

### Notes:

#### Introduction

Linux takes its concept of a file from UNIX. In UNIX, a file is a consecutive number of bytes with no internal structure. Applications have to define their own internal structure (for instance, records). These files are stored and referenced in a file system. One file can have multiple references (file names).

#### Special files

The two standard file types you will deal with under Linux are data and special files. Special files perform a variety of roles:

##### Devices

In Linux, devices can be accessed through special files. A device special file does not use any space on the file system. It is only an access point to the device driver.

Two types of special files exist: character and block special files. The former allows I/O operations in character mode while the latter requires data to be written in block mode through the buffer cache functions. When an I/O request is made on a special file, it is

forwarded to a (pseudo) device driver. A special file is referenced by a major number, which identifies the device type, and a minor number, which identifies the unit.

### Interprocess communication

A pipe is a mechanism for interprocess communication; data written to the pipe by one process can be read by another process. The data is handled in a first-in, first-out (FIFO) order. The pipe has no name; it is created for one use and both ends must be inherited from the single process which created the pipe. A FIFO special file is similar to a pipe, but instead of being an anonymous, temporary connection, a FIFO has a name or names like any other file. Processes open the FIFO by name in order to communicate through it.

## Instructor notes:

**Purpose** — Define what a file is.

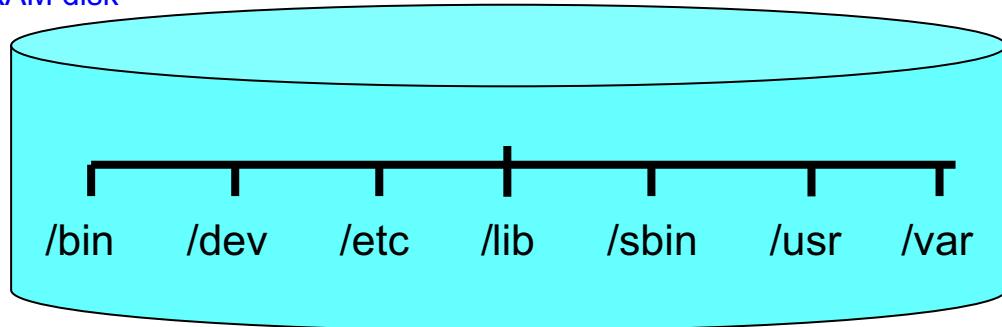
**Details** — This visual, along with the next, is meant to start the discussion at a logical point, the basic element of a file system is a file. While we will discuss some of the structures in this unit, there are many unanswered questions in this unit we will not cover. This discussion is intended for an administrative, or basic support level person, *not* a designer. The advantages or disadvantages of the various file systems is *not* the objective of this unit.

### Additional information —

**Transition statement** — A file is stored in a file system. Let's look at them.

## What is a file system?

- Place to store files and refer to them
- Hierarchical structure through use of directories
- A file system can be stored on any block device
  - Floppy disk
  - Hard disk
  - Partition
  - RAID, LVM volume
  - File (for use with a loop device)
  - RAM disk



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-3. What is a file system?

LX158.0

### Notes:

#### Introduction

The references to a file (the file names) are usually stored in a hierarchical system of directories, subdirectories and so on.

By using a mechanism called the *virtual file system*, the internals of each file system are hidden from the user.

A file system is mounted on a mount point, which is an empty directory in another (already mounted) file system. The root file system is activated at system startup, and contains the mount points for all other file systems.

A file system can be stored in any block device.

**Instructor notes:**

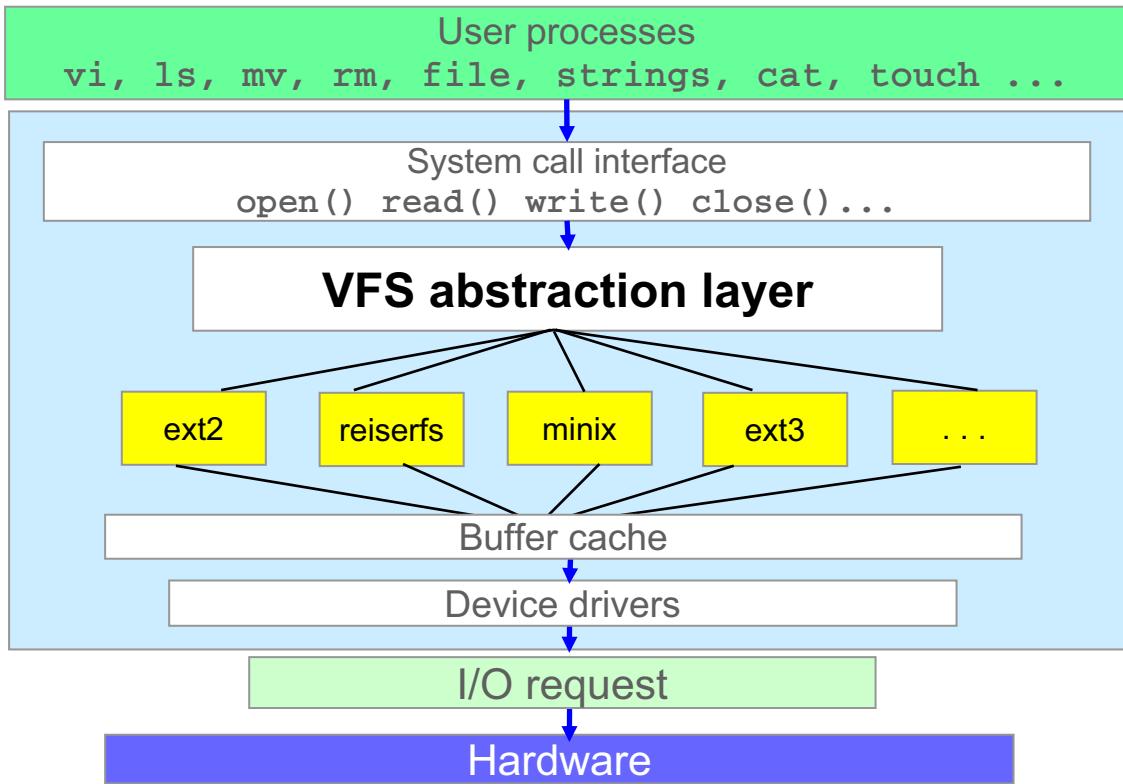
**Purpose** — Define what a file system is.

**Details** — Keep the discussion basic. We are just building up our base at this point.

**Additional information —**

**Transition statement** — Linux supports loads of different file systems. For that to work, we need something called the VFS.

# The virtual file system



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-4. The virtual file system

LX158.0

## Notes:

### Introduction

Linux supports a large number of file systems. All these file systems have their own way of storing data and metadata on disk. A typical user is not interested in the internal workings of these file systems, but needs a single way of dealing with them all. That is the job of the virtual file system switch (VFS) abstraction layer.

### Process flow

Typical user programs such as **vi**, **ls** and others use four primitive system calls to work with files: **open()**, **close()**, **read()** and **write()** (There are a few others for working with directories, permissions and so forth.) These system calls are translated into the appropriate, file system-specific system calls by the VFS layer. Because of this, a user is presented with a uniform interface to each file system.

Note that the VFS layer emulates certain features if the file system does not support these. For instance, Linux is able to mount a VFAT (MS-DOS, Windows) file system.

A VFAT file system does not support permissions however. The VFS layer therefore always presents a default set of permissions to the user. These permissions can be set when the file system is mounted but cannot be changed.

**Instructor notes:**

**Purpose** — Introduce the VFS.

**Details** — As before, keep the discussion basic. We do not have the material for an in-depth discussion of VFS or any trouble-shooting tips. We are only discussing its existence and the ability to support multiple file system types.

The virtual file system interface is structured around a number of generic object types and a number of methods which can be called on these objects.

The basic objects known to the VFS layer are files, file systems, index nodes (i-nodes), and names for i-nodes.

**Additional information** — All i-nodes within a file system are accessed by name. As the name to i-node lookup process might be expensive for some file systems, Linux's VFS layer maintains a cache of currently active and recently used names. This cache is referred to as the *dcache*.

The VFS layer does all management of path names of files and converts them into entries in the dcache before passing allowing the underlying file system to see them. The one exception to this is the target of a symbolic link, which is passed untouched to the underlying file system.

The underlying file system is then expected to interpret it. This seems like a slightly blurred module boundary.

The dcache is made up of lots of struct dentrys (name to i-node translations). Each dentry corresponds to one filename component in the file system and the object associated with that name (if there is one). Each dentry references its parent which must exist in the dcache. The dentry entries also record file system mounting relationships. The dcache is a master of the i-node cache. Whenever a dcache entry exists, the i-node will also exist in the i-node cache.

Conversely whenever there is an i-node in the i-node cache, it will reference a dentry in the dcache.

**Transition statement** — Okay, let's look at the impressive list of file systems that are supported.

## File systems supported

---

- Traditional: ext2
- Second generation: ext3, ReiserFS, IBM JFS, xfs
- Next generation: ext4, GFS2, Reiser4
- FAT-12, FAT-16, FAT-32, VFAT, NTFS (read-only)
- CD-ROM (ISO 9660)
- UMSDOS (UNIX-like FS on MS-DOS)
- NFS (Network File System)
- SMBFS (Windows share), NCPFS (Novell Netware share)
- /proc (for kernel and process information)
- SHMFS (Shared Memory File System)
- GPFS, Lustre (Clustering File Systems)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 10-5. File systems supported

LX158.0

### Notes:

#### Introduction

Linux supports a wealth of file systems. Its traditional file system is ext2, the second extended file system. As development continued, more advanced file systems were added. These include ext3, ReiserFS, IBM's JFS, and XFS. All have distinct advantages over ext2. More recently, we have ext4, Reiser4, and the addition of a number of clustering file systems, GPFS and Lustre the most notable of these.

Many file systems from other operating systems are also supported. NTFS, by default, is read only, however, packages do exist to add write support for certain NTFS versions.

**Instructor notes:**

**Purpose** — List file system types available in Linux.

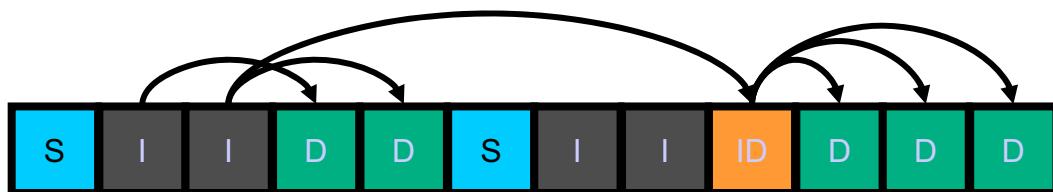
**Details** — Discuss student notes.

**Additional information** — We include JFS in this list, however, it should be pointed out that SUSE Linux has dropped this file system type with SUSE Linux Enterprise Server (SLES) 10, so we will not discuss JFS in any detail later.

**Transition statement** — We are not going to discuss every file system on this sheet. We will only discuss the standard Linux file systems, which, to a large extent, use the same internal structure. Let's look at that.

## Basic file system example: ext2

- Partition divided into blocks of 1024, 2048, or 4096 bytes
  - Block size depends on size of file system and expected usage
- Blocks can have different usage:
  - Superblock
  - Index node (i-node) block
  - Indirect block (double, triple)
  - Data block



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-6. Basic file system example: ext2

LX158.0

### Notes:

#### Introduction

Most file systems used on a Linux system share the same structures typical to a UNIX file system. When creating (formatting) the file system in the disk partition, the partition is split up in blocks of 1024 bytes each (default). Each block is given a specific function:

- Superblock
- I-node block
- Indirect block
- Data block

**Note:** It is not possible to combine functions in a block.

**Instructor notes:**

**Purpose** — Show the internal structure of a typical file system.

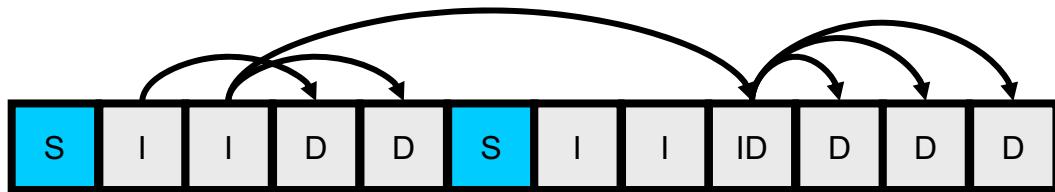
**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Let's look at the functions of these different blocks and start with the superblock.

# Superblock

- First block of file system, several copies (at 8193, 16385, ...)
- Contains general info on file system
  - Last mounted time/place
  - Block size
  - Pointers to free i-nodes
  - Pointers to free blocks
  - Pointer to root of file system
- Use **dumpe2fs** to view filesystem/superblock info



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-7. Superblock

LX158.0

## Notes:

### Introduction

The first block of the file system (block 1) is the superblock<sup>1</sup>. It is a very important block, since it contains information about the rest of the file system. Copies of this data are therefore kept (approximately on block 8193, 16385, and so on). Should block 1 become corrupt, then **mount** attempts to use the other superblocks.

The superblock contains general information about the file system, for instance, the time of last usage, the last used mountpoint, the blocksize, and so on. Furthermore, the superblock (indirectly) points to the list of free i-nodes and the list of free blocks. Last, the superblock contains an (indirect) pointer to the root directory of the file system.

To view the contents of the superblock you can use the **dumpe2fs** command.

<sup>1</sup> Block 1 is the second block in the partition. The first block of the partition, block 0, is never used for the file system since this block might contain a boot loader.

**Instructor notes:**

**Purpose** — Discuss the superblock.

**Details** — The superblock contains a description of the basic size and shape of this file system. The information within it allows the file system manager to use and maintain the file system. Usually only the superblock in block group 0 is read when the file system is mounted but each block group contains a duplicate copy in case of file system corruption.

**Additional information** — Among other information, it holds the:

*Magic Number*: This allows the mounting software to check that this is indeed the superblock for an EXT2 file system. For the current version of EXT2 this is 0xEF53.

*Revision Level*: The major and minor revision levels allow the mounting code to determine whether or not this file system supports features that are only available in particular revisions of the file system.

There are also feature compatibility fields which help the mounting code to determine which new features can safely be used on this file system.

*Mount Count and Maximum Mount Count*: Together, these allow the system to determine if the file system should be fully checked. The mount count is incremented each time the file system is mounted and when it equals the maximum mount count, the warning message maximal mount count reached, running e2fsck is recommended is displayed.

*Block Group Number*: The block group number that holds this copy of the superblock.

*Block Size*: The size of the block for this file system in bytes, for example, 1024 bytes.

*Blocks per Group*: The number of blocks in a group. Like the block size, this is fixed when the file system is created.

*Free Blocks*: The number of free blocks in the file system.

*Free Inodes*: The number of free i-nodes in the file system.

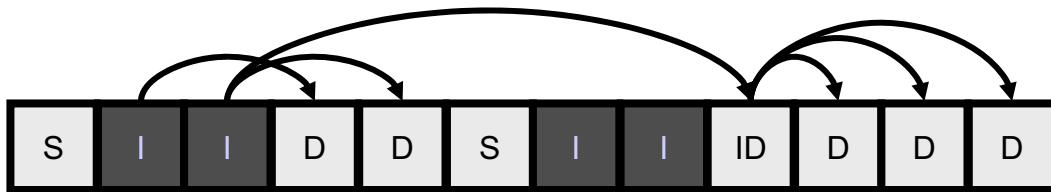
*First Inode*: This is the i-node number of the first i-node in the file system. The first i-node in an EXT2 root file system would be the directory entry for the '/' directory.

**Transition statement** — The next block type is the i-node block. As the name implies, this block contains i-nodes or index nodes.

## i-nodes

- 128 bytes (8 per block of 1024 bytes)
- Contains information about a file: owner, group, type, size, permissions, ctime, atime, mtime, ...
- Contains pointers to data blocks
- Contains pointers to an indirect block, a double indirect block, and a triple indirect block
- Use **stat** command to view inode info

Owner/=gGroup
File type
File size
File permissions
Time stamps: create time access time modification time
Link counter
Additional flags: (ACL, EXT2,_FLAGS)
Pointers to block data



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
 US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-8. i-nodes

LX158.0

### Notes:

#### Introduction

An i-node is 128 bytes large. With a blocksize of 1024 bytes, this means that there are eight i-nodes in a block. Each i-node contains information about a file: user/group information, permissions, size, creation time (ctime), last accessed time (atime) and last modified time (mtime).

It also contains information about the data blocks where the file resides. This structure is a little complicated but very efficient.

#### Structure

The first twelve data blocks (12 KB) are directly addressed; the block numbers are stored in the i-node itself.

The next data blocks are indirectly addressed. The i-node contains a pointer to an indirect block, and the indirect block contains the block numbers of the data blocks.

Since each pointer is four bytes, we can address 256 data blocks, assuming a blocksize of 1024 bytes.

The next 65536 (256\*256) data blocks are double indirectly addressed: the i-node contains a pointer to a double indirect block, the double indirect block contains pointers to indirect blocks, and the indirect block contains pointers to the data blocks (again assuming a blocksize of 1024 bytes).

The next 16777216 (256\*256\*256) data blocks are triple indirectly addressed. If you read this far you should be able to figure out how that works. The theoretical maximum filesize in the ext2 file system is therefore something like 16 GB when 1 K blocks are used.

To view all information in an inode, use the **stat** command.

## Instructor notes:

**Purpose** — Discuss i-nodes.

In the ext2 file system, the i-node is the basic building block; every file and directory in the file system is described by one and only one i-node. The ext2 i-nodes for each block group are kept in the i-node table together with a bitmap that allows the system to keep track of allocated and unallocated i-nodes.

**Details** —

**Additional information** — Among other information, the i-node contains the following fields:

**Mode**: This holds two pieces of information: what the i-node describes and the permissions that users have to it. For EXT2, an i-node can describe a file, directory, symbolic link, block device, character device, or FIFO.

**Owner Information**: The user and group identifiers of the owners of this file or directory. This allows the file system to correctly allow the right sort of accesses.

**Size**: The size of the file in bytes.

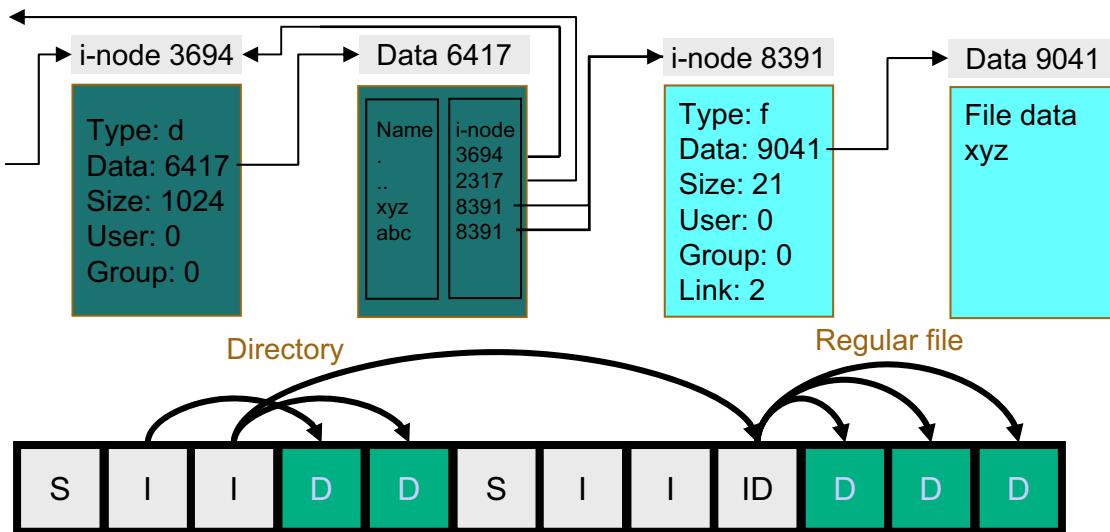
**Timestamps**: The time that the i-node was created and the last time that it was modified.

**Datablocks**: Pointers to the blocks that contain the data that this i-node is describing. The first twelve are pointers to the physical blocks containing the data described by this i-node, and the last three pointers contain more and more levels of indirection. For example, the double indirect blocks pointer points at a block of pointers to blocks of pointers to data blocks. This means that files less than or equal to twelve data blocks in length are more quickly accessed than larger files.

**Transition statement** — The last block type is the data block itself.

## Data blocks

- Contain file data
- File can be a directory, in which case the data is the list of file names and i-nodes in that directory.
- Multiple file names can point to the same i-node!  
(In other words, a file can have multiple names.)



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-9. Data blocks

LX158.0

### Notes:

#### Introduction

The data blocks finally contain the data the end user would refer to as a file.

A file might be of a special type: a directory. In this case, the data block contains the file names in that directory and the number of the corresponding i-node. This leads to a very interesting concept: a file can have multiple names, even in multiple directories, as long as the directories are on the same file system.

**Instructor notes:**

**Purpose** — Discuss data blocks and how they are used in a directory.

**Details** — Follow the example blocks in the visual, pointing out the relationship between them, along with the pointers to additional blocks not seen.

**Additional information —**

**Transition statement** — Let's summarize what you have learned so far about the ext2fs file system.

## Ext2fs summary

- The most important components of a file system are the i-nodes and the data blocks.
- The file system is full if:
  - No more i-nodes are available
  - No more data blocks are available
- So tune your file system according to the number of bytes per file:
  - Blocksize (1024, 2048, or 4096 possible)
  - Bytes per i-node (4096 default)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-10. Ext2fs summary

LX158.0

### Notes:

#### In review

It is not important to know the exact internal structure of the ext2fs file system. What is important to know is that there are two main components of a file system: i-nodes and data blocks. Any file needs an i-node and one or more data blocks. If there are no more i-nodes or data blocks available in the file system, the file system is full.

If you really want to use your file system to the limit, it is important to tune it according to the data you expect. The blocksize can be 1024, 2048, or 4096. This size is chosen when the file system is created, based on the expected usage of the system. If you expect a lot of small files, choose a small blocksize. If you expect a large number of large files, choose a larger blocksize. The bytes per i-node is 4096 by default. With a blocksize of 1024, this means that for every four data blocks there is one i-node available. If you expect a large number of small files, decrease this value since you will probably want one or two i-nodes per data block.

In general, it is easier to explain to the users why a file system is full if there are no more data blocks left than it is to explain that a file system is full because you ran out of i-nodes.

And since an i-node is smaller than a data block, you should usually overestimate the number of i-nodes, just to be sure. The default values of **mke2fs** also do this.

**Instructor notes:**

**Purpose** — Summarize what we have seen so far and list some implications.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — We have seen the common layout of a basic UNIX file system.

Now, let's look at some of the features that make a file system different from another.

# Other file system features

- File systems can have other features that can be useful:
- Access Control Lists (ACL)
  - Allow more extended permissions, not just rwxrwxrwx
  - The most commonly used file systems have ACL support
- Journaling
  - Keeps a journal of operations that are going to take place and operations that were successfully committed
  - Should make recovery from a crash faster
  - Slight performance decrease
  - Most modern file systems use journaling to some degree.
- Extended file attributes
  - Examples: immutable, auto compression, undeletable
- Labels
  - Allow mounting based on label instead of device name
- Performance optimizations

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-11. Other file system features

LX158.0

## Notes:

### Introduction

All file systems are able to store your files, possibly under multiple names. They also all support the default UNIX permissions (rwxrwxrwx). They do however, differ in the additional features that they can offer. Some of the features that can be offered by file systems are:

- **Access Control Lists (ACLs):** These are lists of user and/or group names with the permissions that these users/groups might have on the file. This allows you to set permissions that go further than the standard possibilities. It is possible, for instance, to define that a certain group is able to execute a program with the SUID bit set and that another group is able to execute it, but without the SUID bit.
- **Journaling:** This is a technique where every intended write action is first listed in a journal (a fixed-size file or other partition) and only then performed. If the action has succeeded, this is listed in the journal as well. This of course leads to a performance decrease but yields one important benefit: When the system crashes, you do not have to do an **fsck** of the whole disk to look for inconsistencies but just need to look at the

journal and retrieve all transactions that were started but not finished. Only the disk areas that were involved in those transactions need to be searched.

An **fsck** on a crashed journaled file system typically only takes a few seconds, while a non-journaled file system might easily take several minutes, depending on the size of the file system.

- **Extended File Attributes:** This allows you to specify additional attributes of a file. An example is the **immutable** flag, which prevents anyone from modifying or deleting the file (even root), as long as this flag is set.
- **Labels:** These are labels that are attached to the file system itself (in the superblock). This allows you to specify a file system label instead of a device name in your /etc/fstab file. The advantage of this is that if you add or remove any disks and/or partitions, your file systems can still be found, even though they might now be located on a differently named device. At the point of this writing, only the ext2/ext3 file system supports labels, and only Red Hat is configured to use them.

Apart from this, file systems also differ in various optimization details. For example:

- File systems like ReiserFS and JFS do not use a linear list to hold the contents of a directory, but use binary or B+ trees for this. These trees are far faster to search and thus increase performance if you have a large number (1000 or more) files in one directory. This typically happens on news servers, for instance.
- Some file systems use a variable number of i-nodes, which are added and deleted when needed. This avoids the problem of running out of i-nodes while you still have data blocks left.
- File systems might also use data blocks more efficiently by storing multiple, smaller files in one data block.
- Some file systems can work efficiently with *sparse files*. Sparse files are files which are mostly empty. They are the result of programs that open a new file for writing, and then **Iseek** to a location somewhere in the file to write something there. The area before the written area is empty and need not be saved on disk until the program actually starts writing there. Sparse files are common in databases.
- Newer file systems offer a contiguous area of storage, called an *extent*, for a file when the file starts to be written to. Extents help to reduce file fragmentation and scattering.

***Instructor notes:***

**Purpose** — Discuss other distinguishing features.

**Details** — The following visuals will cover more details about various types of file systems. Use this visual as a transition from the previous visuals that dealt with structural issues.

**Additional information —**

**Transition statement** — Once we have decided what file system we are going to use, we will have to create it.

## Creating a file system

- Creating a file system is done with an **mkfs** variant.
  - **mke2fs**, **mke2fs -j**, **mkfs -t**
  - **mkreiserfs**
  - **mkjfs**
- Typical options:
  - **b** blocksize sets blocksize
  - **i** bytes per i-node sets number of i-nodes
  - **c** checks disk for bad blocks
- Example:

```
# mke2fs -b 1024 -i 4096 -c /dev/sda6
...
Writing inode tables: done
Writing superblocks and filesystem
accounting info: done
...
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-12. Creating a file system

LX158.0

### Notes:

#### Introduction

Once we have decided which block device we are going to use, and the type of file system we want, we create it. This is usually done with some variation on the **mkfs** command, such as **mke2fs**<sup>1</sup>, **mkreiserfs** or **mkjfs**.

Typical options include the blocksize to use and the bytes per i-node number. This last number determines the number of i-nodes to create on the file system and should reflect the average size of the files on your file system, rounded down to the nearest 2<sup>n</sup> kilobytes (1024, 2048, 4096, ... bytes).<sup>2</sup>

<sup>1</sup> **mke2fs** is used to create an ext2 file system. **mkfs -t ext3** creates an ext3 file system. **mkfs -t ext4** creates an ext4 file system.

<sup>2</sup> If you round up rather than down, then you will run out of i-nodes before you run out of data blocks. That is harder to sell to your users.

**Instructor notes:**

**Purpose** — Discuss the creation of file systems.

**Details** — Remind the student that Linux is a dynamic environment, and file system support will change over time. An example is that under SLES, the JFS file system will no longer be included.

At the moment, RHEL uses ext4 by default, while SLES uses ext3. SLES previously used ReiserFS.

**Additional information —**

**Transition statement** — Once a file system is created, you can mount it.

# Mounting a file system

- Using the **mount** command:
  - Supply device name
  - Supply mount point (empty directory)

```
# mount -t ext3 /dev/sda6 /mnt/extra
# mount -o nodev,noexec /dev/system/mylv /usr/local/proj1
```

Optional: Supply file system type

Optional: Supply other options

Optional: Use different superblock

- To show mounted file systems, use **mount** without arguments.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-13. Mounting a file system

LX158.0

## Notes:

### Introduction

Mounting a file system is done with the **mount** command. The syntax is:

**mount [-t <type>] [-o <options>] <device name> <mount point>**

For instance: **mount -t iso9660 -o ro /dev/cdrom /mnt/cdrom** to mount the CD-ROM device /dev/cdrom, which contains an iso9660 file system on the mount point /mnt/cdrom, read-only.

To show all mounted file systems, use the **mount** command without arguments:

```
[root@sys2 /root]# mount
/dev/sda2 on / type ext3 (rw)
/dev/sda6 on /mountpoint type ext3 (rw)
/dev/cdrom on /mnt/cdrom type iso9660 (ro)
none on /proc type proc (rw)
[root@sys2 /root]#
```

***Instructor notes:***

**Purpose** — Discuss mounting of file systems.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Sometimes mounting should be automatic at system startup.  
Then we place the entries in /etc/fstab.

# Mounting file systems at system startup

- Add to /etc/fstab:

```
/dev/sda1  /boot        ext3      defaults      1  2
/dev/sda5  /           ext3      defaults      1  1
/dev/cdrom /mnt/cdrom  iso9660  noauto,ro,user 0  0
/dev/fd0   /mnt/floppy msdos    noauto,user   0  0
/dev/sda6   /mnt/extra  ext3      defaults      0  0
```

- Alternative notation, using ext2/ext3 file system labels:

```
LABEL=/boot /boot        ext3      defaults      1  2
LABEL=/   /           ext3      defaults      1  1
/dev/cdrom /mnt/cdrom  iso9660  noauto,ro,user 0  0
/dev/fd0   /mnt/floppy msdos    noauto,user   0  0
LABEL=extra /mnt/extra  ext3      defaults      0  0
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-14. Mounting file systems at system startup

LX158.0

## Notes:

If file systems need to be mounted automatically at system restart or if you need to create shortcuts for fast mounting of common file systems, add them to /etc/fstab. This file contains lines for each file system to be mounted. Every line consists of six fields:

- The block device which contains the file system.

Recent kernels also allow a “label” to be specified here, instead of the device. This is the label that is stored in the ext2/ext3 superblock. The kernel searches all ext2/ext3 file systems for the file system holding this label and mounts the first file system where the label matches. This is very useful if you make changes to your partition tables or the order of your disks (in particular, SCSI disks). Labels are currently only supported on ext2/ext3 file systems, and the use of these labels also requires modifications of utilities (for example, **mount**) and scripts.

- The mountpoint at which the file system needs to be mounted.

- The type of the file system. Recent kernels also allow the “auto” type, which indicates that the kernel itself should try to figure out the file system type. This is useful for removable media, in particular floppy disks.
- The options.
- A **dump** indicator (see **man fstab**).
- A sequence indicator for **fsck** (see **man fstab**).

**Instructor notes:**

**Purpose** — Discuss the mounting of file systems.

**Details** — Discuss student notes.

**Additional information** — In the /etc/fstab file are also entries for the swapspaces. They are left out here deliberately.

**Transition statement** — There are several options we can use when mounting file systems.

## Mount options

- Various options can be used when mounting a file system:
  - **auto**: Mount file system automatically when booting.
  - **noauto**: Do not mount fs automatically.
  - **user**: Users are allowed to mount this file system.
  - **owner**: Same as user but user must be owner of device
  - **ro**: Read-only
  - **rw**: Read-Write
- For more options, see **man mount**.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-15. Mount options

LX158.0

### Notes:

#### Introduction

There are various options you can specify when mounting a file system. These options change the way the file system behaves while accessing it.

Options can be specified both when mounting a file system manually, by using the **-o** flag, and can be specified in the /etc/fstab file in the fourth column. In both cases, it is important that options should be separated by commas and not by spaces.

Some important options include:

- **noauto**: Do not automatically mount the file system at startup. If this is not specified, the file systems will automatically be mounted at system startup or when issuing the **mount -a** command.
- **user**: Allow ordinary users to mount this file system. Handy for floppy and CD-ROM drives. Only the user that mounted the file system can unmount it.
- **users**: Same as **user**, but every user can unmount the file system.

- **owner:** Same as **user**, but with the restriction that the user that wants to mount the file system has to be the owner of the device.
- **- ro:** Mount the file system read-only.
- **- nodev:** Do not allow usage of block and character special devices on the file system.
- **- noexec:** Do not allow execution of programs on the file system.
- **- nosuid:** Do not allow SUID and SGID bits to take effect. **nodev**, **noexec**, and **nosuid** are mainly used for security reasons. For more options, see **man fstab** and **man mount**.

***Instructor notes:***

**Purpose** — Discuss mount options.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Let's see how to unmount a file system.

## Unmounting file systems

- File system might not be in use: Check with **fuser**.
- Open files
- Programs being executed
- Active directories

```
# fuser -v /usr
          USER           PID      ACCESS COMMAND
/usr      root           Kernel    mount   /usr
```

- Use the **umount** command with either:
  - The device name
  - The mount point
  - Or both

```
# umount /dev/cdrom
# umount /mnt/cdrom
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-16. Unmounting file systems

LX158.0

### Notes:

#### Introduction

Unmounting a file system is done with the **umount** command (**Note**: not **unmount**). You either have to supply the device name or the mount point, and **umount** will figure out the rest.

If file systems are defined in **/etc/fstab**, you can unmount them all with one command:

**umount -a**

Or you can unmount all file systems of a given type:

**umount -t msdos -a**

## ***Instructor notes:***

**Purpose** — Discuss unmounting of file systems.

**Details** —

**Additional information** — Most CD-ROM players in modern computers are ATAPI CD-ROMs, meaning that they are connected to the Intelligent Drive Electronics (IDE) chain like a normal hard disk instead of connected to a special CD-ROM adapter card or sound card. The real device the CD-ROM uses in that case is therefore /dev/hdb (slave on first IDE chain) or /dev/hdc (master on second IDE chain). /dev/cdrom is a symbolic link to the real device.

**Transition statement** — Let's look at some useful commands when playing with file systems.

## Checking a file system

- Checking a file system is done automatically when the system boots.
  - If a file system is cleanly unmounted, no further checks are done.
  - Minor errors repaired automatically.
  - Major errors drop you in a shell; allows you to do a more thorough check manually.
    - `fsck -y /dev/sda1`
- Can start file system checks manually as well with **fsck**.
  - Only on file systems that are mounted read-only or not mounted at all

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-17. Checking a file system

LX158.0

### Notes:

#### Introduction

It is of the utmost importance that the internal structure of a file system is at a consistent state at all times. The Linux kernel is designed to achieve this. On the other hand, for performance reasons, the file system is not updated synchronously with all user program writes. This is called *write caching* and means that a write action by a user is not necessarily automatically done on disk. In fact, it might take up to 30 seconds for this to be done.

If the system crashes, for instance because of a power failure, the file system is left in an unstable state and needs to be repaired before it can be used. This is done by running the **fsck** program, usually from **rc.sysinit**. **fsck** detects the type of file system and runs the specific check program accordingly.

#### Implementation

Although the implementation details might change, the general behavior of all these **fsck** programs is always the same:

- When the **fsck** program detects that the file system was unmounted cleanly, then no further checks are performed.<sup>4</sup>
- If the file system was not clean, the consistency will be checked. On a non-journalized file system, this basically means that the whole file system needs to be scanned, while a journalized file system only needs to scan the file system areas which are listed as possibly dirty here.
- If minor errors are detected, then these are usually corrected automatically.
- If major errors are detected, then the system drops you into a shell and you need to fix these errors manually.

This is typically done with the **fsck -y** command.

File system checks can also be started by hand. This can only be done on file systems that are not mounted at all, or are mounted read-only.

4 *Cleanly unmounted* means that the file system was properly unmounted. This allows the kernel first to bring the file system in a consistent state, where all cached write actions are actually written out. As the last action, the kernel writes the “clean” bit to the superblock.

**Instructor notes:**

**Purpose** — Discuss file system checks.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Well, that is the general information on file systems. Let's look at some details of various file systems as well. We will start with the ext2/ext3 file system.

## ext2/3/4-specific information

- ext3/4 adds journaling to ext2 using a special, hidden .journal file of arbitrary size (recommended: 10 MB)
  - Thus, downwards compatible with **ext2**
  - For new ext3 file systems, use **mke2fs -j**
  - For converting ext2 to ext3, use **tune2fs -j**
- ext4 adds performance optimizations to ext2/3 for large files
- Useful ext2/3/4 commands:
  - **tune2fs** tunes an ext2/3/4 file system
  - **debugfs** debugs an ext2/3/4 file system
  - **chattr** changes ext2/3/4 extended attributes of a file
    - Immutable, Compressed, Undeletable and so forth (see **man chattr** for details)
  - **e2label** changes file system label of an ext2/3/4 file system
  - **resize2fs** can increase the size of a mounted file system and decrease the size of an unmounted file system

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-18. ext2/3/4-specific information

LX158.0

### Notes:

#### ext3

The ext3 file system standard adds journaling capability to the ext2 file system standard. This is implemented using a special, hidden .journal file. The file size of this file is arbitrary, but 10 MB is recommended.

Because of this implementation method, the file system is fully compatible with ext2. It is therefore really easy to upgrade to ext3.

When creating an ext3 file system, use **mke2fs -t ext3**. When upgrading an existing ext2 file system, run the **tune2fs -j** command.

Downgrading ext3 to ext2 is easy too since any (cleanly unmounted) ext3 file system can be mounted as ext2.

#### ext4

The ext4 file system is a further development of ext3. It has larger limits for large files, and adds performance optimizations.

One of the performance optimizations is the use of extents. An extent is a consecutive series of data blocks which are managed as one single unit. For large files, this means that the inode only needs to contain a single reference to an extent, instead of thousands of (direct, indirect, double indirect, triple indirect) references to data blocks.

## File system commands

Some tools that might be useful on an ext2/ext3/ext4 file system are:

- **tune2fs**: Tune an ext2 file system. This allows you to alter the number of i-nodes on your file system, for instance.
- **debugfs**: This allows you to debug an ext2 file system. It allows you to retrieve all information from superblocks, directories and i-nodes, for instance.
- **chattr**: Change attributes of files on an ext2 file system. Files on an ext2 file system can have a number of additional attributes, which can be useful in some situations. Note that not all attributes are currently implemented by the Linux kernel.
- **e2label**: Change the file system label in the superblock. This label can be used in the first column of your /etc/fstab file.
- **resize2fs**: Resize an ext2/ext3 file system. Increasing size can be done on a mounted file system; decreasing size requires the file system to be unmounted.

***Instructor notes:***

**Purpose** — Discuss some ext2/ext3 specific information and commands.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Let's look at some details about ReiserFS too.

## ReiserFS-specific information

- File system for Linux only, created by Hans Reiser
- 32 MB journal by default (minimum 2 MB)
  - Thus, do not use ReiserFS for small file systems.
  - Journal can be in the file system itself or on a separate partition.
- Uses balanced trees instead of linear directory lists.
  - Extremely useful for directories that contain 1000+ files
- Useful commands:
  - **debugreiserfs** debugs a ReiserFS file system.
  - **resize\_reiserfs** resizes a ReiserFS file system.
    - Extending can be done on a mounted file system.
    - Reducing can only be done on an unmounted file system.
  - **reiserfsck** runs a file system check on a ReiserFS.
- Due to legal issues, no longer used as the default file system type in modern distributions.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-19. ReiserFS-specific information

LX158.0

### Notes:

#### Introduction

ReiserFS is a file system that was designed specifically for Linux by Hans Reiser. A few features stand out:

- Journal structure
- Balanced tree design
- Block suballocation

#### Journal structure

ReiserFS uses a 32 MB journal by default. (Its minimum size is 513 blocks of 4 KB each, so a little over 2 MB.) This journal is usually part of the file system, but it can also be stored in a separate partition.

#### Balanced tree design

ReiserFS uses its balanced trees to streamline the process of finding the files and retrieving their security (and other) metadata. For extremely small files, the entire file's data

can actually be stored physically near the file's metadata so that both can be retrieved together with little or no movement of the disk seek mechanism. If an application needs to open many small files rapidly, this approach significantly improves performance.

### Block suballocation

ReiserFS attempts to reduce file fragmentation by using block suballocation. This allows large blocks to be used, but allows use of the unused space at the end of large files.

### ReiserFS file system commands

Some useful commands for ReiserFS are:

- **debugreiserfs**: Debugs a ReiserFS file system.
- **resize\_reiserfs**: Resizes a ReiserFS file system.
- **reiserfsck**: Runs a file system check on a ReiserFS.

**Instructor notes:**

**Purpose** — Discuss ReiserFS-specific issues.

**Details —**

**Additional information** — All file systems listed in this unit implement the concept of a directory as follows:

- The i-node describing the directory contains the file type “d”
- The data blocks associated with the directory contain the list of file names in that directory, along with their associated i-node number.

The difference between the file systems is the way the list of file names is ordered. With the ext2/ext3 file system, this is a linear list, which can only be searched sequentially. That means that if you have 1000 files in your directory and you want to pick up a particular file, you have to search, on average, 500 file names before you get a match. This is really inefficient. On the other hand, a linear list is really easy to implement: It is one of the first data structures that a student programmer learns to program.

Both ReiserFS and JFS have implemented their directories with another data structure, called a binary tree, or variants thereof. In this data structure, all filenames do not form a linear list, but form a hierarchical structure where each node can contain pointers to at most two nodes (compared with a tree: each branch splits into two branches at most.) This data structure, although harder to program, is far more efficient: With a well-balanced binary tree of 1000 nodes, you have to search only about 10 filenames on average to get a match.

A lot of research has been done on creating efficient binary trees, and a lot of variations have been developed: balanced trees, B+ trees, and so forth. Different file systems apply these different efficiency techniques, but their basic concept is the same and leads to a vast performance increase for directories with a large number of files.

For a good introduction to binary trees, refer to <http://cslibrary.stanford.edu/110/>.

The **legal issues** referenced in the sheet refer to the author of the ReiserFS file system, Hans Reiser. Hans was convicted of first-degree murder of his wife in April 2008 and is now serving a 15 years to life sentence in Pleasant Valley State Prison in Fresno County, California. With Hans in prison, continuing support and development of the ReiserFS file system has become very problematic. This is the reason that SUSE dropped ReiserFS as its primary file system. Older distributions using the ReiserFS file system will continue to be supported though. For more information refer to the Wikipedia page on Hans Reiser.

**Transition statement** — Let's quickly compare some of the more popular file systems

# Comparing file systems

Journaled file systems used by Linux:

	<b>ext2</b>	<b>ext3</b>	<b>ext4</b>	<b>jfs</b>	<b>reiser</b>
<b>Journal</b>	No	Yes (10 MB default)	Yes (10 MB default)	Yes (auto resized)	Yes (32 MB default)
<b>Resizeable</b>	Yes, but only when unmounted	Yes	Yes	Yes	Yes
<b>Maximum size</b>	File: 2 TB FS: 16 TB	File: 2 TB FS: 16 TB	File: 16 TB FS: 1 EB	File: 4 PB FS: 32 PB	File: 16 TB FS: 1 EB
<b>Type</b>	i-nodes (completely block oriented)	i-nodes (completely block oriented)	i-nodes (blocks and extents)	i-nodes (allocated in a b-tree)	b-tree

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
 US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-20. Comparing file systems

LX158.0

## Notes:

### Introduction

The visual shows a quick comparison of the file systems found on various Linux distributions. Note that the maximum size for files and file systems shown is always calculated when the maximum blocksize is used. In addition to this, your application might not support files or file systems of the sizes mentioned.

**Instructor notes:**

**Purpose** — Give a quick comparison between different file system types.

**Details** — Our focus is on ext3 and ReiserFS, because those are the primary file system types of the three distributions we are demonstrating. That being said, there are other file system types your students might run into. This is a chance to bring those up.

**Additional information —**

**Transition statement** — There is also something called Shared Memory File System (SHMFS).

## SHMFS-specific information

- SHMFS: POSIX compliant Shared Memory File system
- File system stored in memory, expands when used to required size
- Not persistent across reboot
- Typically mounted on /dev/shm
- Required by certain applications

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-21. SHMFS-specific information

LX158.0

### Notes:

#### Introduction

The last file system we want to cover here is the Shared Memory File System (SHMFS). It is a POSIX compliant file system, which resides in memory. You can think of SHMFS as a RAM disk formatted as a file system. The difference is that SHMFS automatically grows and shrinks as needed, while formatting a RAM disk would yield a file system with a fixed size. But as with a RAM disk, it is not persistent across a reboot.

Most distributions enable an SHMFS by default, typically mounted on /dev/shm. Red Hat does this with an entry in /etc/fstab, while SUSE Linux enables it from /etc/init.d/boot.swap, which in turn is called from /etc/init.d/boot. The maximum size of the SHMFS is configurable but is usually set to half the amount of real memory in the system.

SHMFS is required by some applications, such as Apache HTTP Server, Oracle, and SAP.

**Instructor notes:**

**Purpose** — Discuss SHMFS.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Well, that covers our file systems section. There is just one feature left that is supported on most file systems, and that is quota support.



## 10.2.File system quota

### Instructor topic introduction

**What students will do** — Learn about Quota

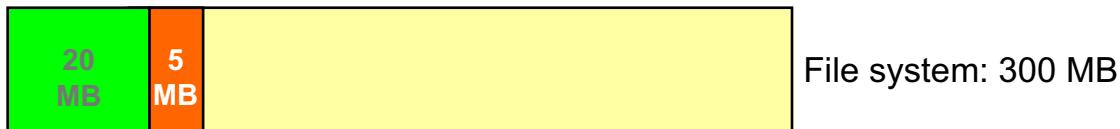
**How students will do it** — Lecture and Exercises

**What students will learn** — Manage Quota

**How this will help students on their job** — Quota allow the student to limit the file system usage of their users.

## Quota concepts

- Quotas limit the amount of data a user/group is allowed to store.
- Defined on a per file system basis
- Based on block and/or i-node usage per user or group
- Two limits per quota: Soft and hard
  - User exceeds soft limit → warning only
  - User exceeds hard limit → error
- Grace period identifies how long the soft limit can be exceeded.
  - After that period, a user gets errors instead of warnings.



Each user can consume only 20 MB permanently and 25 MB temporarily.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-22. Quota concepts

LX158.0

### Notes:

#### Introduction

Quotas are used to limit the amount of data a user can store on a specific file system. A user can have a different quota on different file systems. Quota are usually based on the amount of disk blocks a user has in use, although you can also put limits on the number of i-nodes. In addition to that, you can also create group quotas, which limit the number of blocks/i-nodes a group can use.

A user quota is usually made up of two numbers: the so-called *soft limit* and the *hard limit*. When a user (or group) exceeds the soft limit, the user will receive warnings that the quota limit has been exceeded, but the operation will succeed. When a user tries to exceed the hard limit, the operation will fail.

As soon as the user exceeds the soft limit, the grace period will start. When that period is over, the user will get errors instead of warnings when he or she tries to write files. Thus, by setting the soft limit and the grace limit to a reasonable value, users are able to exceed their soft limit for a short period of time, usually just enough to request a quota upgrade or perform a clean up of their allocated space.

***Instructor notes:***

**Purpose** — Introduce quota.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Let's see how Linux implements all of this.

## Quota implementation on Linux

- Quota support compiled into the kernel
  - No daemon necessary
- Implemented on a per file system basis
  - A user can have different quota on different file systems.
  - Stored in aquota.user and aquota.groups in the root of the file system
- Quota checking should be enabled when mounting the file system.
  - Mount options: **usrquota, grpquota**
  - Can be specified in /etc/fstab
- Quota checking should be turned on after mounting with the **quotaon** command.
  - Automatically executed from bootscript after **mount -a**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-23. Quota implementation on Linux

LX158.0

### Notes:

#### Introduction

Quota support in Linux is compiled into the kernel, so you do not need to run extra daemons. What you do need to do is indicate that a certain file system uses quota when that file system mounts. This is done with two mount options: **usrquota** and **grpquota**. After mounting, you need to turn quota on with the **quotaon** command.

In addition to that, you also need to specify the quota themselves. This is done in the files aquota.user and aquota.group in the root of the file system.

***Instructor notes:***

**Purpose** — Show the Linux implementation of quota.

**Details** — Discuss student notes.

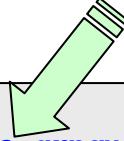
**Additional information** —

**Transition statement** — So what are the exact steps?

## Enabling quota

- Modify /etc/fstab

/dev/sda2	/	ext3	defaults	1	1
/dev/sda4	/home	ext3	defaults,usrquota,grpquota	1	2
/dev/sdb	/mnt/cdrom	iso9660	noauto,owner,ro	0	0
/dev/sda3	swap	swap	defaults	0	0
/dev/fd0	/mnt/floppy	msdos	noauto,owner	0	0
none	/proc	proc	defaults	0	0
none	/dev/pts	devpts	gid=5,mode=620	0	0



- Create aquota.user and aquota.group in the file system's root directory.
- Remount the partition.
- Calculate current usage and turn on quota checking.

```
# touch /home/aquota.user /home/aquota.group
# mount -o remount /home
# quotacheck /home
# quotaon /home
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-24. Enabling quota

LX158.0

### Notes:

#### Introduction

So how do we enable quota? The first step is to change the /etc/fstab file to indicate that a certain file system uses quota. Because you might want to enable quota every time the system boots, that's why we specify it here.

The next step is remounting the partitions. This ensures that all options are re-read from the /etc/fstab file. You can remount a file system using **mount -o remount** or by simply rebooting the machine.

Now that quota is enabled on this file system, we need to calculate the actual usage and store this in the aquota.user and aquota.group file. This is done with the **quotacheck** command.

Finally, we have to turn the quota on with the **quotaon** command. Quota checking is now fully functional.

**Instructor notes:**

**Purpose** — Show how to enable quota on a file system.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — We can now specify the individual quota for each user and group.

## Configuring quota

- Done with the edquota command
  - Starts **\$EDITOR** (default: **vi**) in a subshell.
  - Only edit the block/i-node soft/hard quota number.
- User quota: **edquota -u username**

```
Disk quotas for user tux1 (uid 501):
Filesystem    blocks    soft    hard    inodes    soft    hard
/dev/sda4       10700   20000   25000        407       0       0
/dev/sda9         320     300     350        23       30      50
~
~
~
"/tmp/Edp.a9fSEQK" 3L, 213C
```

- Group quota: **edquota -g groupname**
- Grace period: **edquota -t**
- Copy quota: **edquota -p tux1 -u tux2 tux3 tux4**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-25. Configuring quota

LX158.0

### Notes:

#### Introduction

After quota checking is turned on, we can specify the quota per user or group. This is done with the **edquota** command.

The **edquota** command reads the **aquota.user** and **aquota.group** files (which are binary files), extracts the relevant information, and writes it to a temporary file. It then starts your favorite editor (identified with the **\$EDITOR** shell variable) and lets you edit this temporary file. After you finished, it reads the contents of the temporary file and merges it back into the **aquota.user** and **aquota.group** file. For this reason, you should be careful editing the temporary file. If you change the wrong fields, **edquota** will get confused and will not do what you expect it to do. You are only supposed to edit the fields under **soft** and **hard**: four fields per file system in total.

The syntax of **edquota** is really straightforward. Use the **-u** option to edit user quota, use the **-g** option to edit group quota, and use the **-t** option to edit the grace period (which is the same for everyone on the system).

---

A very useful feature of edquota is the copying of quota information. If you want the users tux2, tux3, and tux4 all to have the same quota limits as the user tux1, just run the command **edquota -p tux1 -u tux2 tux3 tux4**, and you are done.

***Instructor notes:***

**Purpose** — Introduce edquota.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Now, let's see how to verify things.

## Quota information

- **quota** command
  - Reports on the quota of one user
  - Can be executed by anyone
  - A regular user can only view his own quota

```
tux1$ quota
Disk quotas for user tux1 (uid 501):
Filesystem blocks quota limit grace files quota limit grace
/dev/sda4     10700 20000 25000           407      0      0
```

- **repquota** command
  - Reports on the quota of all users and groups
  - Can only be executed by root

```
root# repquota /dev/hda4
                                         Block limits                               File limits
User          used    soft    hard   grace    used    soft    hard   grace
root        --    848804      0      0
.
tux1        ++     1500    1000    1500  7days    112    112    115  none
tux2        --     176     1000    1500
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-26. Quota information

LX158.0

### Notes:

#### Introduction

If you need to know how you are doing with the quota, there are two commands available:

The **quota** command shows the quota of one individual user. It can be executed by anyone on the system, but a regular user can only see his own quota.

The **repquota** command shows all quota information of all users and groups. It can only be executed by root.

***Instructor notes:***

**Purpose** — Introduce the **quota** and **repquota** commands.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — That is it. Let's see the checkpoint questions.

## Checkpoint

1. Assuming a blocksize of 1024, how many i-nodes and data blocks do you need for a file on an ext2 file system?
  - a. With size 0?
  - b. With size 1?
  - c. With size 2000?
  - d. With size 12289 (12 K+1)?
2. What are the two methods of copying a file to a (not yet mounted) MS-DOS floppy?
3. What files are important with respect to quotas?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-27. Checkpoint

LX158.0

### Notes:

## Instructor notes:

Purpose — Check student progress.

Details —

## Checkpoint solutions

---

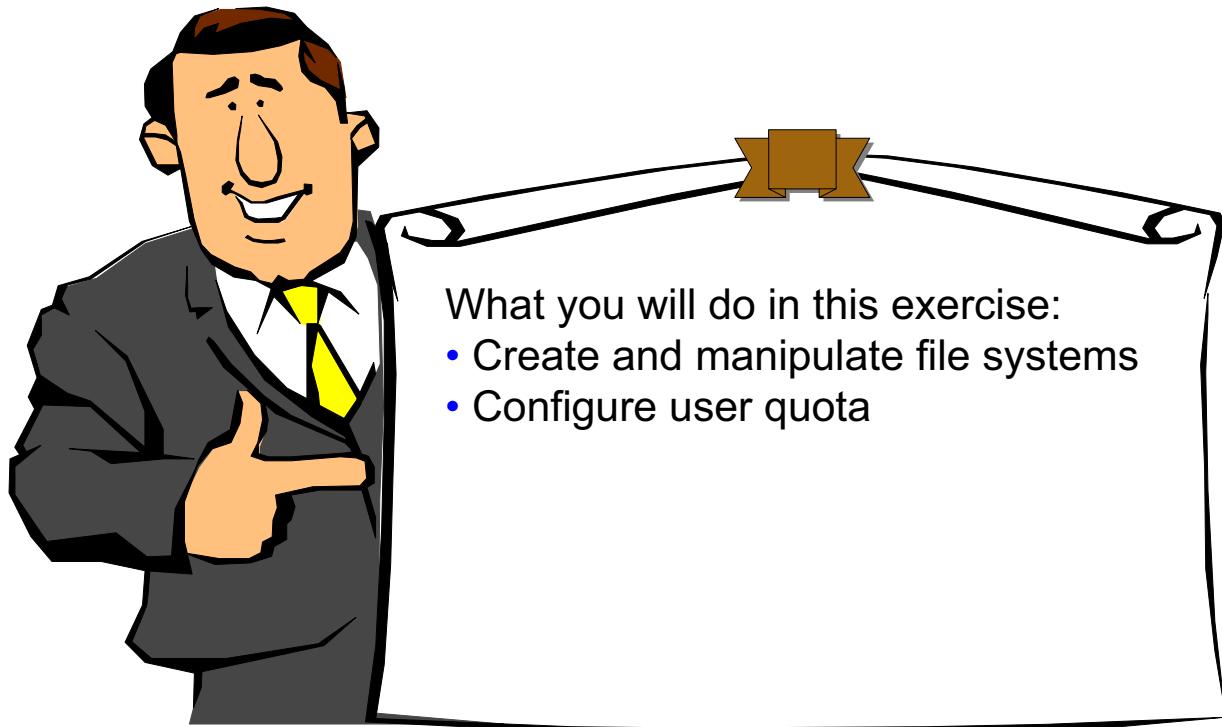
1. Assuming a blocksize of 1024, how many i-nodes and data blocks do you need for a file on an ext2 file system?
  - a. With size 0? [The answer is 1 i-node and 0 data blocks.](#)
  - b. With size 1? [The answer is 1 i-node and 1 data block.](#)
  - c. With size 2000? [The answer is 1 i-node and 2 data blocks.](#)
  - d. With size 12289 (12 K+1)? [The answer is 1 i-node and 12 data blocks directly from the i-node, an indirect block, and an extra data block. Total 14 data blocks.](#)
2. What are the two methods of copying a file to a (not yet mounted) MS-DOS floppy?  
[The answer is mount file system, and use cp command and use mcopy \(from mtools\).](#)
3. What files are important with respect to quotas?  
[The answer is /etc/fstab to specify file systems and /quota.users and /quota.groups.](#)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Additional information — Next, let's proceed with the exercise.

Transition statement —

## Exercise: File systems and file system quota



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-28. Exercise: File systems and file system quota

LX158.0

### **Notes:**

***Instructor notes:***

**Purpose** — Introduce the exercise.

**Details** — Discuss with students how the exercise will proceed. Set a time limit for beginning the next unit.

**Additional information —**

**Transition statement** — Let's summarize what we have learned in this unit.

## Unit summary

Having completed this unit, you should be able to:

- Describe what a file is
- Describe what a file system is
- List possible file systems
- Describe i-nodes
- Create/mount/unmount file systems
- Create predefined mounts
- Set up user and group quota

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 10-29. Unit summary

LX158.0

### Notes:

***Instructor notes:***

**Purpose** — Summarize unit topics.

**Details** —

**Additional information** —

**Transition statement** — Now, on to the next unit.

# Unit 11. Kernel services, kernel configuration and device management

## Estimated time

01:00

## What this unit is about

This unit covers the basics of the Linux kernel, the services it offers and basic configuration of the kernel. It also covers Device Management.

## What you should be able to do

After completing this unit, you should be able to:

- Describe the base characteristics of the Linux kernel
- Work with kernel modules
- List the most important kernel services
- Configure the Linux kernel
- Work with the /dev directory
- Know the traditional difference between block and character devices
- Work with virtual character devices
- Work with the loop device
- Retrieve device information

## How you will check your progress

- Checkpoint questions
- Lab exercises

## Unit objectives

---

After completing this unit, you should be able to:

- Describe the base characteristics of the Linux kernel
- Work with kernel modules
- List the most important kernel services
- Configure the Linux kernel
- Work with the /dev directory
- Know the traditional difference between block and character devices
- Work with virtual character devices
- Work with the loop device
- Retrieve device information

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 11-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**



## **11.1. Kernel, kernel services and kernel configuration**

### **Instructor topic introduction**

**What students will do —**

**How students will do it —**

**What students will learn —**

**How this will help students on their job —**

# Linux kernel

- First released by Linus Torvalds in 1991
- Licensed under GPL v2 (since kernel 0.12 in 1992)
- Ported to 20+ hardware architectures
- Modularized, monolithic design
  - Modules are loaded when needed
  - One address space; kernel and modules run in "ring 0"
- Main features:
  - Preemptive multitasking
  - Multi-threading
  - Virtual memory
  - Shared libraries
  - TCP/IP support

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-2. Linux kernel

LX158.0

## Notes:

A "kernel" is the core component of any operating system. It is the program that runs 24 hours a day, and takes care of core tasks such as process management, memory management, low-level networking tasks, security management, device management and so forth.

Linus Torvalds started working on the Linux kernel in April 1991. He released his work in August 1991 on the Internet, via a post in the newsgroup comp.os.minix. Initially the kernel was released under a self-authored license, but starting version 0.12 the kernel was released under the GPL version 2. People started using and improving the kernel, and the end result is that we now have an enterprise-grade UNIX kernel which is available on the Internet as an Open Source product.

The kernel has been ported to over 20 hardware architectures, including IBMs Power and System z architecture.

The Linux kernel uses a monolithic design that has been modularized. This means that the kernel is in essence one big program that runs in a single address space, and where each

---

part of the kernel can access all devices. Due to the sheer size however, parts of this program are distributed as modules, and only loaded when needed.

On many modern CPUs, multiple security levels, known as "rings" exist. Programs that run in "ring 0" have no security enforced on them by the CPU at all. The higher your ring number, the more security the CPU enforces. Microkernels, such as the GNU Hurd kernel, consist of a large number of small components, and these components run in different rings as required. The components communicate with each other by way of "message passing". This is inherently more secure and therefore considered a "modern" design. However, there is a lot of overhead in message passing, and the design of a microkernel is inherently very complex to debug. For these, and for historic reasons, Linux doesn't really use these CPU security features. All modules run alongside the core kernel image in "ring 0".

Linux supports all features that are expected of a modern UNIX kernel, including:

- Preemptive multitasking: Programs are automatically halted by the kernel (using specific CPU features) when their time is up and another process needs to run, or when an interrupt comes in that needs to be handled first.

Contrast this with "cooperative multitasking", where programs need to relinquish the CPU on their own initiative.

- Multi-threading: Programs are able to split themselves in multiple threads. These threads are individual execution units that share their address space.

This allows one program to handle multiple tasks simultaneously. This is useful if users want to have a certain task (a print job for instance) to be executed in the background, or if the architecture has multiple CPUs available (either multiple cores or SMT).

- Virtual memory: This means that programs do not have direct access to the real memory that is installed in the system. Instead, each program is presented with its own virtual address space. Through memory mapping, the individual pages that make up this virtual address space are mapped to pages in real memory.

This makes memory management for the programmer of applications really easy. But it also provides a mechanism where the kernel can swap out unused pages to disk (swap space), or share pages of program code between different processes.

- Shared libraries: These are files containing common program code that are used in a large number of programs. If used, shared libraries only need to be loaded once into real memory, after which every program can use them.
- TCP/IP support: This means that low-level networking protocols are handled in the kernel itself.

***Instructor notes:***

**Purpose** — Introduce the Linux kernel

**Details** —

**Additional information** —

**Transition statement** — Let's see what the kernel looks like when it's on your system.

# Kernel components

- Kernel image
  - */boot/vmlinuz-version*
  - Contains required core code such as startup, memory manager, scheduler
- Kernel modules
  - */lib/modules/version/\**
  - Contains optional code supporting specific features such as filesystems, drivers, network protocols, security
  - Loaded when necessary
- Initramfs
  - */boot/initramfs-version.img*
  - Contains modules required to access the root filesystem
  - Loaded alongside the kernel image by the boot loader

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-3. Kernel components

LX158.0

## Notes:

The kernel as distributed by Linus Torvalds is a 75+ MB tar.bz2 file containing more than 40.000 files and over 11 million lines of code. But you don't have to deal with this file at all. Your distributor has downloaded and compiled this source code, and it is placed on your system as part of the installation process.

The whole kernel, as placed on your system, has a few components:

- The first component is the compressed Linux kernel image itself. It is generally called *vmlinuz-version*, and placed in the */boot* directory.

This kernel image is loaded into memory by the boot loader and contains core services such as startup code, the memory manager and the scheduler.

- To the greatest extent possible, kernel components are compiled as modules. This allows them to be loaded and unloaded as and when required, greatly reducing memory requirements for the kernel.

Kernel modules are normally stored in */lib/modules/version*.

Hundreds of modules exist. Most of these contain code to handle specific filesystems, device drivers, network protocols, or specific security features.

- The /lib/modules directory can only be accessed by the core kernel image once the root filesystem has been mounted. But the root filesystem normally cannot be mounted until certain required modules are loaded.

To solve this chicken-and-egg problem, certain modules are copied from the /lib/modules directory into an "initramfs" file. This file is loaded into memory alongside the core kernel image by the boot loader. The kernel then accesses this file, loads the relevant modules from this file and thus gains access to the code required to mount the root filesystem.

The kernel image itself, the /lib/modules/*version* directory tree and the corresponding initramfs are contained in a single RPM, and can be installed/upgraded/uninstalled as a single unit.

***Instructor notes:***

**Purpose** — Introduce the kernel components

**Details** —

**Additional information** —

**Transition statement** — We need to take a closer look at these modules.

# Loading modules

- Modules can be loaded manually
  - **insmod**: Loads a single module
  - **lsmod**: Lists all loaded modules
  - **rmmmod**: Unloads a single module
  - **depmod**: Determines module dependencies, store in **modules.dep**
  - **modprobe**: Loads a module + dependencies
  - **modinfo**: Display information about a module
- Modules are loaded dynamically when needed
  - Kernel uses `/lib/modules/version/modules.*map` to determine module name
  - Kernel/modprobe use information in `/etc/modprobe.d/*` to determine correct methods and settings

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-4. Loading modules

LX158.0

## Notes:

One of the main advantages of having a modularized kernel is that the modules can be loaded and unloaded as and when required. This means that only the modules that are actually needed, need to be loaded. And modules can be unloaded when they are no longer needed. This frees up memory for applications.

Several commands allow you to manage the loading/unloading of modules manually:

- **insmod** loads a single module
- **lsmod** displays a list of loaded modules
- **rmmmod** unloads a single module
- Very often modules are dependent on the presence of other modules. The **depmod** command generates dependency information and stores this in the file `/lib/modules/version/modules.dep`.
- **modprobe** is a wrapper program around **insmod** that takes into account the dependency information in `modules.dep`. It also takes into account the module configuration that is contained in `/etc/modprobe.conf` and `/etc/modprobe.d/*`.

- **modinfo** allows you to view information about a specific module.

Most distributions are also configured to load modules automatically when needed. The exact process is a bit too complicated to cover here, but in essence this process works as follows:

1. If the kernel detects a new hardware device, such as a USB device, it will retrieve the Vendor and Product ID via standard USB calls.
2. It will then lookup the required module based on the information contained in the `/lib/modules/version/modules.usbmap` file. Similar files exist for PCI, ISA, firewire and so forth.
3. The kernel will then invoke **modprobe** to load that particular module.
4. **modprobe** will take into account the dependency information in `modules.dep`, and any specific configuration options for this module in `/etc/modprobe.conf` and `/etc/modprobe.d/*`
5. Once loaded, the module will run its initialization code, which will activate the device.

**Instructor notes:**

**Purpose** — Discuss loading and unloading of modules

**Details** —

**Additional information** —

**Transition statement** — Okay, that's how the kernel is structured. Let's look at a few services the kernel offers.

# Kernel services

- Process scheduler, memory management, filesystems, networking, security and other core services
  - [\(Covered in other units\)](#)
- Virtual filesystems: udev/devtmpfs, tmpfs, procfs, sysfs, ...
  - [Access to kernel data structures/information](#)
- D-BUS communication channel
  - [Light-weight inter-process communications](#)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-5. Kernel services

LX158.0

## Notes:

The Linux kernel has to perform a large number of tasks. Important tasks such as process scheduling, memory management, filesystems, networking, security and other core services all have their own unit in this or other Linux courses.

In addition to these core services, there are a few other services that are important, but not important enough for their own unit. They are covered in this unit.

Most of these additional services are offered in the shape of a virtual filesystem. A virtual filesystem is something that looks like a normal filesystem once it's mounted, but is really an interface into the kernel datastructures.

In addition to this, the kernel also has support for something called D-Bus. This is a lightweight inter-process communication channel which can be used for applications to communicate amongst themselves, or with the kernel, using standardized messages.

***Instructor notes:***

**Purpose** — Introduce a few kernel services

**Details** —

**Additional information** — Make sure to stress that the major services all have their own units. It's just a few minor things that need to be covered

**Transition statement** — Let's look at the first virtual filesystem.

## udev/devtmpfs

- **udevd:** Device manager for Linux
  - Separate daemon in user space
  - Listens for device events ("uevents") using netlink (socket interface to kernel)
  - Configures /dev directory based on information in /etc/udev/\* and /lib/udev/\*
    - Proper device name
    - Proper type (block versus char), major and minor number
  - Sends broadcast message by way of D-Bus to all applications
    - Various **applications** can listen to D-Bus for specific device events to start, for instance, a camera manager application
  - Also populates /dev with standard virtual devices (such as /dev/null)
- **devtmpfs:** Kernel internal filesystem mounted on /dev
- Monitor **udev** events with **udevadm monitor**
- Retrieve **udev** information with **udevadm info**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-6. udev/devtmpfs

LX158.0

### Notes:

Traditionally, the /dev directory in a UNIX system is filled with 'device files' or 'device nodes'. These are special files that act as an interface to hardware devices. Since Linux may support thousands of different devices, possibly all simultaneously, this meant that the /dev directory was filled with thousands of entries. Each of these entries need an inode, and thus waste a lot of disk space. Clearly this situation was unmaintainable in the long run.

After several intermediate solutions, Linux today uses the **udev/devtmpfs** method of managing devices and their entries in /dev.

The most common configuration is where the **udevd** daemon is started as part of the startup sequence of the system. **udevd** opens a netlink communications channel to the kernel, and the kernel will forward any device events (such as a physical addition/removal), called **uevents** to the udevd.

**udevd** will then use the configuration information in /etc/udev/ and /lib/udev/ to find out the proper name, type, major and minor number for the device that's been added, and will create the corresponding device node in /dev.

Once the device node in /dev has been created, **udevd** will send out a message on the "system" D-Bus. Any application that has registered to receive these messages will receive it, and may act upon it. This is used, for instance, to start a file browser when a disk device (for instance a USB disk) is added to the system.

**udevd** uses a persistent naming scheme. This means that once a device has been added to the system, the name it ended up with is retained. Once the device is added again, it will get the same name as previously.

**udevd** also populates the /dev directory with standard, virtual device nodes such as /dev/null, /dev/zero, /dev/random and so forth.

Udev events can be monitored with the command **udevadm monitor**. **udevadm** can also be used to retrieve information from the udevd database (using the **info** keyword), to test udev behavior (with the **test** keyword) and a number of other things. See the manual page of **udevadm** for details.

Alongside **udev** you also need **devtmpfs**. This is a kernel internal filesystem, similar to a RAM disk, that is supposed to be mounted on /dev.

**Instructor notes:**

**Purpose** — Discuss dynamic device management in Linux

**Details —**

**Additional information** — You might want to ask the students whether they are familiar with the contents of the /dev directory, and talk briefly about that if they're not. This information was contained in the LX02 course, but not everyone might have attended that course, or remembered the information.

**Transition statement** — Let's look at another virtual filesystem, procfs.

## procfs filesystem

- Virtual filesystem mounted on /proc
- Read access to kernel datastructures
  - Kernel info such as version, cpu info, memory info
  - Process information
  - Device information (deprecated)
- Kernel configuration via /proc/sys datastructure
  - For instance `echo 1 > /proc/sys/net/ipv4/ip_forward`

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-7. procfs filesystem

LX158.0

### **Notes:**

The procfs filesystem is a virtual filesystem that's typically mounted on /proc. It provides access to kernel datastructures.

Initially the information in /proc was limited to kernel information (version, cpu information and so forth) and process information (PID, cmdline and so forth). But since there was no consistent way for device drivers to make available information, device driver authors started to use the /proc filesystem too.

This lead to a mass of unstructured device information in /proc. Kernel developers are now in the process of moving device driver information to a more structured approach in /sys. /sys will be covered in the next visual.

An important directory within /proc is /proc/sys. The virtual files contained in this directory tree are all read-write files, and can be used to change the kernels behavior. As a very simple example, writing a "1" into the virtual file /proc/sys/net/ipv4/ip\_forward turns on IPv4 forwarding.

We will look at kernel configuration through the /proc/sys mechanism later.

**Instructor notes:**

**Purpose** — Discuss /proc.

**Details** —

**Additional information** —

**Transition statement** — Let's look at /sys.

## sysfs filesystem

- Virtual filesystem mounted on /sys
- Successor of /proc specific for device information
- Much better structured than /proc
  - Uses **device tree** and **device identifiers** to locate device information
- Integrates device manipulation such as sleep/wake/configure
  - Greatly reduces the need for **ioctl**s

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-8. sysfs filesystem

LX158.0

### Notes:

As said, sysfs, mounted on the /sys, is the replacement of procfs and /proc for device information. The aim of sysfs is to create a far more structured environment for device driver writers to publish their information and interfaces. To that extent, they are using a device tree and device identifiers to locate the device you are looking for, instead of symbolic names and a flat directory structure.

Device driver writers are also encouraged to use the sysfs filesystem to publish read-write virtual files. These files can then be used to manipulate the device, for instance putting it to sleep, waking it up, and to configure it. This greatly reduces the need for **ioctl**s (system calls for I/O control<sup>1</sup>).

<sup>1</sup> An example of an ioctl is the "eject" command.

**Instructor notes:**

**Purpose** — Discuss sysfs.

**Details** —

**Additional information** — Stress that in the normal operations of the system, the /proc and /sys directories are only rarely accessed directly by the system administrator. Instead, the system administrator would use tools such as **uname**, **ps**, **Ispci** and such to access this information.

**Transition statement** — Let's look at the last kernel service, D-Bus.

## D-Bus

- Simple Inter-Process Communication (IPC) system
- One daemon (**dbus-daemon**) per channel
  - One "system" channel
  - One "user" channel per user logged in (started by KDE/GNOME)
  - Daemon keeps track of services offered by applications
  - Daemon forwards messages according to service requested
- Monitor D-Bus communications with **dbus-monitor**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-9. D-Bus

LX158.0

### Notes:

D-Bus is not a kernel service per se, but is important in the operation of the kernel. That's why we cover it here.

D-Bus is a simple, lightweight Inter-Process Communication system: it enables the sending of messages between various applications on a Linux system. One of those applications is the Linux kernel itself.

D-Bus is implemented by way of the **dbus-daemon**. Several instances of this daemon can be running:

- There will always be one daemon that handles the "system" bus. This system bus is used for system-wide communication.
- Depending on the configuration, there may also be one daemon that handles a "session" or "user" bus. This allows, for instance, KDE or GNOME desktop applications to communicate with each other.

Applications that want to use the D-Bus mechanism open a socket to the D-Bus daemon. They register themselves and identify the types of messages they want to listen to. The

---

daemon keeps track of the applications that connected, and will forward any relevant messages.

You can monitor D-Bus communications with the **dbus-monitor** command.

**Instructor notes:**

**Purpose** — Discuss D-Bus

**Details** —

**Additional information** — Again, it is rare that you need to be involved with D-Bus manually as a system administrator.

**Transition statement** — Okay, we've seen a few kernel services. The last topic is to configure the kernel.

# Configuring the kernel

- Certain kernel configuration options can be set after kernel compile
- Some possibilities for configuration:
  - When kernel image boots
    - Amount of memory used
    - Root file system to mount
  - When modules are loaded
    - Hardware present, hardware settings
  - Through /proc/sys mechanism
    - Software functions, such as IP forwarding

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-10. Configuring the kernel

LX158.0

## Notes:

To a very large extent, kernel configuration is done during the compilation process. This means that major choices, such as which bits to compile as modules or into the core image, and various default settings, can not be changed by the system administrator. Unless the system administrator compiles his own kernel of course. This is covered in an appendix.

There are some settings in the kernel that can be configured by the system administrator though. Depending on the location of the piece of code that is to be configured, and how flexible this bit of code is, there may be three locations where you need to make these changes.

***Instructor notes:***

**Purpose** — Introduce kernel configuration.

**Details** —

**Additional information** —

**Transition statement** — Let's look at the first method, which is used for the kernel image itself.

## Configuring the kernel at boot time

- Done by adding arguments to the kernel line in the boot manager
  - GRUB: add to `kernel` line or enter manually
- Examples:
 

– <code>mem=1024M</code>	Identify amount of memory to kernel
– <code>root=/dev/hda3</code>	Identify root FS to mount
– <code>Ro</code>	Mount root FS readonly
– <code>init=/sbin/init</code>	First program to run
– <code>initrd=/initrd-2.6.9-27.EL.img</code>	Use the initial RAM disk image called initrd-2.6.9-27.EL.img
- For more information:
  - Manual page for **bootparam**
  - `/usr/src/linux/Documentation/kernel-parameters.txt` (SLES)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-11. Configuring the kernel at boot time

LX158.0

### Notes:

The first way of configuring the kernel is when the kernel itself boots. Just like any other program, the kernel accepts arguments to its “command line.” However, since the command line of the kernel cannot be accessed directly, you need to work with your boot loader to achieve this.

In GRUB, as an example, you can either modify the "kernel" stanza in `/boot/grub/grub.conf` file permanently, or add the parameter while the GRUB boot menu is being shown.

An `/etc/grub/grub.conf` stanza for the kernel then looks like this:

```
title new
root (hd0,0)
kernel /vmlinuz-2.6.9-27.EL ro root=/dev/sda5 rhgb quiet mem=128M
initrd /initrd-2.6.9-27.EL.img
```

There are several parameters that can be useful here. Some of these, and their usage, are:

**Table 1: Common kernel parameters**

Parameter	Description
<b>mem=&lt;memory size&gt;M</b>	Disables the auto-detection of memory and limits the amount of memory used to the amount indicated. Note that the capital M (for Megabytes) is required. Failing to use the capital M causes the kernel to use the number of bytes specified, which immediately leads to a kernel panic if it is too small. <b>Note:</b> This parameter can be used, among other things, in the process of sizing a system: trying out the amount of memory that a production system can get by with.
<b>root=&lt;root device&gt;</b>	This identifies the root file system to be mounted. This is normally set up correctly and should not be changed. It can be used, however, if you want to boot your system using a Linux kernel on a floppy disk.
<b>ro</b>	This value ensures that the root file system is mounted read-only. That makes it possible for the bootup scripts to perform an <code>fsck</code> on the file system. The bootup scripts then do a remount to mount the file system read-write.
<b>init=&lt;program to start&gt;</b>	This identifies the program that should be started first, as soon as the kernel finishes booting. Normally, this is <code>/sbin/init</code> , but if that program is corrupt, or <code>/etc/inittab</code> is corrupt, you can also specify, for instance, <code>init=/bin/bash</code> . This gives you a <code>bash</code> prompt immediately and allows you to do recovery of <code>init</code> and <code>/etc/inittab</code> . Note, however, that specifying <code>init=/bin/bash</code> performs no startup scripts whatsoever. This means that only the root file system is mounted, <b>read-only</b> . You will have to do a <code>remount</code> to read-write of the root file system, and possibly <code>mount</code> other file systems as well, in order to do something useful.
<b>initrd=&lt;Initial RAM disk&gt;</b>	Specifies the Initial RAM Disk to use.

Notice that all of the parameters are for core kernel services such as memory management.

***Instructor notes:***

**Purpose** — Discuss configuring the kernel image.

**Details** —

**Additional information** —

**Transition statement** — Let's look at configuring modules.

## Configuring modules at load time

- Specify in /etc/modprobe.conf or /etc/modprobe.d/<file>

```
# cat /etc/modprobe.conf
alias eth0 eepro100
alias eth1 eepro100
options eth0 irq=9
options eth1 irq=10
```

- alias** identifies the module which implements a device
- Options are specific for each module
  - Use **modinfo** to obtain specific information
- Pre-install, install, and post-install execute scripts when loading a module
- Pre-remove, remove, and post-remove execute scripts when unloading a module

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-12. Configuring modules at load time

LX158.0

### Notes:

When modules are checked for dependencies with **depmod** and when they are loaded with **modprobe**, the options from /etc/modules.conf and the files in /etc/modprobe.d/ are being read. There are four things that can be specified here:

- The *alias* specifies the name of the module that is to be loaded to support a specific device. In the example shown in the visual, if someone wants to use the eth0 device, the kernel automatically loads the eepro100 module, which contains the kernel code for that device.
- The *options* line specifies the specific options to be passed to the module when it is being loaded. This can be very useful if you have two or more identical Ethernet cards as in the example. The options line is then used to distinguish them from each other.

Other uses of the *options* line exist as well. As an example, you can use them to force Ethernet cards into *full-duplex mode*, or force Token-Ring cards to a *ringspeed* of 16 Mbps.

For specific information about the options that a module supports you will need to run `modinfo -p <modulename>` or dig into the source. (Most modules have a list of possible options right at the start of the source code.)

- The pre-install, install, and post-install lines allow you to specify scripts that are to be started when loading a module.
- The pre-remove, remove, and post-remove lines allow you to specify scripts that are to be started when unloading a module.

Although most distributions do not use it, it is also possible to put an *include* statement in the modules.conf file, which ensures that other files (typically located in /etc/modules.d) are included. This helps keep your modules.conf file clean.

***Instructor notes:***

**Purpose** — Discuss configuring modules

**Details** —

**Additional information** —

**Transition statement** — Certain bits of kernel code also allow dynamic configuration. Let's look.

## Configuring the running kernel

- Certain kernel settings can be changed while running
- All these settings have an entry somewhere in /proc/sys
- Example: Setting up IP-forwarding

```
# cat /proc/sys/net/ipv4/ip_forward
0
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

- **sysctl** command gives easy interface to this:

```
# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
# sysctl -w net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

- The **sysctl -a** option prints out all current settings, the **-p** option reads settings from /etc/sysctl.conf

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-13. Configuring the running kernel

LX158.0

### Notes:

Several kernel parameters can be changed at run time. An example of this is IP forwarding, which can be turned on and off while the system is running. All these changeable parameters have a virtual file representation in /proc/sys.

To list the current setting, simply list the file to the screen with the **cat** command. To change a setting, simply **echo** the new setting to the file.

These changes, however, are not persistent. Because of this, the **sysctl** utility has been created which can do this for you: it allows you to store all settings in a file, /etc/sysctl.conf or /etc/sysconfig/sysctl. As part of the bootup scripts, the **sysctl -p** command is executed. This reads all settings from /etc/sysctl.conf or /etc/sysconfig/sysctl and applies them.

With the **sysctl** command you can also list and change settings manually, but this is not often used.

**Instructor notes:**

**Purpose** — Discuss the /proc/sys filesystem and the **sysctl** tool

**Details** —

**Additional information** —

**Transition statement** — One last thing we need to cover: Upgrading the kernel.

# Upgrading a kernel

- Do NOT use **rpm -U**
  - This will delete the */lib/modules/version* tree of the current kernel
- Correct procedure:
  - **rpm -i** to install new kernel alongside current kernel
  - Modify/check boot loader configuration
  - Reboot into new kernel
  - Test
  - **rpm -e** of the old kernel image (or leave in place)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-14. Upgrading a kernel

LX158.0

## Notes:

Hundreds of people make contributions to the kernel: Fixing bugs, adding new functionality, tuning performance and so forth. Using the "official" kernel of Linus Torvalds, these changes will eventually filter through to the distributors, and the distributor then makes new kernels available for their users to use. These kernels, just like all other applications on a Linux system, are distributed in RPM format.

Although it's technically possible, it is NOT a good idea to upgrade a running kernel with the **rpm -U** command. The reason for this is that an upgrade also deletes all files associated with the current version of the product. In this case, the */lib/modules/version* directory tree.

So if your kernel needs to load a module after the upgrade, the module will not be available. This may eventually lead to a system crash.

The proper way of upgrading a kernel is to install the new kernel first with the **rpm -i** command. All files in a kernel RPM are always tagged with the version number somehow, so there should not be a conflict with the existing kernel files.

Once the new kernel is installed, you check/update the boot loader configuration so that the new kernel is actually available in the boot menu. Most distributions include post-install scripts in the kernel RPM package that does this for you, but it makes sense to check anyway.

You then reboot the system into the new kernel and verify that everything works.

Once you are satisfied that your new kernel performs properly, you can delete the old kernel with the **rpm -e** command. But you may also leave the old kernel in place.

## 11.2. Device management

### Instructor topic introduction

**What students will do —**

**How students will do it —**

**What students will learn —**

**How this will help students on their job —**

# UNIX device management

- Traditionally all UNIX devices have a representation in /dev
  - Character devices: serial, byte-by-byte access
  - Block devices: random, block-based access
- Today's Linux implementation
  - **devtmpfs** mounted on /dev
  - **udev** manages /dev entries dynamically
  - Not all devices represented in /dev, for instance no network adapters
  - Device information located in **/proc** (deprecated) and **/sys**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-15. UNIX device management

LX158.0

## Notes:

Traditionally, all devices on a UNIX system were represented by an entry in /dev. By reading from, and writing to these device representations as if they were normal files, you could manipulate their contents. And for the few things that you could do with a device that could not be done, or represented, by reading from or writing to the device, there were special **ioctl**s (I/O Controls) defined.

Two groups of devices existed: character and block devices.

Character devices are devices that can be read from, or written to, byte-for-byte. Access is also serial: you were not able to seek() in the device.

Block devices are devices that are typically read from, or written to, block-for-block, where the block size is typically 512 bytes or a multiple of that number. Access is also random: you are allowed to seek() back and forth in the device.

These device representations and classification were very limited, as modern systems support a lot of devices where reading from them, or writing to them, doesn't really make sense. Take for instance a network adapter. What would you read from them, or write to

them? Just the data that you want to send is not enough; you also need to include IP and MAC addresses for instance. For this reason, not all devices that are actually in a system, are represented by an entry in /dev.

Another problem is that a typical modern UNIX system supports thousands of different types of devices, and may support hundreds of instances of the same device. Having to create /dev entries for all these devices is a massive task in a static environment (filesystem), and takes up valuable disk space.

For this reason device management in Linux has been made far more advanced and dynamic:

- A special RAM filesystem, **devtmpfs** or **tmpfs** is mounted on /dev.
- A special daemon, **udevd**, is running and will keep track of any devices that are added/removed to/from the system (through a communications channel with the kernel). In reaction to this, it will create or remove the corresponding device representation in /dev.
- Detailed device information can be gotten from **/proc** (deprecated) and **/sys**. Both of these are virtual filesystems that offer direct access to various datastructures in the kernel.

The net result of this is that the /dev directory on a modern Linux system only contains representations of devices that are actually physically present in the system (plus a few virtual devices), and for which it makes sense to have a device representation in /dev because you can read from them or write to them.

## **Instructor notes:**

**Purpose** — Introduce device management

**Details** —

**Additional information** — It makes sense to do an **ls /dev** of your system at this stage. If you have a USB stick or something similar, insert it and run **ls /dev** again, so that people see the changes.

**Transition statement** — Let's first talk about character devices.

# Character devices

- A *character device* is any device that does not allow random access (seeks).
- Examples:
  - Console (keyboard, mouse)
  - Serial terminals
  - Printers
  - Sound card
  - Random number generator
  - Null device
  - Many others!

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-16. Character devices

LX158.0

## Notes:

Character devices are devices that are written to, and read from, byte for byte. They also do not allow you to seek() backwards and forwards in the file representation: They can only be read/written sequentially.

A lot of devices in Linux are character devices:

- The console itself
- Any serial terminals that are attached to the system. (A serial terminal is a combination of monitor and keyboard, which is attached through a serial cable to a serial port.)
- Printers
- Sound cards
- The random number generator
- The null device, which is typically used to discard unwanted data
- Most other devices which connect to the machine: video cameras, network adapters, microphones, and so on

***Instructor notes:***

**Purpose** — Introduce character devices.

**Details** —

**Additional information** —

**Transition statement** — Almost all character devices have a representation in /dev. Let's look at that.

# Character device naming

- All character devices have a representation in /dev

```
# ls -l /dev
crw----- 1 root root      5,  1 Oct 18 2002 /dev/console
crw----- 1 root root     14,  3 Oct 18 2002 /dev/dsp
crw----- 1 root lp       6,  0 Oct 18 2002 /dev/lp0
crw-rw-rw- 1 root root     1,  3 Oct 18 2002 /dev/null
crw----- 1 root root    10,  1 Oct 18 2002 /dev/psaux
crw-r--r-- 1 root root     1,  8 Oct 18 2002 /dev/random
crw-w---- 1 root root     4,  0 Oct 18 2002 /dev/tty0
crw-rw---- 1 root uucp    4,64 Oct 18 2002 /dev/ttys0
crw-rw---- 1 root root     1,  9 Oct 18 2002 /dev/urandom
crw-rw-rw- 1 root root     1,  5 Oct 18 2002 /dev/zero
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-17. Character device naming

LX158.0

## Notes:

### Introduction

All character devices have a representation in /dev. As you can see, the permission fields as shown by **ls -l** all start with a "c", which indicates a character device.

The fifth and sixth columns represent the MAJOR and MINOR device number. This is the way user-space programs refer to hardware devices that they wish to use.

A list of the major/minor device numbers can also be found in the kernel tree as Documentation/devices.txt.

***Instructor notes:***

**Purpose** — Review the notation of character devices in /dev.

**Details** —

**Additional information** —

**Transition statement** — Let's first look at four virtual devices.

# Virtual character devices

- Null devices
  - `/dev/null`: Bit bucket; used for unwanted output
  - `/dev/zero`: Infinite supply of binary zeroes
- Random devices
  - `/dev/random`: Entropy pool: Blocks if empty
  - `/dev/urandom`: Entropy pool: Switches to pseudo-random if empty

Example: Creating a empty file (32 MB)

```
# dd if=/dev/zero of=/tmp/swapfile bs=1M count=32
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-18. Virtual character devices

LX158.0

## Notes:

On any Linux system, there are four virtual character devices. These devices have a representation in `/dev` but do not have matching hardware.

These devices are:

- `/dev/null` Used as the “bit bucket”, to discard unwanted output of a command or script.

Example: `$ find / 2> /dev/null`

- `/dev/zero` Used as an infinite supply of binary zeros: If you do a `cat /dev/zero`, the output will be all ASCII character 0's. Unfortunately, this is an undisplayable character, so you need to do `hexdump -v /dev/zero` to see anything at all. `/dev/zero` is typically used to create large, empty files. The following command, for example, creates a 1M file:

**`$ dd if=/dev/zero of=bigfile bs=1M count=1`**

- `/dev/random` Truly random numbers are really important in the field of computer security. It is really hard to generate truly random numbers on a “deterministic device” such as a computer.

In the past, programs requiring random numbers have always used pseudo-random numbers, and each program had its own implementation to generate these. This has caused a lot of security problems. To solve this, Linux implements the /dev/random device, which holds a large number of random numbers (called the *entropy pool*).

These random numbers are truly random, and are derived from random events in the outside world, such as mouse movements. It is for instance really illustrative to do: **\$ hexdump /dev/random** And then move the mouse.

- /dev/urandom The problem with /dev/random is that the entropy pool is generally not overly large: after a few hundred to thousand random characters, the entropy pool is empty. If a program requires more than this amount of randomness, it should have to wait before someone moves the mouse. Obviously, mouse events are rare on a heavily loaded server in a computer room.

To solve this problem, /dev/urandom was introduced. This device generates truly random numbers as long as the entropy pool is not empty, but starts generating pseudo-random numbers (based on the earlier random numbers) as soon as the entropy pool is empty.

***Instructor notes:***

**Purpose** — Introduce a few virtual character devices.

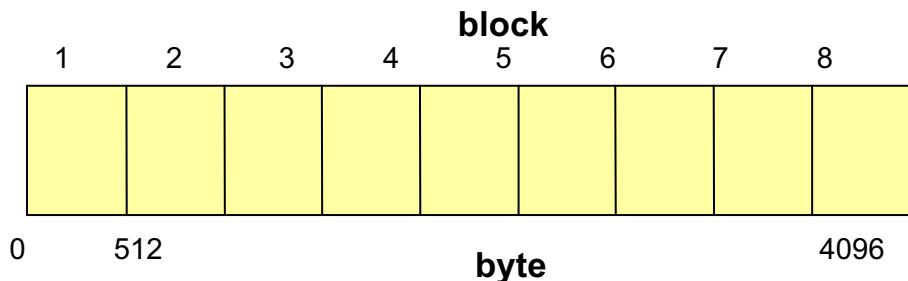
**Details** —

**Additional information** —

**Transition statement** — Okay, let's also look at block devices.

## Block devices

- A *block device* is any device that allows random access ("seeks") and that is divided into "blocks" of a given size.



- Typical block devices:
  - Hard disks (and partitions)
  - USB flash drives
  - Virtual block devices (RAID and LVM)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-19. Block devices

LX158.0

### Notes:

A block device in the Linux world is any device which allows “random” access. This means that it is possible to write something to location n, and then go backwards to read something from location m. In other words: a block device is any device that supports the “seek” command. Typical examples are hard disks, hard disk partitions, floppy disks, USB flash drives, RAM disks, LVM volumes, RAID volumes, and files.

A block device can be used for different things, for example to hold a file system, as a swap space, or “raw,” for instance, using tar. But as you will see in this discussion, it can also be used for LVM and/or RAID.

***Instructor notes:***

**Purpose** — Introduce block devices.

**Details** —

**Additional information** —

**Transition statement** — Let's first look at block device naming.

## Block device naming

- All block devices have a special file representation in /dev

```
# ls -l /dev
total 1
brw-rw---- 1 root floppy 2, 0 Apr  4 03:37 fd0
brw-r----- 1 root      disk 3, 0 Apr  4 03:37 sda
brw-r----- 1 root      disk 3, 1 Apr  4 03:37 sda1
brw-r----- 1 root      disk 3, 2 Apr  4 03:37 sda2
brw-r----- 1 root      disk 3, 3 Apr  4 03:37 sda3
brw-rw---- 1 root      disk 22,0 Apr  4 03:37 sdc
brw-r----- 1 root      disk 7, 0 Apr  4 03:37 loop0
brw-r----- 1 root      disk 9, 0 Apr  4 03:37 md0
. . .
```

- fd0, fd1, ...: floppy disk (max 8)
- hda, hdb, ...: IDE hard disk (max 8)
- sda, sdb, ...: SCSI hard disk (max 128)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-20. Block device naming

LX158.0

### Notes:

Block devices have a representation in /dev as well. You can identify a block device since the permissions field starts with "b".

A few common block devices are:

- fd0, fd1, ...: Floppy disks
- hda, hdb, ...: IDE hard disks
- sda, sdb, ...: SCSI hard disks

***Instructor notes:***

**Purpose —** Show block devices in /dev

**Details —**

**Additional information —**

**Transition statement —** Let's also look at a special block device, the loop device.

## The loop device

- The *loop device* is used to access files as block devices.
- Linux supports a maximum of 16 loop devices by default.

```
# mount -o loop bootnet.img /mnt/floppy
# mount -o loop,ro cdimage.iso /media/cdrom
```

- Loop devices also support encryption and offsets.
  - Use **losetup** to initially set up the loop device.
  - Can then **mount** and **umount** the device transparently

```
# dd if=/dev/zero of=secrets.enc bs=1M count=32
# losetup -e blowfish /dev/loop0 -o 1024 secrets.enc
... asks for password to be used as encryption key ...
# mke2fs /dev/loop0
# losetup -d /dev/loop0
# mount -o loop,encryption=blowfish secrets.enc /mnt/secrets
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-21. The loop device

LX158.0

### Notes:

Files are block devices too. The most obvious example of this is a tar file, which is essentially an image of a tape. In most cases, a file can be specified where a block device is typically used, and vice versa.

There is one exception to this though: A file containing a file system cannot be mounted directly. For this to succeed, the use of a special *loop device* is needed. Linux supports a maximum of 16 of these devices by default, but this can be changed with a kernel recompile.

Linux automatically invoke one of these devices if the **-o loop** option is specified with the **mount** command, as shown in the visual. This allows you to mount, for instance, floppy disk or ISO images.

The **mount -o loop** command actually invokes the **losetup** command to couple a file to a /dev/loop device. You can also invoke **losetup** manually, and that gives you the opportunity to enable encryption on the device as well. Note that this only works when the **cryptoloop** kernel module is loaded (**modprobe cryptoloop**).

---

You can also specify an offset into the file. This means that the block device doesn't start at byte 0 of the file, but somewhere further down the file. This can be useful, for instance, if you have a raw image of a full hard disk, and you want to bypass the master boot record, partition table and so forth, so that you can access the filesystem in a partition.

The encryption methods that are available are dependent on the kernel version and kernel compilation options, though, and in practice distributions that have a 2.6 or newer kernel have a good set of encryption methods available.

***Instructor notes:***

**Purpose** — Discuss the loop device

**Details** —

**Additional information** —

**Transition statement** — Okay, we have seen how devices are detected, and we looked at a few specific devices. Now let's see how we can retrieve device information in general.

## **lspci command output**

```
# lspci
00:00.0 Host bridge: Intel 82855PM Processor to I/O Controller
00:01.0 PCI bridge: Intel 82855PM Processor to AGP Controller
00:1e.0 PCI bridge: Intel 82801 Mobile PCI Bridge
00:1f.0 ISA bridge: Intel 82801DBM LPC Interface Bridge
00:1f.1 IDE interface: Intel 82801DBM IDE Controller
00:1f.3 SMBus: Intel 82801DB/DBL/DBM SMBus Controller
00:1f.5 Multimedia audio controller: Intel 82801DB/DBL/DBM AC'97 Audio Controller
01:00.0 VGA compatible controller: ATI Technologies Inc M10 NT [FireGL Mobility T2]
```

>>>>> Plug in a PCI Modem card here <<<<<

```
# lspci
00:00.0 Host bridge: Intel 82855PM Processor to I/O Controller
00:01.0 PCI bridge: Intel 82855PM Processor to AGP Controller
00:1e.0 PCI bridge: Intel 82801 Mobile PCI Bridge
00:1f.0 ISA bridge: Intel 82801DBM LPC Interface Bridge
00:1f.1 IDE interface: Intel 82801DBM IDE Controller
00:1f.3 SMBus: Intel 82801DB/DBL/DBM SMBus Controller
00:1f.5 Multimedia audio controller: Intel 82801DB/DBL/DBM AC'97 Audio Controller
01:00.0 VGA compatible controller: ATI Technologies Inc M10 NT [FireGL Mobility T2]
00:1f.6 Modem: Intel 82801DB/DBL/DBM AC'97 Modem Controller
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-22. lspci command output

LX158.0

### **Notes:**

#### **Introduction**

The command **lspci** is a utility that can be used to display information about all of the PCI busses in the system and all devices connected to them. This output of the **lspci** command is shown above, both before a device is plugged in and after.

***Instructor notes:***

**Purpose** — Show the output of the **lspci** command.

**Details** —

**Additional information** —

**Transition statement** — Let's look at **lsusb** as well.

## lsusb command output

```
# lsusb
Bus 005 Device 001: ID 0000:0000
Bus 004 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 001: ID 0000:0000

>>>>>> Plug in a USB Mouse here <<<<<<

# lsusb
Bus 005 Device 001: ID 0000:0000
Bus 004 Device 001: ID 0000:0000
Bus 003 Device 002: ID 046d:c50e Logitech Cordless Mouse Receiver
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 001: ID 0000:0000
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-23. lsusb command output

LX158.0

### Notes:

#### Introduction

This output of the **lsusb** command is shown above, both before a device is plugged in and after.

Some devices, depending upon how “smart” they are, might automatically mount. In this example, a mass storage device is plugged into a USB port. During the next moment, the USB device is discovered and mounted, after which the **df** command is run and shows:

**# df**

File system	1K-blocks	Used	Available	Use%	Mounted on
/dev/hda5	19840892	17745864	107088895%	/	
/dev/hda3	93327	17921	7058721%	/boot	
/dev/shm	1037732	0	10377320%	/dev/shm	
/dev/sda1	999552	30720	9688324%	/media/CRUZER-1GB	

***Instructor notes:***

**Purpose** — Discuss hot-pluggable USB devices.

**Details** — Note that not all USB devices will actually be discovered and/or mounted automatically.

**Additional information —**

**Transition statement** — If the information from `lspci` and `lsusb` is not enough, we can also run the `udevadm info` command.

# **udevadm info command output**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-24. udevadm info command output

---

LX158.0

### **Notes:**

The **udevadm info** command lists all the information that the **udevd** daemon knows about a device. This is taken from the /proc and /sys virtual filesystems.

As you can see, the information is extremely detailed, up to and including things like serial numbers, firmware levels and very detailed capabilities.

***Instructor notes:***

**Purpose** — Discuss the **udevadm info** command

**Details** —

**Additional information** —

**Transition statement** — That's it. Checkpoint time!

## Checkpoint (1 of 2)

1. The Linux kernel is:
  - a. a monolithic kernel
  - b. a monolithic, modularized kernel
  - c. a microkernel
  - d. a modularized microkernel
2. What information can be found in the /proc virtual filesystem?
3. Name three ways to configure the kernel.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-25. Checkpoint (1 of 2)

LX158.0

### Notes:

**Instructor notes:**

**Purpose** — Discuss kernel upgrades

**Details** —

## Checkpoint solutions (1 of 2)

---

1. The Linux kernel is:

- a. a monolithic kernel
- b. a monolithic, modularized kernel
- c. a microkernel
- d. a modularized microkernel

The answer is a monolithic, modularized kernel.

2. What information can be found in the /proc virtual filesystem?

The answer is kernel, process and device information.

3. Name three ways to configure the kernel.

The answers are with a command line parameter (via the bootloader), when loading a module (via /etc/modprobe.conf), and using the /proc/sys virtual filesystem (through sysctl).

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information** —

**Transition statement** — Checkpoint time.

## Checkpoint (2 of 2)

4. True or false: A character device allows random seeks.
5. What is the difference between /dev/random and /dev/urandom?
6. Name a few commands that allow you to retrieve device information.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-26. Checkpoint (2 of 2)

LX158.0

### Notes:

**Instructor notes:**

Purpose —

Details —

## Checkpoint solutions (2 of 2)

---

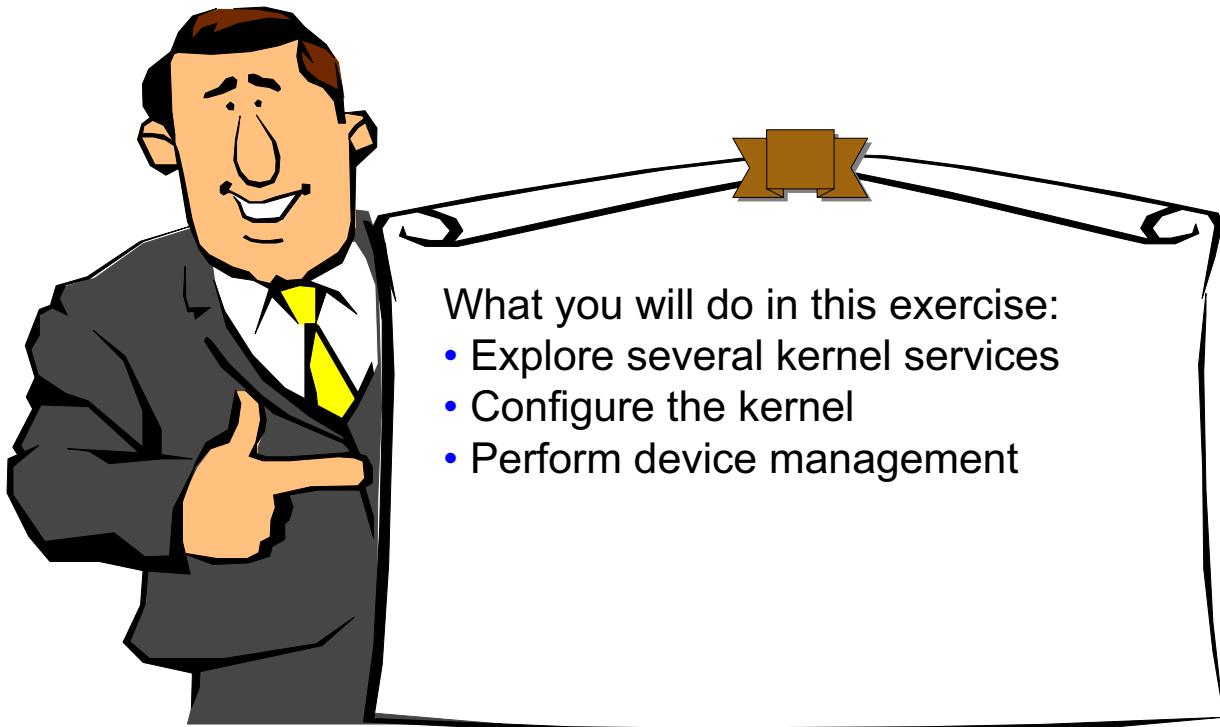
4. True or false: A character device allows random seeks.  
The answer is true.
5. What is the difference between /dev/random and /dev/urandom?  
The answer is /dev/urandom doesn't block when the entropy pool runs out.
6. Name a few commands that allow you to retrieve device information.  
The answers are lspci, lsusb and udevadm info.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —**

**Transition statement —**

# Exercise: Kernel services, kernel configuration and device management



What you will do in this exercise:

- Explore several kernel services
- Configure the kernel
- Perform device management

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-27. Exercise: Kernel services, kernel configuration and device management

LX158.0

## Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Unit summary

Having completed this unit, you should be able to:

- Describe the base characteristics of the Linux kernel
- Work with kernel modules
- List the most important kernel services
- Configure the Linux kernel
- Work with the /dev directory
- Know the traditional difference between block and character devices
- Work with virtual character devices
- Work with the loop device
- Retrieve device information

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 11-28. Unit summary

LX158.0

### Notes:

Having completed this unit, you should be able to understand:

- The Linux kernel is a monolithic, modularized kernel
- Device management is done by **udev** and reflected in the /dev filesystem, where the **devtmpfs** virtual filesystem is mounted
- Kernel, process and device information can be found in /proc and /sys
- The kernel can be configured at the command line (through the bootloader), when loading a module (using /etc/modprobe.conf) and through the /proc/sys interface
- In a modern Linux distribution, a **devtmpfs** filesystem is mounted on /dev, and entries in /dev are managed by the **udevd** daemon
- You can use a file as a block device through the use of the loop device

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

# Unit 12. Memory management

## Estimated time

00:30

## What this unit is about

This unit teaches you how Linux manages its memory.

## What you should be able to do

After completing this unit, you should be able to:

- Describe the principles of memory management in Linux
- Create paging space partitions
- Create paging space files
- Interpret results and reports generated by standard Linux tools

## How you will check your progress

Accountability:

- Checkpoint questions
- Machine exercises

## Unit objectives

---

After completing this unit, you should be able to:

- Describe the principles of memory management in Linux
- Create paging space partitions
- Create paging space files
- Interpret results and reports generated by standard Linux tools

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 12-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

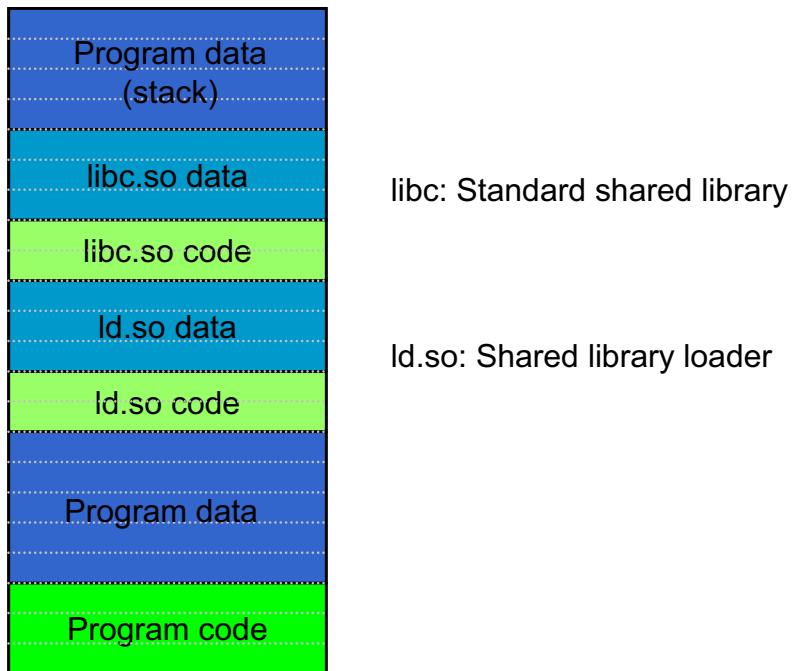
**Transition statement —**

# Memory usage: Process view

Memory usage  
example of a single  
program  
(4k pages)

Use  
**pmap -x <PID>**  
to view process  
memory usage

Total amount of  
memory required for  
the program is  
called the "virtual  
image size" (VIRT)



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-2. Memory usage: Process view

LX158.0

## Notes:

Linux memory management works in "pages" of a fixed size. On most architectures, a page size of 4KB is used, although some architectures use different page sizes, and some architectures also support "huge pages", where large amounts of memory (typically around 4 MB) can be allocated and managed as one big unit.

A process that's loaded into memory takes up several of these pages. Typically pages are used for a number of things:

- The program code itself.
- The program data: The data pages that have been allocated by the kernel because the program requested the kernel to do so. These pages contain internal datastructures needed for the program.
- The Shared Library (**ld.so**) loader. This is a small piece of code that loads shared libraries when needed. It needs to be attached to all programs that use shared libraries.
- The ld.so loader needs a few KB of data to function.

- Shared libraries, such as **libc**, are pieces of code that are compiled, stored on the system and made available to all programs. This is because programs typically use a lot of common functionality, for instance to access files, write text to the screen, interface with network adapters and protocols and so forth. If each programmer would reinvent the wheel, the overhead of doing so would be tremendous. Therefore only a few programmers concern themselves with the creation and optimization of these shared libraries. Once they are finished, the code is available on the system for everybody to use.

There are thousands of shared libraries available on a typical Linux system. Most of these are stored in /usr/lib.

- Some shared libraries need a few KB of memory for their own internal datastructures.
- The last bit of memory is typically the "stack". This mostly holds subroutine variables. For the purposes of this unit however, we can consider the stack as holding the same kind of data as the program data block we saw earlier.

The typical size of the memory, as viewed by the program, depends greatly on the program itself. A small program might only consume a few pages, while large programs may use several GBs of memory.

## **Instructor notes:**

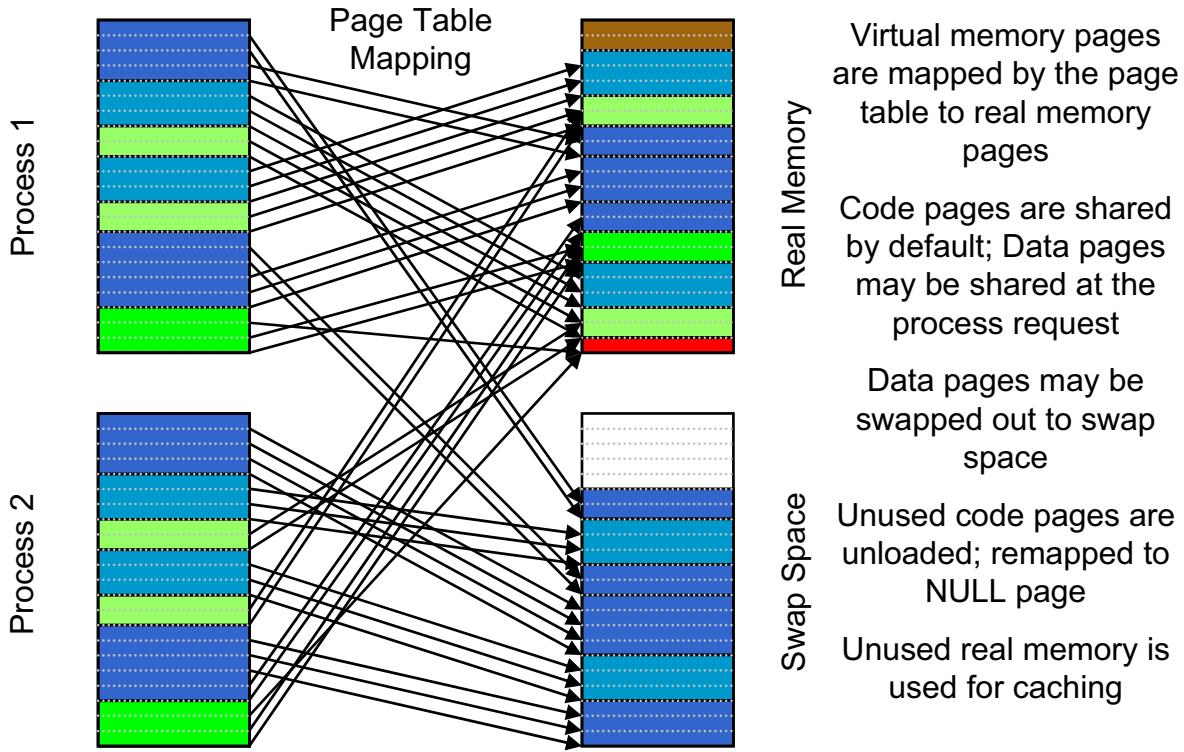
**Purpose** — Discuss the process view of memory.

**Details** —

**Additional information** — Note that all the memory that's listed here can be directly addressed by the process. But the process is typically not allowed to write outside its own memory region.

**Transition statement** — Our unit would be finished if this single process was the only process on the system. But it's not, and memory management has become very complex. Let's see how "Virtual Memory" works.

# Virtual memory



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-3. Virtual memory

LX158.0

## Notes:

If the process we saw on the previous page was the only process running, memory management would be very easy. But on a typical Linux system, you might have hundreds of processes running, all competing for memory. For this reason a number of optimizations have been added.

All these optimizations are dependent on a concept called "Virtual Memory". Virtual memory means that the process view of the memory that's available to it, does not reflect reality. Instead, the kernel creates a "Page Table", which is a lookup table that creates a mapping between what a process thinks its memory model is (the "virtual memory") and the location where the memory pages are really stored in real memory.

The Page Table format is CPU architecture specific. It's the kernel which creates and manages the page table, but the CPU needs to be able to lookup the translation between a virtual address and a real address directly, without calling kernel routines. Otherwise virtual memory would become incredibly slow.

So if a process wants to write something to a virtual memory page, it's the CPU which translates that address to the real memory address.

By using Virtual Memory, a number of optimizations are suddenly possible, and are used in Linux.

First, program code, both of the process itself and of any shared libraries, is only loaded into memory when it is actually needed. A big program may have hundreds, or even thousands of memory pages with program code, but the user may never need a certain bit of code. If that bit of code is not loaded into memory, it doesn't consume any resources, so this saves memory.

Technically this is implemented by letting the virtual memory address of these pieces of code point to the "null page" in the page table. When a program tries to access a virtual memory page that maps to this null page, an interrupt is generated, the program is halted and control is passed to the kernel. The kernel can then load this page into real memory, update the page table reference to point to this freshly loaded page, and the program can continue running.

Second, program code, both of the process itself and of any shared libraries, is only loaded into memory once. So if you have multiple instances of the same program, or multiple processes that need the same shared library, all their virtual memory pages simply point to the same real memory page, where this bit of code is stored.

Third, any program code pages that have not been used recently will be freed when the system runs low on memory. The virtual pages are then remapped to the null page. When the code is needed again, the pages are simply loaded from disk again.

Fourth, any program data pages that have not been used recently may be paged out to a special area on disk called the "paging space" or "swap space". The virtual memory mapping is then updated so that the CPU again generates an interrupt if these data pages are needed again. This allows the kernel to bring in the data page from disk, fix the mapping again, and let the process continue.

Data pages, by the way, are not shared by default. However, there are mechanisms such as System V shared memory, that allows programs to request shared memory pages from the kernel. These shared memory pages can then be used for inter-process communication. On the other hand, code pages are shared by default.

The last optimization is that any unused real memory is automatically used to cache file and filesystem data. This means that a file that has been accessed recently will likely still be cached in memory. The next time a file is accessed, you can read it from memory instead of having to wait for a relatively slow hard disk.

Data caching is also used for "write caching". Any write request to disk will first be stored in memory. This allows the block device driver to sort the write requests in the order that generates the best throughput with the lowest latency. Which increases the overall performance of the system.

**Instructor notes:**

**Purpose** — Discuss virtual memory

**Details** —

**Additional information** — Stress that memory management is an incredibly complex subject, and we have only scraped the surface here. It doesn't matter all that much if the students don't understand the intimate details, as long as they realize that there's not a one-on-one relation between virtual memory (as the process experience it) and real memory.

**Transition statement** — Okay, so now the million dollar question: How much memory does my process consume?

## Memory metrics

- **VIRT** (Virtual Image Size): Total amount of memory that's visible to the process.
- **RSS** (Resident Set Size): Amount of process memory that's currently in real memory: Data + Code
- **SHR** (Shared memory): Amount of process memory that is, or can be, shared with other applications
  - Mostly code, but sometimes data as well
- **SWAP = VIRT - RES**
  - Note that SWAP also incorporates unloaded code pages
- **Cache**: File data cache
- **Buffers**: File/directory/filesystem metainformation cache (superblock, inodes and so forth)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-4. Memory metrics

LX158.0

### Notes:

A question that is asked very frequently is: "How much memory is my process consuming?" Because of the complexities of virtual memory management, there is no clear answer on this. There are a number of metrics that are useful to answer this question.

**VIRT**, for Virtual Image Size, is the total amount of virtual memory that is somehow mapped in the page table. In other words, it's the total amount of memory that's visible from the process. If the process was the only process on the system, and no swap space or caching was done, and we're not taking into account memory set aside by the kernel for its own code, data structures, device mappings and so forth, this would be the total amount of memory consumed on the system.

**RSS**, for Resident Set Size, is the total amount of memory that the process uses, and that currently resides in memory. But note that this includes any shared memory pages, so if you add up all the RSS numbers, you get a number that is a lot larger than the actual real memory consumption.

**SHR**, for Shared, is the amount of process memory (RSS) that is, or potentially can be shared with other applications. This mostly applies to code pages, who are almost always

---

shareable. But there may also be pages of data memory that can be shared, if the application uses System V shared memory.

**SWAP** is the amount of virtual memory of the process that somehow doesn't reside in real memory. That may of course be data pages that are swapped out into the swap space, but may also cover code pages that were not needed, or not needed anymore, and thus have not been loaded yet, or have been unloaded.

Those four metrics can be used to answers questions such as "How much memory do I need to run one process X", "How much memory do I need to run X and Y, assuming they use Z as shared library", or "How much memory do I need to run an additional process X, when I already have at least one process X running?"

There are also two other metrics that allow you to describe how much memory is not used for applications, but is used for caching.

**Cache** is the amount of file data cache, and **buffers** is the amount of file/directory/filesystem metainformation cache (superblocks, inodes and so forth).

## **Instructor notes:**

**Purpose** — Describe a few metrics that are important to figure out the amount of memory consumed by processes.

**Details** —

**Additional information** —

**Transition statement** — Let's look at a series of commands that allow you to figure out the memory usage of your system and processes.

## free command, /proc/meminfo

```
sys1# free
      total        used        free      shared      buffers      cached
Mem:   3953920    3842316     111604          0    390024    1266412
  -/+ buffers/cache:  2185880    1768040
Swap:  4194296     275040    3919256

sys1# cat /proc/meminfo
MemTotal:      3953920 kB
MemFree:       116832 kB
Buffers:       390328 kB
Cached:        1259172 kB
SwapCached:    18096 kB
Active:        2175772 kB
Inactive:      1254712 kB
Active(anon):  1317016 kB
Inactive(anon): 585852 kB
Active(file):  858756 kB
Inactive(file): 668860 kB
...
...
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-5. free command, /proc/meminfo

LX158.0

### Notes:

The free command provides a high-level view of system memory and swap resources. More detailed information can be found in /proc/meminfo.

In regards to overall system memory performance, the example shown in the visual shows:

- The system has ~3.9 GB of memory available for applications (real memory - reserved bits - kernel binary code)
- The system is using ~3.8 GB of memory
- There is ~111 MB of memory free
- Out of the ~3.8 GB memory used, about 390 MB is used for buffers (filesystem metainformation) and about 1.2 GB for file data cache.
- If you want to know how much memory is used by applications, you need to subtract the buffers and cache from the memory usage. If you do that, you see that about 2.1 GB of memory is in use by processes, and if we did not do any caching or buffering, 1.7 GB of memory would be free.
- The system is using of ~275 MB out of the 4.1 GB of swap space

The "shared" parameter in the output of "free" is deprecated and will always be zero. You should ignore this - it has no relevance whatsoever.

The /proc/meminfo file provides current system-wide memory performance statistics. The contents of /proc/meminfo is far more detailed than the output of the free command. The visual only shows the first few lines.

In the visual, both examples are from the same system. The reason for the difference in, for instance, MemFree is that the free command itself has different requirements compared to the cat command. The output of the free command is obviously generated while free is running, while the output of the cat command is generated while cat is running. So there will always be a slight difference. Memory management is, after all, a dynamic and ongoing process.

The following table lists the memory statistics shown in /proc/meminfo and their description. All output values are shown in KB.

MemTotal	Total amount of physical memory (real memory - reserved bits - kernel binary code).
MemFree	Total amount of free physical memory.
Buffers	Total amount of memory being used for buffers.
Cached	Total amount of memory being used for page cache.
SwapCached	Memory that once was swapped out, is swapped back in, but still also is in the swapfile (if memory is needed, it does not need to be swapped out again because it is already in the swapfile. This saves I/O).
Active	Total memory currently active (will not be swapped, unless absolutely necessary).
Inactive	Total memory currently inactive (can be swapped, if needed).
HighTotal	Total memory allocated as high memory (ZONE_HIMEM used for Intel architecture; PPC64 does not use high memory).
HighFree	Total high memory that is free.
LowTotal	Total memory allocated as low memory.
LowFree	Total low memory that is free.
SwapTotal	Total amount of swap space (swap space allocated for use; in this example, a partition assigned as swap 0x82).
SwapFree	Total amount of swap space available for use.
Dirty	Memory waiting to be written to disk.
Writeback	Memory currently being written to disk.
Mapped	Total amount of memory mapped into a process's virtual address space.
Slab	Total amount of memory assigned to slab caches.

---

Committed_AS	Total amount of memory allocated to processes. Linux overallocates memory to processes. This is the total memory if all processes utilize all of their allocated memory.
PageTables	Total amount of memory that is reserved for kernel page tables.
VmallocTotal	Total amount of memory that is usable for vmalloc.
VmallocUsed	Total amount of vmalloc memory used.
VmallocChunk	Largest contiguous chuck of memory that can be vmalloc.
HugePages_Total	Total number of reserved huge pages.
HugePages_Free	Total number of free huge pages.
Hugepagesize	Size of a huge page (architecture dependent; 4 MB for i386).

**Instructor notes:**

**Purpose** — Introduce students to the memory statistics shown in /proc/meminfo.

**Details** — The /proc/meminfo file provides a good overall look at current system memory statistics.

**Additional information** —

**Transition statement** — Next, let's look at the ps command.

## ps command

- The process status command **ps** generates a snapshot of process information.
- Large number of output formatting options

```
# ps -o vsz,rss,tsiz,dsiz,majflt,minflt,cmd 15299
    VSZ   RSS TSIZ DSIZ MAJFLT MINFLT CMD
  45396  5868  312 45083      0    757 /usr/sbin/httpd2-prefork -f
/etc/apache2/ht
# ps aux|sort -nr +3|grep -v USER|head -5
root      9291  0.0 11.8 70812 45700 pts/3      S1     Apr18    0:28
/usr/lib/YaST2/
bin/y2base sw_single qt --fullscreen
root      3610  0.0  6.4 29116 24828 tty7      Ss+   Apr04    0:34
/usr/X11R6/bin/
X :0 -audit 0 -auth /var/lib/gdm/:0.Xauth -nolisten tcp vt7
root      3124  0.0  4.7 63716 18208 ?          RNs1 Apr04    1:09
/usr/lib/zmd/
zmd-bin /usr/lib/zmd/zmd.exe
root      27576 0.0  4.5 30208 17464 ?          S1     Apr15    0:11
/usr/bin/mono
/usr/lib/zen-updater/ZenUpdater.exe
root      4642  0.0  4.5 30292 17432 ?          S1     Apr04    0:31
/usr/bin/mono
/usr/lib/zen-updater/ZenUpdater.exe
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-6. ps command

LX158.0

### Notes:

#### Introduction

As we have seen so far, there are a number of tools to track system memory utilization. One tool that is useful in looking at how a process is consuming memory is the **ps** command. From the previous unit, we have seen how the **ps** command is a basic tool that can be used to report a number of statistics about running processes on the system. It reports static values, such as:

- Process ID
- Parent process ID
- Controlling TTY
- CPU Utilization
- Memory Utilization
- Accumulated CPU time
- Nice value
- Priority

The ps command has a number of options to define information to display and how to format the output. The command ps L will list the various fields that can be displayed.

In regards to overall system memory utilization, the example shown in the visual shows the following range of data:

- The first example shows the command httpd-prefork has:
  - A virtual set size (VSS) of 45396 KB
  - A resident set size (RSS) of 5868 KB
  - A text size (TSIZ) of 312 KB
  - A data size (DSIZ) of 45083 KB
  - Created minor page faults (MINFLT, pages that can be provided from cache) totaling 757 since it began execution
- The second example shows the top five consumers of memory (%MEM) running on the system. The highest consumer of memory is the /usr/lib/YaST2/bin/y2base process (PID 9291).

## Operation

When using the ps command, a number of options and options styles can be utilized.

The command recognizes the following format styles:

- UNIX options, which can be grouped and must be preceded by a dash
- BSD options, which can be grouped and must not be used with a dash
- GNU long options, which are preceded by two dashes

Thus, it is likely that the ps command options you have used on other operating systems such as AIX will probably work on Linux.

The ps command can be used as a diagnostic tool to find processes causing performance issues or to track a specific process. When using it to locate a process causing a performance issue, the ps command can be used with options to format the display of details about a process. For example, to show all system processes with output with virtual set size, resident set size, text size, data size, major faults, and minor faults, enter the following:

```
# ps -eo vsz,rss,tsiz,dsiz,majflt,minflt,cmd  
VSZ      RSS TSIZ DSIZ MAJFLT MINFLT CMD  
628    284 537    90     8    558 init [3]  
0      0   0    0     0      0 [migration/0]  
0      0   0    0     0      0 [ksoftirqd/0]  
0      0   0    0     0      0 [migration/1]  
0      0   0    0     0      0 [ksoftirqd/1]
```

```

...
11320      3104 156 11163      0      4 /usr/sbin/rsct/bin/IBM.DRMd
11320      3104 156 11163      0      2 /usr/sbin/rsct/bin/IBM.DRMd
11320      3104 156 11163      0      88 /usr/sbin/rsct/bin/IBM.DRMd
1688       656 150   1537      0  54397 /sbin/iprupdate --daemon
...

```

To track a specific process ID, the ps command can be used to display data about that particular process. For example, to display output formatted data for process ID 6003, enter the following:

```
# ps -o vsz,rss,tsiz,dsiz,majflt,minflt,cmd 6003
VSZ      RSS TSIZ DSIZ          MAJFLT MINFLT CMD
1704    568 274    1429      0      171  /bin/ksh ./busycpu
```

### Command options

The ps command has a number of options and commands that can change how information is displayed while it is running. The command syntax for ps is as follows:

```
# ps [ options ]
```

Commonly used options are as follows:

<b>aux</b>	Show all processes running on the system (BSD) syntax
<b>axjf</b>	Show all processes with a process tree (BSD syntax)
<b>-o &lt;format&gt;</b>	Output format control. See man page for detailed list of selectable fields, or use: # ps L

For complete information, check the man page.

**Instructor notes:**

**Purpose** — Introduce students to the ps command as it relates to process utilization of memory.

**Details** — When discussing the ps command this time around, emphasize how it can be used to track a process's use of system memory.

**Additional information —**

**Transition statement** — Next, let's look at the top command.

## top command

- top provides comprehensive real-time performance data.

```
top - 04:46:19 up 16 days, 1:10, 7 users, load average: 0.00, 0.00, 0.00
Tasks: 117 total, 1 running, 115 sleeping, 1 stopped, 0 zombie
Cpu(s): 0.3% us, 1.0% sy, 0.0% ni, 98.7% id, 0.0% wa, 0.0% hi, 0.0% si
Mem: 385592k total, 378184k used, 7408k free, 44192k buffers
Swap: 1044216k total, 24k used, 1044192k free, 127920k cached

      PID USER      PR  NI    VIRT    RES    SHR   S %CPU %MEM     TIME+ COMMAND
20619 root      17   0  2188 1032  776 R  1.6  0.3  0:00.52 top
      1 root      16   0   716  288  248 S  0.0  0.1  0:01.65 init
      2 root      34  19     0     0     0 S  0.0  0.0  0:00.00 ksoftirqd/0
      3 root      RT   0     0     0     0 S  0.0  0.0  0:00.00 watchdog/0
      4 root      10  -5     0     0     0 S  0.0  0.0  0:00.05 events/0
      5 root      10  -5     0     0     0 S  0.0  0.0  0:00.09 khelper
      6 root      10  -5     0     0     0 S  0.0  0.0  0:00.00 kthread
      8 root      10  -5     0     0     0 S  0.0  0.0  0:00.12 kblockd/0
      9 root      18  -5     0     0     0 S  0.0  0.0  0:00.00 kacpid
     113 root     11  -5     0     0     0 S  0.0  0.0  0:00.00 aio/0
     112 root     15   0     0     0     0 S  0.0  0.0  0:01.59 kswapd0
     318 root     11  -5     0     0     0 S  0.0  0.0  0:00.00 cqueue/0
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-7. top command

LX158.0

### Notes:

#### Introduction

The top command is a common UNIX/ Linux utility that provides a comprehensive real-time performance data view of a running system. It has the capability to display system summary information and a list of processes currently being scheduled. The display is user configurable, and can be customized to display information by:

- Type
- Size
- Order

In regards to overall system memory performance, the example shown in the visual shows:

- The system has ~376 MB of memory (real memory - reserved bits - kernel binary code)
- The system is using ~369 MB of memory
- There is ~7.2 MB of memory free
- The system is using ~43 MB of memory for buffers
- The process top is consuming 0.3% of memory

## Operation

The default configuration of top will display system summary information followed by a list of scheduled processes. The configuration can be modified to be persistent across restarts by writing a configuration file with the W command (\$HOME/.toprc)

### Command options

The top command has a number of options and commands that can change how information is displayed while it is running. The command syntax for top is as follows:

```
# top [ options ] [ -d delay ] [ -n iterations ][ -p process_id ]
```

Commonly used options are as follows:

- b** Starts top in batch mode. Useful to send output from top to another performance program.
- d delay** Delay between screen updates.
- n iterations** The number of iterations to run before exiting top.
- u user** Monitor only processes with an effective UID or user name matching that given.
- pN1 -pN2 . . .** Monitor only processes with specified process IDs. This option can be given up to 20 times, or you can provide a comma delimited list with up to 20 PIDs. Co-mingling both approaches is permitted.

Commands within top toggle settings on/off. Some useful commands are as follows:

- A** Sorts the display by top consumers of various system resources; very useful to quickly identify the performance hungry tasks on a system
- f** Enters an interactive configuration screen for top; very helpful to set up top for a specific task

**Note:** Enter j to add the last used CPU column.

- o** Allows you to interactively select the ordering within top
- m** Shows only running tasks

### RPM package

The top command can be found in the following RPM packages:

- SLES - procps-X.X.X-X.X
- RHEL/Fedora - procps-X.X.X-X.X

**Instructor notes:**

**Purpose** — Re-introduce students to the top command as it relates to system memory.

**Details** — The top command provides a real-time summary display of what is happening on the system. It is a commonly used performance analysis command on UNIX/Linux systems.

**Additional information —**

**Transition statement** — Next, let's visit the procinfo-ng command.

## Procinfo-ng command

- Provides system status from information found in /proc

```
# procinfo-ng
Linux 2.6.16-rc1-git3-7-default (geeko@buildhost) (gcc 4.1.0 20060123) #1 1CPU [sys2]

Memory:      Total        Used        Free        Shared       Buffers
Mem:        385592      380532      5060          0      44680
Swap:      1044216           24     1044192

Bootup: Tue Apr  4 03:36:16 2006   Load average: 0.00 0.00 0.00 1/156 20559

user :      1:51:45.35  0.5% page in :    1599745 disk 1: 106169r 1633986w
nice :      0:02:11.66  0.0% page out: 11273264 disk 2:      406r      0w
system:    0:30:55.60  0.1% page act:  571729
IOWait:    0:09:50.56  0.0% page dea:  503875
hw irq:    0:00:30.53  0.0% page flt:  59170706
sw irq:    0:00:04.39  0.0% swap in :      14
idle :    15d 22:26:34.03 99.3% swap out:      20
uptime:   16d  1:01:52.48 context : 567926192

irq 0: 346506380 timer           irq 7:      2 parport0 [3]
irq 1: 2106 i8042               irq 8:      2 rtc
irq 2: 0 cascade [4]            irq 9:      1 acpi
irq 3: 4                         irq 10:     0 uhci_hcd:usb1
irq 4: 4                         irq 12:    17546 i8042
irq 5: 139662 eth0              irq 14:    1740168 ide0
irq 6: 5                         irq 15:    12419696 ide1
uptime: 3d 18:16:20.77         context : 11717465      interrupts: 1055180
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-8. Procinfo-ng command

LX158.0

### Notes:

#### Introduction

The SLES release provides the procinfo command to display system status information based on values gathered from the /proc virtual file system. The procinfo command can be run as a single snapshot, continuous update, or continuous update with delta values (-d option) to display the system status.

In regards to overall system memory utilization, the example shown in the visual shows:

- The system has ~376 MB of memory of total memory (real memory - reserved bits - kernel binary)
- The system is using ~371 MB of memory
- There is ~4.9 MB of memory free
- The system is using of ~43.6 MB of memory for buffers

## Operation

Looking at the example shown in the visual, the procinfo command was executed to display to current system statistics. The statistics presented are presented below:

<b>Total</b>	Total amount of physical memory (real memory - reserved bits - kernel binary code)
<b>Used</b>	Total amount of allocated memory
<b>Free</b>	Total amount of free memory
<b>Shared</b>	Obsolete - Ignore
<b>Buffers</b>	Total amount of memory used for buffers
<b>page in</b>	Total number of blocks read in from disk
<b>page out</b>	Total number of blocks written to disk
<b>swap in</b>	Total number of pages read in from swap
<b>swap out</b>	Total number of pages written to swap

## Command options

The procinfo command has a number of options and commands that can change how information is displayed while it is running. The command syntax for procinfo is as follows:

```
# procinfo [ options ] [ -n seconds ] [ -f file ]
```

Commonly used options are as follows:

- f Run in full screen mode
- d Display delta of statistics between samples rather than totals
- D Display statistics totals
- n Delay between interval sample seconds
- f filename Output statistics to a file

## RPM package

The procinfo command can be found in the following RPM package:

- SLES - procinfo-X-X.X

## **Instructor notes:**

**Purpose** — Introduce the students to the procinfo command as it relates to system memory.

**Details** — The procinfo command displays system status information found in the virtual /proc file system. It has a number of fields related to overall system memory performance.

**Additional information** — The book Optimizing Linux Performance A Hands-On Guide to Linux Performance Tools by Phillip G. Ezolt, Copyright 2005 Hewlett-Packard Development Company, L.P., 0-13-148682-9, notes that the following entries are broken with the 2.6 kernel. Need to verify this statement for current versions of Linux on PPC64:

- page in
- page out
- swap in
- swap out

**Transition statement** — Next, let's look at the vmstat command.

## vmstat command

- Print continuous report on virtual memory and CPU statistics

```
# vmstat 4 5
procs -----memory----- swap-----io-----system-----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa
1 0 24 7540 44668 127956 0 0 1 8 13 7 0 0 99 0
0 0 24 7540 44696 127956 0 0 0 8 261 415 0 0 100 0
0 0 24 7540 44696 127956 0 0 0 5 261 414 0 0 100 0
0 0 24 7540 44708 127956 0 0 0 5 261 407 0 0 100 0
0 0 24 7532 44720 127956 0 0 0 5 261 413 0 0 100 0

# vmstat -a 4 5
procs -----memory----- swap-----io-----system-----cpu-----
r b swpd free inact active si so bi bo in cs us sy id wa
1 0 24 6980 75772 245028 0 0 1 8 13 7 0 0 99 0
0 0 24 6856 75796 245036 0 0 0 13 262 430 0 0 100 0
0 0 24 6856 75808 245088 0 0 0 29 264 425 0 0 100 0
0 0 24 6856 75840 245088 0 0 0 14 262 421 0 0 100 0
0 0 24 6732 75852 245092 0 0 0 7 261 438 0 0 100 0
```

Figure 12-9. vmstat command

LX158.0

### Notes:

#### Introduction

The vmstat command is designed to report statistics about processes, memory, paging, block I/O, traps, and CPU activity. When looking at system-wide CPU-related performance issues, the vmstat command can provide a high-level snapshot of what the system is doing.

The vmstat command runs in either average or sample modes. If the vmstat command is issued without a delay option, vmstat will display system averages since the system was booted. If a sample delay is specified, the first line of output will show the system averages since the system was booted and, after that, it will display statistics captured during the sample delay period.

In regards to overall system memory performance, the first example shown in the visual shows:

- The system is not swapping memory out (0 pages)
- The system has ~7.3 MB of free memory
- The kernel is using ~43.6 MB of memory for buffers
- The kernel is using ~124.9 MB of memory for cache

The second example shown in the visual shows the use of the -a option. This option replaces the buff/cache columns with inact (inactive) and active memory columns.

## Operation

Looking at the example shown in the visual, the vmstat command was executed to display to standard out with a four-second delay between sample intervals for five times. Using this method placed vmstat in interactive mode. Using vmstat in average mode displays values averaged since the system was booted:

```
# vmstat
```

The values related to memory activity either in interactive or average mode are as follows:

- swpd** The amount of virtual memory used
- free** The amount of free memory
- buff** The amount of memory used as buffers
- cache** The amount of memory used as cache
- si** Total amount of memory swapped in
- so** Total amount of memory swapped out
- active** The amount of active memory
- inactive** The amount of inactive memory

## Command options

The vmstat command has a number of options and commands that can change how information is displayed while it is running. The command syntax for vmstat is as follows:

```
# vmstat [ options ] [delay [ count ] ]
```

Commonly used options are as follows:

- a** Displays active/inactive memory versus the default display of buffers/cache utilization
- s** Outputs a single snapshot of memory and CPU related statistics
- m** Displays slab information
- delay** Number of seconds to wait between sample interval
- count** Number of times to sample the system

For complete information, check the man page.

## RPM package

The vmstat command can be found in the following RPM packages:

- SLES - procps-X.X.X-X.X
- RHEL/Fedora - procps-X.X.X-X.X

**Instructor notes:**

**Purpose** — Re-introduce students to the vmstat command as it relates to system memory.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Next, let's look at the vmstat -s command as it relates to system memory.

## vmstat -s command

- Print single table of virtual memory and CPU statistics since the system was booted

```
# vmstat -s
 385592  total memory
 379108  used memory
 245156  active memory
 76124   inactive memory
 6484    free memory
 45508   buffer memory
 128312  swap cache
1044216  total swap
     24  used swap
1044192  free swap
 670664 non-nice user cpu ticks
 13166 nice user cpu ticks
.
.
.
 1601781 pages paged in
11281804 pages paged out
    14 pages swapped in
    20 pages swapped out
.
.
.
 220549 forks
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-10. vmstat -s command

LX158.0

### Notes:

#### Introduction

To gather a single snapshot of memory and CPU related statistics with the vmstat command, use the -s option. This option gives detailed information regarding memory and swap utilization.

In the example shown in the visual, the values shown are those since the system was booted. Using the values displayed:

- The system has ~376 MB of memory (real memory - reserved bits - kernel binary code)
- The system is using ~370 MB of memory
- There is ~6.33 MB of memory free
- The kernel is using ~44.44 MB of memory for buffers

**Instructor notes:**

**Purpose** — Re-introduce the students to the -s option to the vmstat command as it relates to system memory.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Okay, those are the commands that allow us to track memory usage. If we need more detail, there's also more information in the /proc directory.

## Process memory: /proc/PID/status

- Each process has a subdirectory under /proc
- The status file provides a wide variety of information, including memory utilization statistics.
- The statm file provides process memory usage information.

```
# cat /proc/15299/status
Name: httpd2-prefork
State: S (sleeping)
SleepAVG: 88%
Tgid: 15299
Pid: 15299
PPid: 1
TracerPid: 0
Uid: 0 0 0 0
Gid: 0 0 0 0
FDSize: 32
Groups: 0
VmPeak: 45396 kB
VmSize: 45396 kB
VmLck: 0 kB
VmHWM: 5868 kB
VmRSS: 5868 kB
VmData: 2024 kB
VmStk: 84 kB
VmExe: 316 kB
VmLib: 8668 kB
VmPTE: 24 kB
Threads: 1
SigQ: 1/3063
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000001000
SigCgt: 000000018800466b
CapInh: 0000000000000000
CapPrm: 0000000fffffefff
CapEff: 0000000fffffefff
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-11. Process memory: /proc/PID/status

LX158.0

### Notes:

#### Introduction

The /proc virtual file systems provides a view into the kernel's configuration and operation. The kernel provides a directory structure under /proc for each currently running process. The directory is named after the process identification (PID) and contains details about the process. For example, the init process (PID 1) has a directory structure /proc/1. A number of files and subdirectories can be found under the PID directory structure. Some sample files are:

- **cmdline**: Contains the command that was used to start the process
- **cwd**: Contains a link to the current working directory of the process
- **environ**: Contains a list of the environment variables that the process has available
- **exe**: Contains a link to the program that is running in the process
- **mem**: Contains the memory contents of the process
- **stat**: Contains process status information

- **statm**: Contains process memory usage information
- **status**: Contains process status information in a readable format
- **maps**: Contains the mapping of the process address space that is mapped to a file **/proc/PID/status**

The contents of the status file provides a wide variety of information, including memory utilization statistics. Some of the information it contains is:

- State: Running/sleeping
- Process ID
- Parent Process ID
- User ID
- Group ID
- Number of threads
- Virtual Memory statistics

The information related to the virtual memory statistics are as follows:

<b>VmSize</b>	The size of the virtual memory allocated to the process
<b>VmLck</b>	The amount of locked memory
<b>VmRSS</b>	The amount of memory mapped into physical memory
<b>VmData</b>	The size of the data segment
<b>VmStk</b>	The stack size
<b>VmExe</b>	The size of the executable segment
<b>VmLib</b>	The size of the library code

**Note:** This information can be viewed in a more efficient manner by using tools like top or ps.

### **/proc/<PID>/statm**

The contents of the statm file contains the status of memory in use by the process. For example, to display the contents of the init process (PID 1) statm file, enter the following:

```
# cat /proc/1/statm
```

```
157 71 62 120 0 37 0
```

The seven values displayed show different memory statistics. The following table shows the value positions and their descriptions:

1. Total program size (KB)
2. Size of memory portions (KB)
3. Number of pages that are shared

4. Number of pages that are code
5. Number of pages of data/stack
6. Number of pages of library
7. Number of dirty pages

## **Instructor notes:**

**Purpose** — To introduce students to the /proc/PID/status and statm files as they relate to process memory utilization.

**Details** — The /proc/PID directory structure contains information about currently running processes. Under each PID directory is a file call status that contains memory utilization information.

### **Additional information —**

**Transition statement** — Next, let's look at the /proc/PID/maps file.

# Process memory: /proc/PID/maps

```
# cat /proc/15299/maps
80000000-8004f000 r-xp 00000000 03:06 99234      /usr/sbin/httpd2-prefork
8004f000-80052000 rw-p 0004e000 03:06 99234      /usr/sbin/httpd2-prefork
80052000-8020b000 rw-p 80052000 00:00 0          [heap]
b554e000-b754e000 rw-s 00000000 00:07 4456453      /SYSV00000000 (deleted)
b754e000-b7583000 r--s 00000000 03:06 137752      /var/run/nscd/dbAveZie (deleted)
b7583000-b75b8000 r--s 00000000 03:06 137751      /var/run/nscd/group
b75b8000-b75ed000 r--s 00000000 03:06 137750      /var/run/nscd/passwd
b75ed000-b7713000 r-xp 00000000 03:06 51630       /usr/lib/libxml2.so.2.6.23
b7713000-b771c000 rw-p 00126000 03:06 51630       /usr/lib/libxml2.so.2.6.23
b771c000-b771d000 rw-p b771c000 00:00 0          [stack]
.
.
.
b7f6c000-b7f6e000 r-xp 00000000 03:06 106457      /usr/lib/apache2/mod_auth_basic.so
b7f6e000-b7f6f000 rw-p 00001000 03:06 106457      /usr/lib/apache2/mod_auth_basic.so
b7f6f000-b7f71000 r-xp 00000000 03:06 106455      /usr/lib/apache2/mod_alias.so
b7f71000-b7f72000 rw-p 00002000 03:06 106455      /usr/lib/apache2/mod_alias.so
b7f72000-b7f73000 r-xp 00000000 03:06 106454      /usr/lib/apache2/mod_actions.so
b7f73000-b7f74000 rw-p 00001000 03:06 106454      /usr/lib/apache2/mod_actions.so
b7f74000-b7f8e000 r-xp 00000000 03:06 14195       /lib/ld-2.3.90.so
b7f8e000-b7f90000 rw-p 00019000 03:06 14195       /lib/ld-2.3.90.so
bf83e000-bf853000 rw-p bf83e000 00:00 0          [stack]
fffffe000-fffff000 ---p 00000000 00:00 0          [vdso]
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-12. Process memory: /proc/PID/maps

LX158.0

## Notes:

### Introduction

The **/proc/PID/maps** file contains information related to the process memory map that is mapped to a file (typically to a library). The columns from left to right contain the following information:

- Address space associated with mapping
- Permissions of the memory region
  - **r**: Read
  - **w**: Write
  - **x**: Execute
  - **s**: Shared
  - **p**: Private
- Offset from the beginning of the file where the mapping starts

- The device where the mapped file is located (major/minor device number)
- The i-node number of the file
- The file that the region is mapped to

From the example on the visual, the memory region b75b8000-b75ed000:

- Has permissions of read and shared
- Has a zero offset into the mapped file
- The mapped file is located on major/minor device number 03:06
- The i-node of the mapped file is 137750
- The mapped file is /var/run/nscd/passwd

**Instructor notes:**

**Purpose** — Introduce students to the contents of the /proc/PID/maps file as it relates to process memory utilization.

**Details** — The maps file gives a view into what files (typically a library) that a process has mapped to.

**Additional information —**

**Transition statement** — Okay, that concludes the part of this unit where we can see how much memory we have, and how it is consumed. Let's look at a few management tasks. Surprisingly, there's not a lot we can do about how memory is used. Linux memory management is highly automated and optimized. The only thing we can really manage is the swap space.

## Creating paging space: Partition/LV/RAID

- We need an empty partition/LV/RAID volume.
  - Partition type 82 (Linux swap)
- Create paging space in that partition.

```
# mkswap /dev/sdb1
```

- Activate paging space and set its priority to 42.

```
# swapon -p 42 /dev/sdb1
```

- Check swap space in procfs (/proc).

```
# cat /proc/swaps
Filename           Type     Size   Used  Priority
/dev/mapper/VolGroup00-LogVol01  partition 1048568  160    -1
/dev/hdb1          partition  65563   4096   42
```

- Deactivating paging space is done using **swapoff**.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-13. Creating paging space: Partition/LV/RAID

LX158.0

### Notes:

#### Introduction

Paging space can be created either using a partition or a file. The visual shows the steps needed to use a partition as swap space. There are three steps in creating and activating paging space using a partition:

First, using the fdisk command, create an empty partition with a partition ID of 82 (Linux swap). This empty partition can be created from empty disk space, LVM logical volume, or a RAID volume.

Next, initialize a paging space in that partition with the mkswap command.

Finally, activate the paging space by using the swapon command. If the paging space needs to be activated at system startup, add an entry for this paging space to the /etc/fstab file.

## Minimum and maximum size of paging space

The minimum size of the paging space is 40 KB, and the maximum size is 2 GB (architecture dependent) when using kernel version 2.2 and up. In addition to that, the maximum number of paging spaces is 8. See the manual page of mkswap for details.



### Note

The Linux 2.4.10 kernel and later supports the use of up to 32 swap spaces on a system.

## Deactivating a paging space

Deactivating a paging space is done using the swapoff command. In contrast to most UNIX versions, this is possible on a running system as long as the space can be missed. If the amount of total memory becomes less than the amount needed, Linux starts to kill off random processes, so be careful with this command.

**Instructor notes:**

**Purpose** — Introduce students to the steps needed to create paging space using a partition/LV/RAID.

**Details** — Discuss student notes.

**Additional information** — The mkswap manual page has a breakdown of old and new style types of swap areas. See the mkswap manual page for details.

**Transition statement** — Let's look at how to create paging space using a file.

## Creating paging space: File

- Create a large file to act as swap space.

```
# dd if=/dev/zero of=/var/tmp/swapspace bs=1M count=64
# mkswap /var/tmp/swapspace
```

- Activate paging space.

```
# swapon /var/tmp/swapspace
```

- Check swap space in procfs (/proc).

```
# cat /proc/swaps
Filename           Type      Size    Used   Priority
/dev/mapper/VolGroup00-LogVol01 partition 1048568  160     -1
/swapspace          file       65528   0      -2
```

- Deactivate paging space using **swapoff**.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-14. Creating paging space: File

LX158.0

### Notes:

#### Introduction

It is possible to use a file for paging space too. This is less efficient (gives poorer performance) than paging space implemented with a partition/LV/RAID. It therefore should be used only in an emergency. The procedure for that is nearly the same, only you have to create a large file first, instead of a partition. Thus, the sequence becomes (for a 64 MB swapfile):

```
# dd if=/dev/zero of=/var/tmp/swapspace bs=1M count=64
# mkswap /var/tmp/swapspace
# swapon /var/tmp/swapspace
```

#### Deactivating a paging space

As mentioned earlier, deactivating a paging space is done by using the swapoff command.

### **Instructor notes:**

**Purpose** — Introduce students on how to create paging space using a file.

**Details** — Discuss student notes.

### **Additional information —**

**Transition statement** — The kernel will always try to avoid swap space, as swap space is really slow compared to main memory. But the kernel will also want to use main memory for caching. So there's a tradeoff to be made here, and that's more or less the only thing we can tune with regards to memory management. Let's see.

## /proc/sys/vm/swappiness

- Sets the "swappiness" of the kernel
  - Value between 0 and 100
  - Default 60
  - 0 means will try to avoid swapping at all costs
  - 100 means will always try to swap out inactive pages so memory becomes available for caching
- Various other tuning parameters in /proc/sys/vm
- Performance tuning:
  - Create baseline of performance metrics
  - Change one parameter, test, compare results with baseline
  - Document

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-15. /proc/sys/vm/swappiness

LX158.0

### **Notes:**

The kernel, as part of its memory management task, needs to make a trade-off: Do I need to swap a data page out to free up memory for a cached page, or not. Both of these have penalties: A swapped out data page needs to be brought into memory, eventually, which takes time. But any file that's cached in memory will save time, assuming that the file is going to be read again, at some point in the future.

Linux is not good at predicting the future, and therefore Linux allows the user to tune the behavior of the memory management subsystem. The most important parameter in this respect is the vm.swappiness parameter. It determines how "swappy" the kernel is: How often it will swap out a process data page in favor of caching a file data page.

The parameter is expressed as a number between 0 and 100. 0 means that data page swapping will be avoided at all costs. Swapping will only occur if the amount of memory that is required by applications exceeds the available real memory. In contrast, 100 means that the kernel will always try to keep its filesystem cache and buffers as large as possible, by swapping as many data pages as possible.

The default value of vm.swappiness is 60. If you decide to modify this value, do it properly: First determine the baseline performance of your system by running a standardized test suite appropriate to your usage pattern. Modify the parameter and run the test suite again. Only change one parameter at the time, then test.

This allows you to find the optimum value. You can add this to your /etc/sysctl.conf file so that it becomes permanent. And don't forget to document this.

**Instructor notes:**

**Purpose** — Discuss vm.swappiness

**Details** —

**Additional information** —

**Transition statement** — Okay, done.

## Checkpoint

---

1. Which file in /proc shows current system-wide memory performance statistics?
2. List two commands that provide system memory status.
3. What is the difference between a paging partition and a paging file? Which is more efficient?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 12-16. Checkpoint

LX158.0

### Notes:

**Instructor notes:****Purpose —****Details —**

## Checkpoint solutions

1. Which file in /proc shows current system-wide memory performance statistics?

The answer is [meminfo](#).

2. List two commands that provide system memory status.

The answer is [procinfo](#), [free](#), [top](#), [vmstat](#), [ps](#). (Answers may vary.)

3. What is the difference between a paging partition and a paging file? Which is more efficient?

The answers are [a paging partition is directly written in the partition table and to disk, while a paging file has to go through the file system](#). A paging partition is more efficient.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —****Transition statement —**

## Exercise: Memory management

---



What you will do in this exercise:

- Perform various memory management activities

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 12-17. Exercise: Memory management

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Unit summary

---

Having completed this unit, you should be able to:

- Describe the principles of memory management in Linux
- Create paging space partitions
- Create paging space files
- Interpret results and reports generated by standard Linux tools

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 12-18. Unit summary

LX158.0

### Notes:

Having completed this unit, you should understand:

- The kernel uses a virtual memory model
- A page cache is used to store recently accessed files
- Paging partitions or paging files can be created to increase virtual memory
- There are many Linux tools that can be used to extract system and process specific memory statistics

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**



# Unit 13. Virtualization

## Estimated time

01:00

## What this unit is about

This unit covers the virtualization solutions that are available for Linux. It discusses the kernel virtualization module (KVM), and will introduce both Xen and QEMU-KVM as virtualization solutions. A generic method for managing virtualization, libvirt, will also be covered.

## What you should be able to do

After completing this unit, you should be able to:

- List the different virtualization options in Linux
- Discuss the KVM kernel feature
- Discuss QEMU emulation
- Configure Xen
- Configure QEMU-KVM
- Discuss libvirt and use the Virtual Machine Manager

## How you will check your progress

- Checkpoint questions
- Lab exercises

## Unit objectives

---

After completing this unit, you should be able to:

- List the different virtualization options in Linux
- Discuss the KVM kernel feature
- Discuss QEMU emulation
- Configure Xen
- Configure QEMU-KVM
- Discuss libvirt and use the Virtual Machine Manager

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 13-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Virtualization overview

- Hardware emulation: Emulate a full system in software
  - Can be used for cross-architecture testing, such as ARM on x86
- Virtualization: Virtualize all hardware
  - Can be used to run multiple OSs on single system
- Paravirtualization: Virtualize hardware with OS support
  - The virtualized hardware is "special" and requires OS support/drivers
- Partitioning
  - Split the physical hardware into multiple, independent partitions
  - "Logical Partitioning" uses a hypervisor, "Physical Partitioning" does not
  - Not used in the x86 world, but very common, for example, on IBM Power

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-2. Virtualization overview

LX158.0

### Notes:

Virtualization is a hot topic today. It allows the use of more standardized environments, optimum use of resources and is one of the foundations for Cloud Computing. Different organizations have come up with different virtualization mechanisms.

The first, and oldest, form of virtualization is **hardware emulation**. This means that a certain piece of hardware (most often the **CPU** and peripherals) is emulated in software. This is typically done to develop and test software for a hardware platform that is not yet available, or is not suitable for software development and testing. Hardware emulation is the most flexible of all virtualization solutions but the disadvantage is that it is very, very slow compared to "native" hardware speeds.

**Virtualization** means that a hardware platform is partially emulated, but performance critical functions, such as the CPU instructions, are executed natively on the CPU. Obviously the CPU has to have some measure of support for this. The hardware that is emulated (typically the hard disk, network adapter and such) is an emulation of hardware that exists in real life. Because of this, virtualization allows you to run Operating Systems that were originally not intended to be virtualized.

**Paravirtualization** is very similar to virtualization. The main difference is that the hardware that is virtualized/emulated is not necessarily hardware that exists in real life. This makes paravirtualization a lot more efficient than virtualization. After all, in the real world devices like network adapters have to deal with interference, packet collisions, marginal data cables and such, which do not exist in the virtual world. So a paravirtual device and its associated device driver can be much simpler than a real-world virtualized device and its associated device driver.

The disadvantage of paravirtualization is that the operating system has to have support for the paravirtualized hardware. As the paravirtualized hardware is not available in the real world, there may not be an incentive to add this support to the OS, particularly not if the paravirtualization solution competes or conflicts with the OS manufacturers own virtualization solution.

**Partitioning** means that the systems hardware is split into independent partitions, each holding its own operating system. These partitions are completely independent and isolated from each other. With **Physical partitioning**, the partitions are created on an electrical level. With **Logical Partitioning (LPAR)** a hypervisor creates the partitions and handles hardware isolation. LPAR technology can also be combined with virtualization if required. In IBMs Power hardware, for instance, you can use dedicated or shared CPUs, dedicated or shared memory, and dedicated or shared I/O devices.

In the Intel/AMD world, partitioning is a solution that is only rarely applied. The solutions covered in this unit are all based on virtualization or paravirtualization.

***Instructor notes:***

**Purpose** — Introduce virtualization.

**Details** —

**Additional information** — Note that this sheet is not specific to Linux, but covers virtualization/partitioning solutions in general

**Transition statement** — Let's look at the virtualization solutions that are available for Linux.

# Virtualization solutions for Linux

- QEMU (Quick Emulator)
  - Emulates i386, x86\_64, ppc64 and others on various hardware
  - Relatively slow
- Xen
  - Paravirtualization and Full Virtualization
    - Full Virtualization requires KVM
  - Start the Xen Hypervisor first, then run Linux in "Domain 0"
  - Other OSs run in "Domain U"
- QEMU-KVM
  - Virtualization based on QEMU framework and KVM
- VMWare, MS Virtual Server and others
  - Commercial solutions

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-3. Virtualization solutions for Linux

LX158.0

## Notes:

There are several virtualization solutions for Linux available.

**QEMU** (Quick EMULATOR) is a software package that includes emulators for a large number of CPU architectures. As it is an emulator, it is relatively slow. Performance improvements have been made by executing CPU instructions natively on the CPU, but this only works if you emulate a platform on that particular platform (i386 on 386, x86\_64 on x86\_64, ppc64 on ppc64 and so forth).

QEMU is not a production-quality solution and is not supported out-of-the-box by enterprise distributions. However, as we will see later, QEMU-KVM is built on top of the QEMU framework. That is the reason we will cover QEMU briefly.

**Xen** was the first production-ready virtualization solution for Linux. Initially it supported paravirtualization only. This meant that effectively it could only run Linux as guest OS. Later it received full virtualization capabilities as well, meaning that for instance Microsoft Windows variants could also run under Xen. This requires KVM, however.

**QEMU-KVM** was intended to create a lightweight virtualization solution for Linux that would use the KVM feature. It was built as a separate project using the QEMU framework. Work is underway to integrate this back into the main QEMU development project.

Furthermore, several commercial solutions for virtualization on Linux exist. VMware is the oldest of these, with a full suite of tools that support everything from a single image on a desktop (VMware Player) to full datacenter virtualization. But other companies have virtualization products as well, such as Microsoft Virtual Server.

***Instructor notes:***

**Purpose** — Discuss virtualization solutions for Linux.

**Details** —

**Additional information** —

**Transition statement** — Okay, we've mentioned this KVM thing a few times already. Let's look at what that is.

## Kernel virtualization module

- Linux kernel module to make use of the virtualization capabilities of modern x86\_64 processors
  - Intel: Needs VT-x capability
  - AMD: Needs AMD-V capability
- Delivers near-native speeds
- To view processor support:
  - [http://www.linux-kvm.org/page/Processor\\_support](http://www.linux-kvm.org/page/Processor_support)
  - `cat /proc/cpuinfo`, look for **vmx** (Intel) or **svm** (AMD)
  - `lsmod`; check for **kvm** and **kvm-intel** or **kvm-amd**
- KVM allows "nesting" of virtual machines
  - Not fully supported yet
- KVM supports live migration of virtual machines

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-4. Kernel virtualization module

LX158.0

### Notes:

Around 2005, both Intel and AMD, independently of each other, added virtualization extensions to their processors. Before that time, any type of x86 virtualization had to use extensive (software-based) rewrites of instructions in order for virtualization to work safely.

Intel called its feature VT-x, while AMD uses the term AMD-V. Once the chips with these features were able, and the chipsets supporting these chips were also modified to support these features, kernel developers started work on Linux support for these features.

Linux 2.6.20 (released in February 2007) was the first Linux kernel to support these features, in the form of the Kernel Virtualization Module (KVM).

Using these virtualization features means that extensive instruction rewrites are no longer needed. By far the majority of CPU instructions that a virtual machine may issue, can be executed natively in the hardware. Only a few instructions are trapped by the CPU, and forwarded to the "ring 0" kernel for virtualization. This means that any virtual machine that is virtualized using KVM, runs at near-native speeds. (Of course it still needs to compete for CPU time with other processes or virtual machines. Furthermore, other hardware such as

---

the hard disk or network adapter, may still be virtualized in software, and thus relatively slow.)

In order to use KVM, you need an Intel processor that supports VT-x, or an AMD processor that supports AMD-V. In general that means a processor produced after the mid-2000s, but even after that period, both Intel and AMD still manufactured chips without VT-x or AMD-V.

To verify your CPU supports VT-x or AMD-V, view the virtual file **/proc/cpuinfo**. If you see the **vmx** flag (Intel) or **svm** (AMD) flag, your processor will support KVM. Note however, that these virtualization features are by default disabled in your BIOS. You need to enable them, after which a cold shutdown/restart is required to activate them.

Alternatively, you can simply take a look at the **kvm** modules that were loaded. If the modules (**kvm** and **kvm-intel** on an Intel machine, and **kvm** and **kvm-amd** on an AMD machine) were loaded properly, then your system fully supports KVM.

KVM has another advantage. It allows "nesting" of virtual machines. This means you can run a virtual machine inside a virtual machine, which in turns runs on a physical machine. This can be particularly useful in large datacenters where you assign virtual machines to users. Users can then run virtualization inside their own virtual machines if necessary.

The current state of Linux support for nesting of virtual machines is not production-ready, unfortunately. This means the enterprise distributions do not support it.

Also, KVM allows live migration of virtual machines. Not all virtualization solutions support this though. As this unit is only an introduction to virtualization, and as it is unlikely that your classroom will have the necessary infrastructure for live migration, we will not explore this in this course.

For more information about X86 virtualization, look at  
[http://en.wikipedia.org/wiki/X86\\_virtualization](http://en.wikipedia.org/wiki/X86_virtualization).

**Instructor notes:**

**Purpose** — Discuss KVM

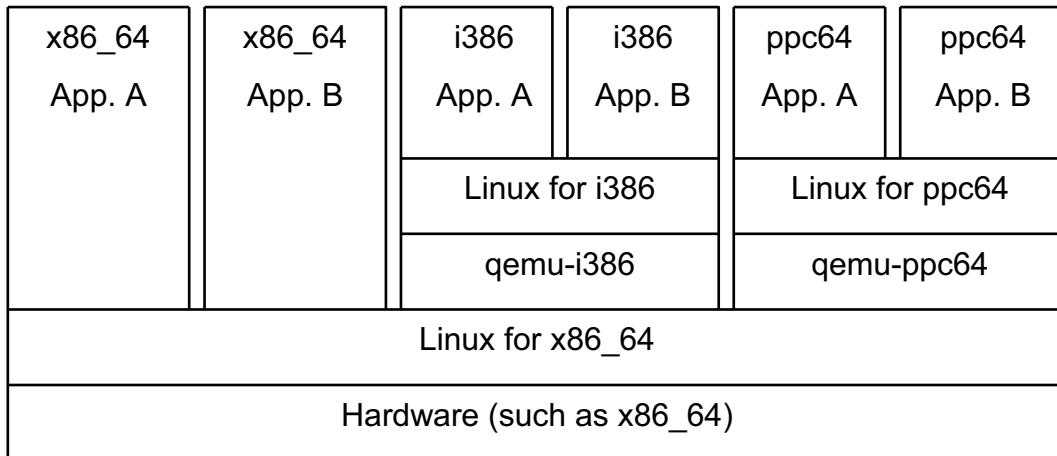
**Details** —

**Additional information** —

**Transition statement** — Okay, so KVM plays a very important role these days. We will meet KVM again in a few minutes. First, we want to introduce QEMU, because it's a good illustration on how virtualization works, and we will need the QEMU framework later on.

## Quick Emulator

- Full software emulator of various hardware platforms:
  - Intel/AMD x86, x86\_64
  - IBM PowerPC, IBM Power4+
  - Sparc, MIPS, ARM, ...



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
 US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-5. Quick Emulator

LX158.0

### Notes:

QEMU (Quick EMUlator) is a suite of emulators for various platforms. It can be used to emulate various Intel/AMD architectures (both 32 and 64 bit), the IBM Power architecture and various others.

QEMU is not included as default by enterprise distributions such as RHEL and SLES, but can be installed using third-party RPMs. Additionally, you can compile QEMU from source.

**Instructor notes:**

**Purpose** — Introduce QEMU

**Details** —

**Additional information** — Once again, QEMU is only introduced here because we need the framework later for QEMU-KVM.

**Transition statement** — Let's see how QEMU works.

## QEMU example

- `qemu-img create -f qcow2 vdisk.img 10G`
  - Creates a 10G disk image in (default) qcow2 format, called vdisk.img
  - QEMU also supports VMware and other disk formats
    - (No support for VMware split disks: convert with `vmware-vdiskmgr`)
- `qemu-system-x86_64 -hda vdisk.img`
  - Emulate an `x86_64` system, using `vdisk.img` as `hda`
- `qemu-system-ppc64 -hda vdisk.img`
  - Emulate an `PowerPC_64` system, using `vdisk.img` as `hda`
- `qemu-system-arch` runs as a regular process in Linux
  - Can be managed just like any other (`nice`, `kill`, ...)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
 US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-6. QEMU example

LX158.0

### Notes:

QEMU is really easy to configure.

The first step is to create a file in your filesystem that will act as the hard disk for the virtual machine. This is done using the **`qemu-img create`** command. The default QEMU format is the **qcow2** file format. This is a "sparse" format, which means that disk space will only be allocated once data is actually written to disk. QEMU also supports the use of partitions, logical volumes, LUNs and full disks, and it supports file formats of other virtualization solutions, including VMware. Note that there is no support for VMware "split disks", where a single VMware disk is split into multiple files, typically 2 GB or less to circumvent file system restrictions. In this case you need to use the **`vmware-vdiskmgr`** tool (part of the VMware suite) to convert the files into a single disk image file.

Once you have the disk configured, you can start the QEMU emulator, with the command **`qemu-system-architecture`**, where *architecture* is the name of the architecture you want to emulate. This process runs as a regular process in Linux, and can be managed like any other process.

There are several options for **`qemu-system-architecture`**, which we will see later.

***Instructor notes:***

**Purpose —** Show a QEMU example

**Details —**

**Additional information —**

**Transition statement —** Okay, enough about QEMU. Let's talk about Xen.

## Xen: Overview

- Originally a research project from the University of Cambridge, now developed by XenSource
- Open source (licensed under GPL)
- Included in many new Linux distributions
- Current version (V4.0)
- Citrix acquired XenSource in 2007
- Guests can run as PV (paravirtualized) guests and as HVM (Hardware Virtual Machine) guests
  - PV requires modifications to the guest.
  - HVM requires no modifications to the guest, but require VT-x/AMD-V support (processor feature) and KVM (kernel feature)



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-7. Xen: Overview

LX158.0

### Notes:

Xen is an open source software project that provides high-performance resource-managed virtualization on the x86 processor architecture. It allows multiple operating system instances to run concurrently on a single physical computer. Xen manages the computer's hardware resources so they are shared effectively among the operating system instances, called domains.

Open means open software.

Xen has grown dramatically over the past few years. With the release of v3.1 in May 2007 and the wide acceptance by key Linux distributors such as Novell and Red Hat, Xen is becoming a player in the arena of Enterprise Linux. Because it is an open source product, the cost factor becomes important as well.

In late 2007, XenSource was acquired by Citrix.

Xen initially only supported paravirtualization. Practically speaking this meant you could only use Linux guests, as other operating systems did not have the driver and CPU support to run in a paravirtualized environment.

When KVM support was added to Linux, Xen was able to use this feature as well. That means you can also use Hardware Virtual Machines (HVM) virtualization. This is full virtualization which allows you to run any x86\_64 operating system on top of Xen.

Xen will automatically determine whether paravirtualization (PV) or Hardware Virtual Machine (HVM) virtualization is required. In fact, in a lot of cases Xen will actually support a hybrid mode, where the CPU is virtualized using HVM, while devices such as network adapters are virtualized using PV. This typically offers the best performance, but does require device drivers for the PV devices.

**Instructor notes:**

**Purpose** — Introduce Xen

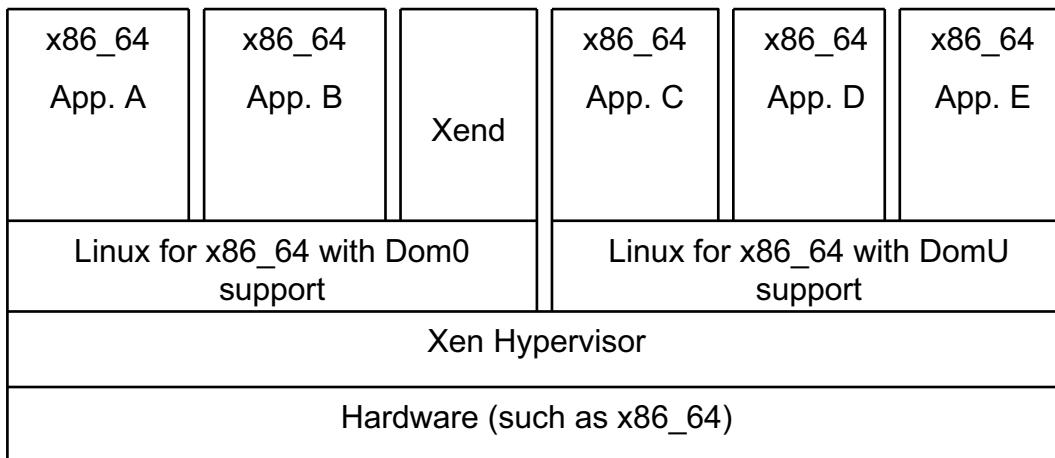
**Details** —

**Additional information** —

**Transition statement** — Let's look at the Xen architecture.

## Xen: Architecture

- Xen Hypervisor on "bare metal"
- Dom0 machine runs virtualization tasks and management
- DomU machines are paravirtualized or fully virtualized guests



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-8. Xen: Architecture

LX158.0

### Notes:

The visual shows the Xen architecture. Instead of booting a Linux kernel directly on the hardware (with the aid of a boot loader such as GRUB), we now boot the Xen Hypervisor directly onto the hardware (again with the aid of a boot loader). The Xen Hypervisor will then boot your Linux kernel in the "Domain Zero" ("Dom0") domain.

This Linux system is a normal Linux system that is capable of running a variety of applications, including server and desktop applications. It will also run the **xend** management daemon, and will run tasks on behalf of the other Xen domains.

Additional Xen domains, called "User Domains" or "DomU" domains are created as and when required. They will contain paravirtualized or fully virtualized guests. The Xen Hypervisor makes sure all domains are scheduled onto the CPU properly, and will perform virtualization of other hardware. If required, it will pass hardware requests from DomU domains to the Dom0 domain for execution. Because of this, it is still possible to use files inside the Dom0 domain, as disk images for a DomU system.

**Instructor notes:**

**Purpose** — Discuss the Xen architecture

**Details** —

**Additional information** —

**Transition statement** — Let's look at the installation of Xen.

## Xen: Installation

- Not included in all distributions
  - RHEL5, SLES11: yes
  - RHEL6: no
- When included, distributions will install all packages through a common bundle or component group
  - If not included, RPMs can be found elsewhere on the internet
- The main components are:
  - Xen Hypervisor
  - Xen-enabled kernel (Dom0 and/or DomU)
  - **xend** daemon
  - Xen utilities

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-9. Xen: Installation

LX158.0

### Notes:

The Xen packages are typically included in most modern distributions, but not automatically installed. Many distributions include a bundle or component group that eases the installation.

The components of Xen are:

- The Xen Hypervisor
- The Xen-enabled kernel, with support for Dom0 and/or DomU operation.
- The **xend** management daemon, which communicates with the Xen Hypervisor.
- Command line utilities such as **xm**.

Once the installation of the packages is complete, the system needs to be rebooted and the Xen kernel selected from the boot loader.

In addition to the Xen packages, you will also want to install the **libvirt** packages. We will discuss them later.

Red Hat has stopped supporting Xen in RHEL6, favoring QEMU-KVM instead. There are third-party RPMs available to support Xen on RHEL6, though.

**Instructor notes:**

**Purpose** — Discuss Xen installation

**Details** — SLES configures your grub.conf/menu.lst file automatically with the proper stanzas. On a Red Hat system, using the third-party RPMs, you need to do this yourself.

**Additional information** —

**Transition statement** — Let's see how we boot Xen.

## **Booting the Xen Hypervisor and Dom0 kernel**

- Verify Xen kernel listed in GRUB configuration:

```
title Xen
root (hd0,2)
kernel /boot/xen.gz
module /boot/vmlinuz-version-xen-dom0 root=...
module /boot/initrd-version-xen
```

- Reboot and select the Xen kernel from the bootloader menu
- Watch for the extra (XEN): messages on bootup
  - Allocating processor and memory resources
  - Setting up bridged network adapters
- Linux Kernel runs in Domain0
  - Only OS instance with direct hardware access
  - But other than that it is a normal OS

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-10. Booting the Xen Hypervisor and Dom0 kernel

LX158.0

### **Notes:**

On reboot, select the new Xen kernel from the bootloader menu (GRUB or LILO). As the kernel is initializing an all new set of messages will appear. They will be prefixed with (XEN:) and describe Xen detecting CPU and memory, as well as re-configuring the network adapters.

Once Xen is done, it will boot the kernel (together with an initrd or initramfs) in Dom0.

Depending on your distribution, Dom0 support may be included in the default kernel, or you may need to run a kernel that is specifically compiled with Dom0 support built-in.

***Instructor notes:***

**Purpose** — Discuss the Xen boot process

**Details** —

**Additional information** —

**Transition statement** — Let's see how we setup a DomU guest.

## Xen DomU configuration file

- Located in /etc/xen/vm directory
- Common values to edit (minimum)
  - **name**: Name of the client
  - **builder**: Type of virtualization ("linux" for PV, "hvm" for HVM)
  - **vcpus**: Number of virtual CPUs
  - **memory**: Domain's memory in MB
  - **disk**: Which partitions in domain map to which device
  - **vfb**: Virtual Framebuffer configuration
  - **vif**: Virtual (network) interface configuration
  - Boot loader configuration
- Virtual disk creation command:
  - **dd if=/dev/zero of=/xen/image.img bs=1G count=2**
  - **dd if=/dev/zero of=/xen/image.img bs=1k seek=2048K count=1**
  - Can also use LVs, disk partitions, LUNs or whole disks (faster)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-11. Xen DomU configuration file

LX158.0

### Notes:

Guest domains must have a Xen configuration file created so Xen knows how to boot them. These files are all stored in the /etc/xen/vm subdirectory.

Administrators can adjust the values as necessary to reflect their own system and network configuration. Details can be obtained from the *Xen User Manual*.

Furthermore, for each DomU domain you will need to assign a physical disk, LV or LUN, or create a file to act as a virtual disk. Creating virtual disks is done with the **dd** command.

Xen itself does not support sparse disks, but relies on the underlying OS (particularly the filesystem) to support that feature. In Linux, you can create a sparse file with the following command:

**dd if=/dev/zero of=/xen/image.img bs=1k seek=2048k count=1**

This will create a "sparse" file, with 2048K 1KB blocks (2 GB) empty, unallocated space, followed by a 1k block of zeroes. So this file will initially only consume 1 KB in disk size, and the other disk blocks are only allocated once data is written to it. (Note that if you do an **ls -l** on this file, its size will show up as 2G+1K, but it will only consume one disk block for

data, plus a handful of disk blocks for the various indirect blocks. The actual amount of disk blocks (including blocks for metadata such as indirect blocks) consumed by a file can be viewed with **ls -lks** or **du -k**.

In production environments, using logical volumes, partitions, full disks or LUNs is considerably faster than using files or sparse files, as files tend to get fragmented over a filesystem.

***Instructor notes:***

**Purpose** — Discuss setting up a DomU domain.

**Details** —

**Additional information** —

**Transition statement** — Let's look at the content of a DomU configuration file.

## Xen configuration file: PV example

```
# cat /etc/xen/vm/client1
name = "client1"
builder = "linux"
vcpus = 1
memory = 512
disk = [ 'file:/images/client1-hda.img,hda,w' ]
vfb = [ 'type=vnc' ]
vif = [ '' ]
bootloader = "/usr/bin/pygrub"
#kernel = "/boot/vmlinuz-xen"
#extra = "root=/dev/hda2 resume=/dev/hda1 splash=silent showopts"
#ramdisk = "/boot/initrd-xen"
# For installation:
#kernel = "/mnt/sles11/boot/i386/vmlinuz-xen"
#extra = "initrd=initrd splash=silent showopts"
#ramdisk = "/mnt/sles11/boot/i386/initrd-xen"
```

- You cannot run GRUB or syslinux in a PV environment
  - Xen needs to load the kernel and initrd from the Dom0 domain
  - **pygrub** loads the kernel and initrd file from the disk image, and then ensures Xen loads it before starting the domain
  - For installation, specify **kernel** and **ramdisk** manually from inst. media

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-12. Xen configuration file: PV example

LX158.0

### Notes:

A sample configuration file for a paravirtualized domain is shown in the visual.

The file identifies the name of the domain, the type of virtualization (builder = "linux" means a Linux paravirtualized domain), the number of virtual CPUs, the amount of memory in megabytes, the virtual disk and the virtual framebuffer.

Furthermore, the virtual (network) interface is the default configuration, which is a bridged configuration to the primary network adapter.

The last few lines configure the bootloader configuration. In a paravirtualized domain, this is one of the most complex issues to setup. The reason for this is that the GRUB bootloader, and the bootloaders that are part of the syslinux suite (including isolinux, the bootloader that's used to boot from CD/DVD) cannot be used in a paravirtualized environment.

The way to solve this issue is to store the kernel and initrd files in the Dom0 domain, and letting Xen load these files into memory before the domain is started. You achieve this with the "kernel=" and "ramdisk=" configuration lines.

Maintaining a kernel and initrd file in the Dom0 domain can be a problem however. If you update the operating system running inside a DomU domain, that update process will update the kernel and initrd files inside the DomU domain, but not the files stored in the Dom0 domain. That means that after each update or upgrade, you might need to copy over those files to the Dom0 domain and modify the Xen configuration file. This will quickly become tedious, especially if the Dom0 and DomU domains are managed by different people or groups.

As an alternative, the **pygrub** boot loader has been developed. This is a Python script that runs before Xen starts the domain. It will extract the kernel and initrd files from the disk image file, based on the grub.conf/menu.lst file in the disk image file. It will then feed this kernel and initrd to Xen for loading into the domain. It will also extract any kernel options from the grub.conf/menu.lst file and feed these to Xen as an 'extra' line. When pygrub has finished, Xen will start the domain. With this pygrub boot loader you do not need an external kernel and initrd anymore, making the upgrade/update process transparent to the DomU maintainer.

The pygrub boot loader will not work for the installation though. So for the initial boot of the installation, you will need to specify the kernel and initrd files manually. These files can be found on the installation media. Using the 'extra' line you can supply kernel options. This allows you to start an AutoYaST installation, for instance.

***Instructor notes:***

**Purpose** — Show a sample PV configuration file.

**Details** —

**Additional information** —

**Transition statement** — In contrast, let's look at an HVM example too.

## Xen configuration file: HVM example

```
# cat /etc/xen/vm/client1
name = "client1"
builder = "hvm"
kernel = "/usr/lib/xen/boot/hvmloader"
vcpus = 1
memory = 512
disk = [ 'file:/images/client1-hda.img,hda,w',
          'file:/images/sles11.iso,hdc:cdrom,r' ]
vfb = [ 'type=vnc' ]
vif = [ '' ]
boot="c"
# For installation:
#boot="d"
```

- HVM requires KVM support

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-13. Xen configuration file: HVM example

LX158.0

### Notes:

The visual shows a configuration file for a Hardware Virtual Machine setup. As said before, HVM is required to run an operating system that does not support paravirtualization, such as Microsoft Windows.

To run Xen domains in HVM mode requires the KVM kernel module, and requires CPU support (VT-x or AMD-V, as appropriate).

In case of an HVM domain, the boot process needs to imitate the boot process of a physical machine as closely as possible. This means we need to load a BIOS that performs basic configuration of the boot devices, then loads the first sector of the boot disk and executes it. This first task is accomplished by the **hvmloader** kernel. Depending on the exact version of Xen, **hvmloader** either uses the RomBIOS of the Bochs project (an x86 emulator), or the SeaBIOS of the QEMU project.

Because both GRUB and syslinux are able to run in an HVM environment, we do not need to worry about special procedures to load the kernel and initrd files. We can simply tell the BIOS to load the bootloader from the hard disk or CD/DVD. For the installation, we specify 'boot="d"' (using the MS-DOS "D:" notation for the CD/DVD) to boot from CD/DVD. Once

the operating system is installed, we specify 'boot="c"' to boot from hard disk. (If you are positive that your disk image file is empty, you can also specify 'boot="cd"', which will attempt to boot from C: first. If that does not succeed, it will boot from D:. This means that you will not need to reconfigure your domain halfway through the installation.)

In an HVM environment you would normally virtualize all devices. However, it is possible to specify that the HVM environment should include paravirtualized devices as well. Obviously you do need special device drivers for these devices, which may not be included in the OS but may need to be installed separately. But the use of paravirtualized devices can be considerably faster than the virtualization/emulation of physical devices.

**Instructor notes:**

**Purpose** — Show an HVM example.

**Details —**

**Additional information** — In the earliest versions of Xen it was also possible to run in a software-based full virtualization mode. This is similar to HVM in that a full machine, including the CPU, would be virtualized and allowed you to use, for instance, Microsoft Windows. But because so much hardware needed to be emulated, this was incredibly slow, similar to QEMU emulation. This mode is no longer supported. The only way to run non-paravirtualized operating systems such as Microsoft Windows under Xen today is to use HVM, which in turn requires KVM and a CPU that supports KVM.

**Transition statement** — Let's look at a few more bits and pieces that are part of the Xen infrastructure.

## Xen domain management (1 of 2)

- Done with the **xend** node control daemon
- Written in Python
- Automatically started when dom0 is booted
- Supports HTTP interface; normally disabled
- Can be started on the command line
  - `# xend start | stop | restart | status`
- SLES10, SLES11, RHEL5 include SysV init scripts for **xend**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-14. Xen domain management (1 of 2)

LX158.0

### Notes:

Xen management is accomplished by daemon and by command line. These tools can be accessed through command line and GUI management.

The **xen** daemon (**xend**) performs system management functions related to virtual machines. It forms a central point of control for a machine and is normally controlled via the **xm** command line tool. **xend** must be running in order to start and manage virtual machines.

The **xend** daemon also incorporates a web interface, but this is by default disabled. To enable it, modify the `/etc/xen/xend-config.sxp` file to set the `xend-http-server` option to `yes`. There are several other http options that control the behavior of the HTTP server.

It is necessary for **xend** to run as root because it needs access to privileged system management functions.

**xend** logs events to `/var/log/xend.log` and `/var/log/xend-debug.log`.

***Instructor notes:***

**Purpose —** Discuss **xend**.

**Details —**

**Additional information —**

**Transition statement —** Let's look at the command for Xen management too.

## Xen domain management (2 of 2)

- Primary tool: **xm** command
- Common sub-commands:
  - **create**: Create (start) a domain
  - **console**: Attach to console
  - **shutdown**: Stop a domain (gracefully)
  - **destroy**: Remove a domain
  - **list**: Display all known domains and states
  - **save**: Save state and config of a domain
  - **restore**: Restore domain
- Autostart domains by creating symlink in /etc/xen/auto to configuration file
  - Domains will be started by the SysV **xendomains** init script

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-15. Xen domain management (2 of 2)

LX158.0

### Notes:

#### xm command

The **xm** command is the primary tool for managing Xen from the console. The general format of an **xm** command line is:

```
# xm command [switches] [arguments] [variables]
```

The available switches and arguments are dependent on the command chosen. The variables can be set using declarations of the form variable=value and command line declarations override any of the values in the configuration file being used, including the standard variables described above and any custom variables.

To autostart domains when the system starts, you can symlink their configuration files into the /etc/xen/auto directory.

**Instructor notes:**

**Purpose** — Introduce **xm**.

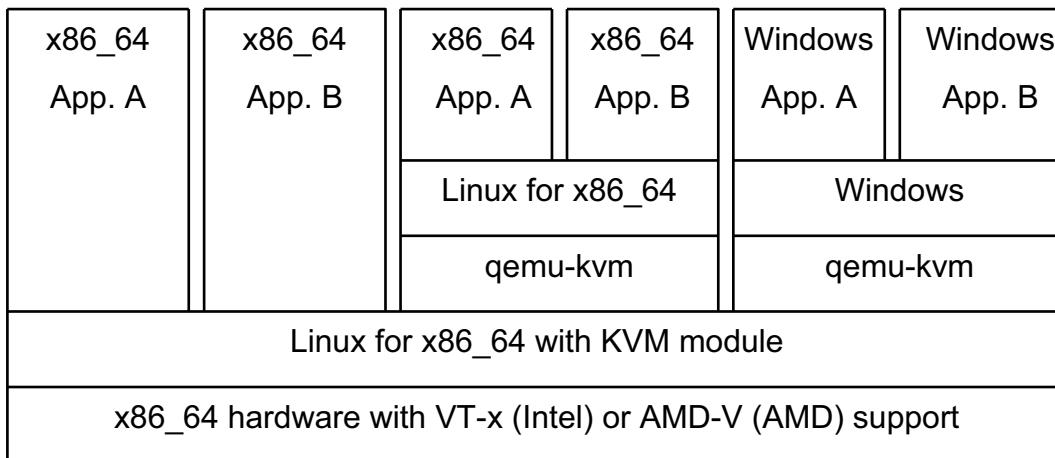
**Details** —

**Additional information** — We are not going to cover **xm** in great detail. In most environments, Xen will be managed using the Virtual Machine Manager which we will see in a few minutes.

**Transition statement** — Okay, that's it for Xen. We will now look at QEMU-KVM.

## QEMU-KVM

- Modification to QEMU framework to use the KVM kernel/processor feature
- Supported in SLES11 in addition to Xen
- Only supported virtualization solution in RHEL6



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-16. QEMU-KVM

LX158.0

### Notes:

QEMU-KVM is a modification of the QEMU framework, so that QEMU can use the KVM feature of the kernel, and the VT-x/AMD-V feature of your processor. This makes it possible to run virtual machines under QEMU at near-native speeds.

As QEMU-KVM relies on the VT-x/AMD-V feature, it can only be used on the x86\_64 architecture. Work is underway to support KVM on the IBM Power architecture as well, but this is not production-ready yet.

QEMU-KVM works very similar to QEMU. The main difference is that QEMU-KVM does not emulate the CPU instructions in software, but rather executes them on the CPU directly, relying on the virtualization features of the CPU to "trap" any dangerous instructions. These instructions will then be virtualized as required.

Because QEMU-KVM does full virtualization, it is possible to run other operating systems, including Microsoft Windows, unmodified under QEMU-KVM.

***Instructor notes:***

**Purpose —** Introduce QEMU-KVM

**Details —**

**Additional information —**

**Transition statement —** Let's see how we start QEMU-KVM.

## Starting QEMU-KVM

- `qemu-img create -f qcow2 vdisk.img 10G`
  - Creates a 10G disk image in (default) qcow2 format, called vdisk.img
- `qemu-kvm -hda vdisk.img`
  - Virtualize an x86\_64 system with KVM, using vdisk.img as hda
- Useful qemu-kvm options:
  - smp <n>**: Set number of CPUs
  - m <memory>**: Memory in MB
  - fda <file>, -hda <file>, -cdrom <file>**: fd/disk/cd image
  - drive <options>**: Configure direct I/O to devices
  - net <options>**: Setup networking
  - vnc <options>**: Configure VNC access to console
  - boot <disk>**: Boot from <disk>

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-17. Starting QEMU-KVM

LX158.0

### Notes:

Starting QEMU-KVM is very similar to starting QEMU. You first need to create a disk image with **qemu-img**, or make sure a disk, disk partition, logical volume or LUN is available. Then, start **qemu-kvm** with the appropriate options.

The visual shows the most commonly used options, but there are far more options available to configure your virtual machine exactly the way you want it.

**Instructor notes:**

**Purpose** — Discuss starting QEMU-KVM

**Details** —

**Additional information** — Once again, don't spend a lot of time discussing the options. Refer the students to the manual page. In real life, students will use libvirt and the virtual machine manager to create their QEMU-KVM virtual machines.

**Transition statement** — Let's last take a look at libvirt.

## Libvirt

- Library and tools to support Xen and QEMU-KVM virtualization
  - Client/Server design with secured (encrypted) connections
- Daemon (**libvirtd**)
- Graphical User Interface (**virt-manager**)
- Command Line Interface (**virsh**)
- Application Programming Interface
- VM Guest Setup Wizard
- Virtual bridge (**virbr0**) with NAT to outside world

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-18. Libvirt

LX158.0

### Notes:

Both Xen and QEMU-KVM have the problem that they are, to a large extent, managed by command line commands that act on the local system only. They may or may not have a wizard or other tools to setup virtual machines, and they may or may not have tools for remote management.

**Libvirt** is an effort to eliminate these deficiencies. It is intended as a solution-agnostic interface for virtualization. At this time, it supports both Xen and QEMU-KVM virtualization. Furthermore, it supports secured (encrypted) communications in its client-server design, which means you can also use it to manage cloud infrastructures.

**Libvirt** contains several parts, all of which are built on top of a library of virtualization functions:

- The **libvirtd** daemon needs to run on each system that hosts virtual machines and wants to participate in **libvirt** management. It listens to network connections, accepts connections from **libvirt** clients such as the Virtual Machine Manager, and executes the commands. It also provides status and performance information to the clients.

- The graphical Virtual Machine Manager (**virt-manager**) connects to the **libvirtd** daemon. It offers a GUI in which you can create, start, stop, destroy and manage virtual machines.
- A command line client, **virsh** is also available. It performs the same functions as **virt-manager** but does so from the command line. As such, it can be used in shell scripts, or in environments where a GUI is not available.
- **libvirt** also provides an Application Programming Interface that allows other developers to integrate **libvirt** functionality into their own programs.
- Last, **libvirt** includes a wizard that will help you setup virtual machines. This wizard supports both Xen and QEMU-KVM based machines.

The **libvirt daemon**, once it starts, by default creates an internal network, called virbr0, to which all virtual machines can be attached. The libvirt host will have IP address 192.168.122.1 by default, and will perform Network Address Translation (NAT) to the outside world through the use of iptables. All virtual machines need to be given an IP address in this IP range too, either statically or by configuring a DHCP server on this virbr0 interface, if they require outside connectivity through virbr0.

It is very common to setup a Linux bridge to the outside world using the **brctl** tool, and connect the virtual machines to this bridge instead of virbr0. Setting up bridged networking is covered in appendix C of this course.

***Instructor notes:***

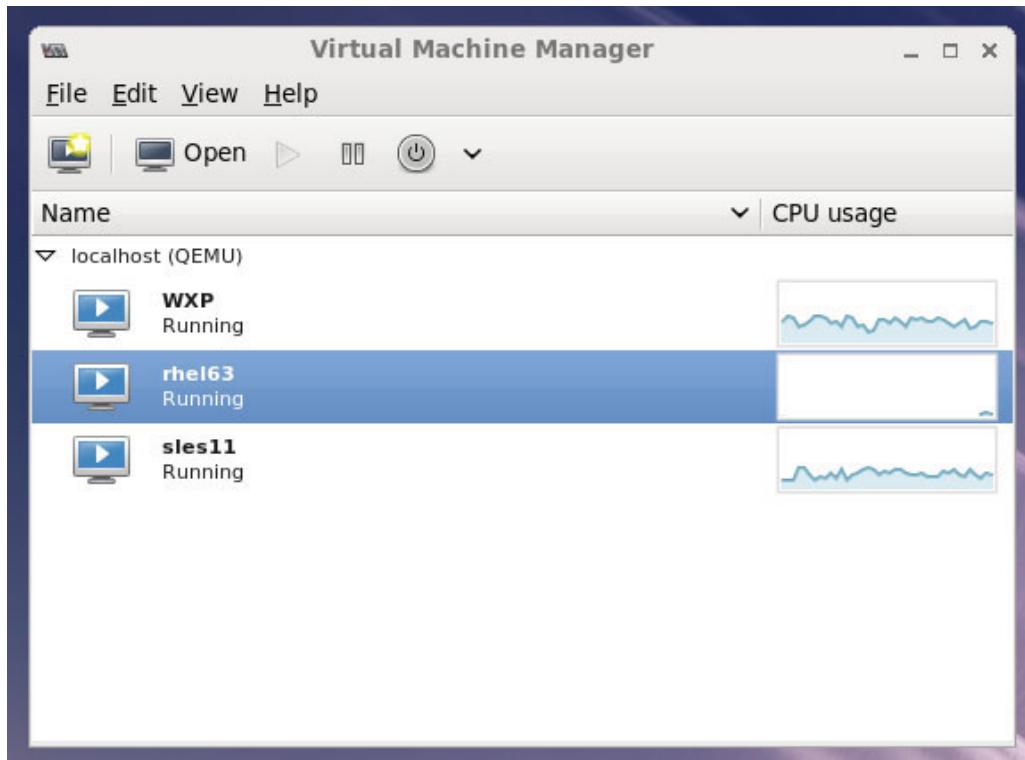
**Purpose —** Introduce libvirt

**Details —**

**Additional information —**

**Transition statement —** Let's see what it looks like.

# Virtual Machine Manager screenshot



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-19. Virtual Machine Manager screenshot

LX158.0

## Notes:

The visual shows a screenshot of the Virtual Machine Manager in action. Three virtual machines have been defined on the local system, all three running under QEMU-KVM.

You can use the GUI to start, stop and modify these virtual machines, create new virtual machines or destroy existing ones.

***Instructor notes:***

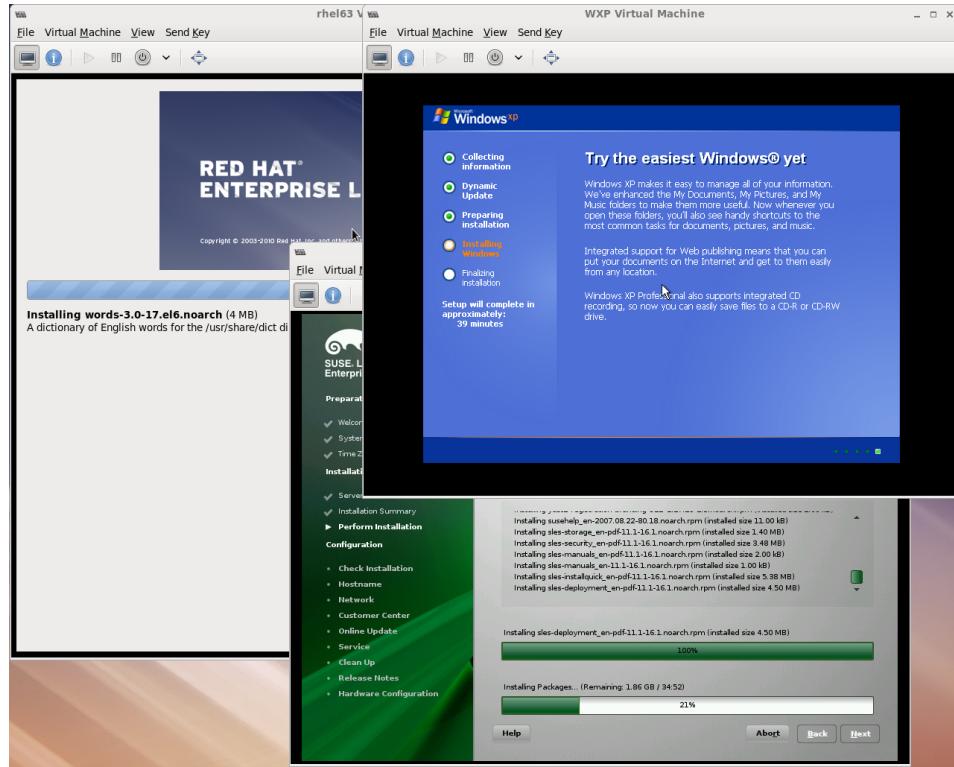
**Purpose —** Demonstrate VMM.

**Details —**

**Additional information —**

**Transition statement —** Okay, let's see if this works.

# Virtualization with KVM example



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-20. Virtualization with KVM example

LX158.0

## Notes:

Here's a typical example of the whole stack in action. This particular system is a Red Hat Enterprise Linux 6.3 system running on a 64-bit Intel processor with VT-x enabled.

Using QEMU-KVM we created three different virtual machines, and are currently performing installations of RHEL6.3, SLES11 and Windows XP.

***Instructor notes:***

**Purpose** — Show virtualization in action.

**Details** —

**Additional information** —

**Transition statement** — That's it. You're going to do this in the exercises as well.

## Checkpoint

1. Which open source solutions for virtualization are supported in Linux?
2. What is KVM?
3. What is libvirt?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-21. Checkpoint

LX158.0

### Notes:

**Instructor notes:**

Purpose —

Details —

## Checkpoint solutions

---

1. Which open source solutions for virtualization are supported in Linux?

The answer is Xen and QEMU-KVM.

2. What is KVM?

The answer is Kernel Virtualization Module - the module that provides support for the VT-x (Intel) or AMD-V (AMD) processor feature. This is a requirement for full virtualization (HVM).

3. What is libvirt?

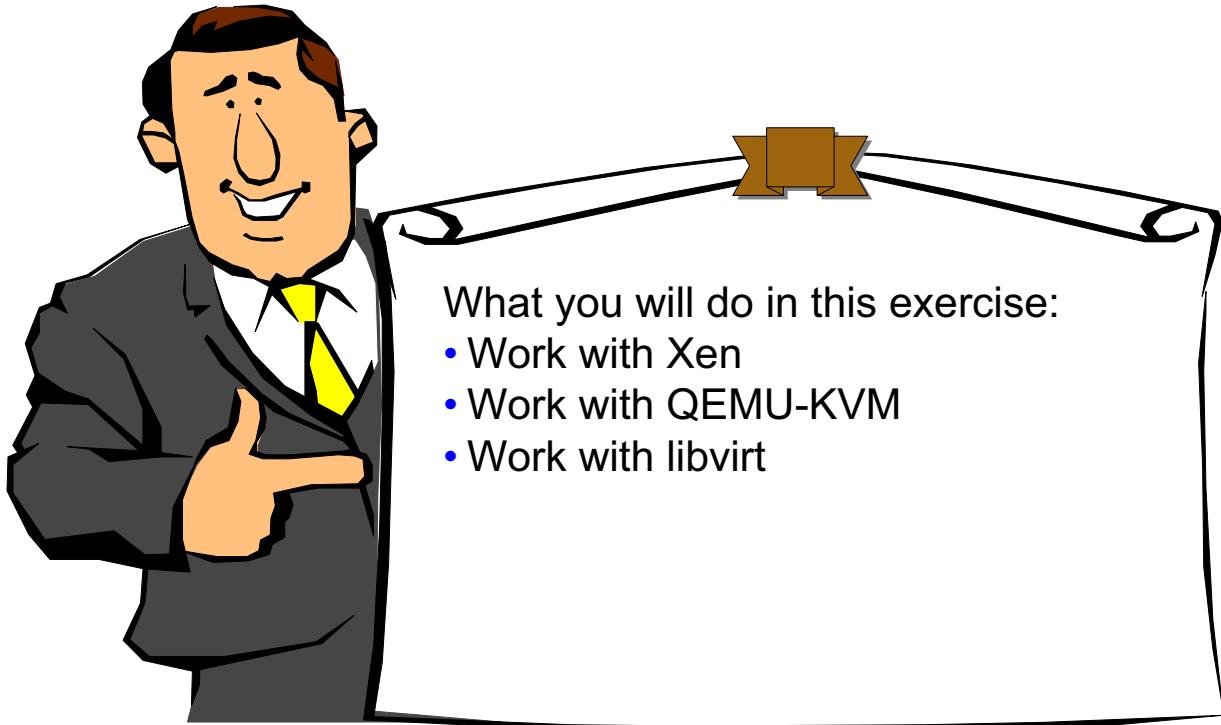
The answer is a library that provides a solution-agnostic API, daemon, GUI, CLI and tools for virtualization on Linux.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —**

**Transition statement —**

## Exercise: Virtualization



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-22. Exercise: Virtualization

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Unit summary

Having completed this unit, you should be able to:

- List the different virtualization options in Linux
- Discuss the KVM kernel feature
- Discuss QEMU emulation
- Configure Xen
- Configure QEMU-KVM
- Discuss libvirt and use the Virtual Machine Manager

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 13-23. Unit summary

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

# Unit 14. Logging and troubleshooting

## Estimated time

00:45

## What this unit is about

This unit teaches you the basics of troubleshooting a Linux system.

## What you should be able to do

After completing this unit, you should be able to:

- Describe logging concepts
- Configure the **syslog**, **syslog-ng** and **rsyslog** daemons
- Use the **logger** program
- Use the **logrotate** program
- Discuss log analysis
- Perform basic problem determination
- Utilize Rescue Mode to perform system recovery

## How you will check your progress

Accountability:

- Checkpoint questions
- Machine exercises

## Unit objectives

---

After completing this unit, you should be able to:

- Describe logging concepts
- Configure the syslog, syslog-ng and rsyslog daemons
- Use the logger program
- Use the logrotate program
- Discuss log analysis
- Perform basic problem determination
- Utilize Rescue Mode to perform system recovery

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 14-1. Unit objectives

LX158.0

### Notes:

## 14.1.Logging

### Instructor topic introduction

**What students will do —**

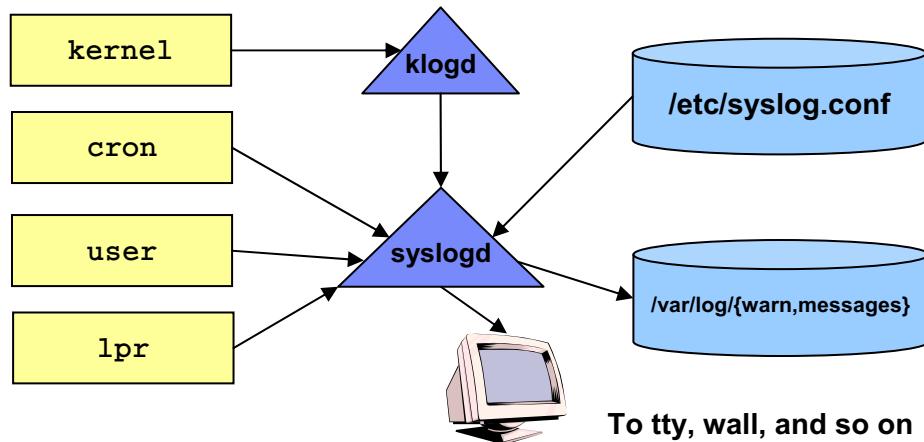
**How students will do it —**

**What students will learn —**

**How this will help students on their job —**

# Logging concepts

- Various daemons generate log information
- All log items are sent to the **system log** daemon
  - Tagged with facility and priority
  - Through UDP/IP or UNIX socket
- **system log daemon** decides what to do, based on its configuration file



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-2. Logging concepts

LX158.0

## Notes:

### Introduction

Various daemons generate information which might be of interest. Since these daemons do not run as foreground processes, they cannot print that information to the screen. Because of that, and because you might want to keep this information for later reference, this logging information is usually stored on disk.

In the early days of UNIX, every program wrote this information to its own logging file. This worked quite well for the programmer of the daemon but was the system administrator's nightmare:

- Every log file had its own syntax
- Every daemon had its own way of selecting which items to log
- It was nearly impossible to do other things with the log items, like sending them to another host or displaying things on the console.

## Logging daemons

Because of these limitations, most daemons (but not all!) make use of a facility called the syslog daemon. The concept is very simple:

- Every daemon that wants something to be logged creates the log message. It then tags this message with a facility (where it comes from) and a priority (how important is the message). It then sends this item to the syslog daemon, either through UDP/IP or through a UNIX socket (a special file in the file system).
- The syslogd daemon receives the message and decides, based on the facility and priority fields, what to do with the message. This can be one or more of the following actions:
  - Discard it
  - Send it to the syslogd on another system
  - Add it to a file on disk
  - Write it to a user (similar to the write command)
  - Write it to all users (similar to the wall command)

The syslogd daemon is configured through the /etc/syslogd.conf file.

There is one program that does not log through the syslog daemon directly, and that is the kernel itself. For technical reasons the kernel developers chose not to include the syslog system calls in the kernel itself but used a simplified scheme to do kernel logging. The kernel log daemon (klogd) receives the kernel log input, converts it into syslog format, and logs it to the syslogd daemon. It is then handled as normal syslog input. The klogd daemon is usually started and stopped together with the syslogd daemon.

**Instructor notes:**

**Purpose** — Introduce the syslogd daemon.

**Details** —

**Additional information** — The klogd also performs address to symbolic name translation through the System.map file. See the kernel compile unit for more information.

**Transition statement** — Let's look at those facilities and priorities in a little more detail.

## Facilities and priorities

- Each log item is tagged with a facility and a priority.
- Facility identifies the source:
  - auth
  - cron
  - kern
  - lpr
  - ...
- Priority identifies the importance:
  - panic
  - crit
  - warn
  - info
  - debug
  - ...
- For a complete list, see manual page

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 14-3. Facilities and priorities

LX158.0

### **Notes:**

#### **Introduction**

In addition to the message itself, each log message is tagged with a "facility" and a "priority".

#### **Facilities**

The facility defines the source of the message. The following facilities are defined:

- auth (authentication)
- auth-priv (authentication privileged; items logged here might contain sensitive information such as unencrypted passwords)
- cron (scheduling)
- daemon (any daemon)
- ftp (ftp daemon)
- kern (kernel messages)

- lpr (printing subsystem)
- mail (mail subsystem)
- news (news subsystem)
- security (same as auth; should no longer be used)
- syslog (the syslog daemon itself)
- user (user messages)
- uucp (UNIX to UNIX copy)
- local0 through local7 (for custom applications)

## Priorities

The priority defines the importance of the message. The following priorities are defined:

- emerg (wake the whole staff; break out the emergency handbooks)
- panic (same as emerg; should no longer be used)
- alert (alert the sysadmin)
- crit (something is failing)
- err (something is going wrong, but it is probably not very serious)
- error (same as err; should no longer be used)
- warning (something might go wrong)
- warn (same as warning; should no longer be used)
- notice (something to keep an eye on)
- info (general information)
- debug (debugging information; should normally be discarded)

The priority is only an indication of the seriousness of the message. If you have a Linux server with two applications on it (example: a mission-critical Dynamic Host Configuration Protocol (DHCP) server and a mail server which is only used to send statistic information twice a day), you will probably pay more attention to a warning from the DHCP server than to a panic of the mail server.

***Instructor notes:***

**Purpose** — Talk some more about facilities and priorities.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Now that we know what tags can be used on the messages, let's see how to handle them.

# Traditional syslogd configuration

- /etc/syslog.conf:

```

kern.warn                                /dev/ttys10
*.info;mail.none;authpriv.none;cron.none   /var/log/messages
mail.*                                     /var/log/mail
authpriv.*                                 /var/log/secure
cron.*                                     /var/log/cron
*.emerg                                    *
*.emerg                                    @sysadmin.acme.com

```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
 US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-4. Traditional syslogd configuration

LX158.0

## Notes:

### Introduction

The traditional UNIX system logger, first found in BSD UNIX, is called **syslogd**. It is configured through the /etc/syslog.conf file.

The file above is an example configuration file. Each line of the file contains two fields: the selector and the action field.

The selector field determines for which messages this action is valid. This is indicated by specifying **facility.priority**, which means that the action is valid for all log messages from **facility** with priority **priority** or higher (if you specify facility.=priority, only the specified priority matches). Multiple selectors might be specified on one line, as long as they are separated by a semicolon and do not contain any spaces. In addition to that, the wildcard \* can be used, which matches all facilities or priorities.

### /etc/syslog.conf fields

The action field determines what to do with the log items that match. There are several possibilities:

- Append it to a file, in which case the action is the file name. You need to specify the full pathname of the file, starting with a /. It is possible to specify special files as well, like /dev/console.
- Send it to someone by using the write command. In this case, the action is the username of the recipient. Multiple recipients can be specified, separated by a comma.
- Send it to everyone on the system using wall. In this case the action is a \*.
- Send it to the syslogd daemon on another system. In this case the action is a @, followed by the hostname of the receiving system.

## Working with remote systems

When sending the message to another system, the selection criteria from that /etc/syslog.conf file are applied too. The log items are sent over the network unencrypted. If your log messages contain privileged information, such as plain-text passwords, they might be intercepted.

In order to receive log messages on this other system, you need to allow incoming UDP traffic on port 514, and you need to configure the syslog daemon for incoming messages through this port. This is done by starting the syslog daemon with the -r option. You can typically enable this in the startup configuration file /etc/sysconfig/syslog, which is read by the startup script /etc/init.d/syslog.

### **Instructor notes:**

**Purpose** — Show the syntax of the /etc/syslog.conf file.

**Details** — Discuss student notes.

**Additional information** — Walk through the /etc/syslog.conf file, explaining every line.

**Transition statement** — The traditional syslog daemon has a number of limitations. That's why different people have developed more advanced logging daemons. Two examples of these are Syslog-NG and Rsyslog. Let's take a look at Syslog-NG first.

## Syslog-NG

- Syslog-Next Generation
- Improvements over traditional BSD syslog:
  - Content-based filtering in addition to facility/priority
  - TCP transport
  - Millisecond granularity in timestamps
  - Message tagging with host name of relay
  - TLS encryption
  - RFC 5424 support
- Used in SLES11
- Configuration file: /etc/syslog-ng/syslog-ng.conf

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-5. Syslog-NG

LX158.0

### Notes:

Syslog-NG (Syslog-Next Generation) is a further development of the traditional UNIX syslogd. It has a number of improvements:

- It allows filtering based on message content, in addition to facility/priority
- It supports the use of TCP/IP, in addition to UDP/IP, for remote logging
- It supports millisecond granularity in timestamp. This is very convenient when debugging message communication in clusters, for instance
- It supports message tagging with the name of the relay. This way, you can see where a message originated and which path it took before it ended up on the final log server.
- It supports encryption by the use of the TLS protocol
- It supports RFC 5424, which is a fairly recent Internet standard that aims to correct various deficiencies in the original document (RFC 3164) that described the syslogd standard, and adds various enhancements.

Syslog-NG is the syslog daemon that is used in SLES11. The configuration file of Syslog-NG is /etc/syslog-ng/syslog-ng.conf

**Instructor notes:**

**Purpose** — Introduce Syslog-NG

**Details** —

**Additional information** —

**Transition statement** — Let's take a look at that configuration file.

## /etc/syslog-ng/syslog-ng.conf example

```

source src { internal(); unix-dgram("/dev/log"); udp( ip("0.0.0.0")
port(514) ); };

filter f_console { level(warn) and facility(kern) };
filter f_mail { facility(mail) };
filter f_messages { not facility(mail) };

destination console { pipe("/dev/tty10" owner(-1) group(-1) perm(-1) )};
destination mail { file("/var/log/mail") };
destination messages { file("/var/log/messages") };

log { source(src); filter(f_console); destination(console); };
log { source(src); filter(f_mail); destination(mail); };
log { source(src); filter(f_messages); destination(messages); };

```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
 US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-6. /etc/syslog-ng/syslog-ng.conf example

LX158.0

### Notes:

The visual shows an example syslog-ng.conf file. In the file, you see a number of things being defined.

- The first definition defines a source "src". This source definition is applicable to anything that is a Syslog-NG-internal message, a message arriving on the UNIX socket /dev/log, and a message arriving on the UDP port 514. Implicitly, this definition also creates/opens the UNIX socket /dev/log and UDP port 514.
- The second set of definitions define a number of filtering rules. These filtering rules, in the example, identify console messages, mail messages and generic messages.
- The third set of definitions define a number of destinations.
- The fourth set of definitions finally form the link between source, filter and destination. They define where to send a message that originates from a certain source and conforms to a specific filter.

**Instructor notes:**

**Purpose** — Show an example syslog-ng.conf file.

**Details** —

**Additional information** — The visual is only a small example. You might want to show an actual syslog-ng.conf file from a SUSE system instead.

**Transition statement** — Another syslog daemon is rsyslog.

## Rsyslog

---

- Rainer's Syslog
- Improvements over traditional BSD syslog:
  - Content-based filtering in addition to facility/priority
  - TCP transport
  - Millisecond granularity in timestamps
  - Message tagging with host name of relay
  - TLS encryption
  - Buffering
  - Logging directly into databases
- Used in RHEL6
- Configuration file: /etc/rsyslog.conf

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-7. Rsyslog

LX158.0

### Notes:

Rsyslog is a system logging daemon written by Rainer Gerhards. It was specifically written to compete with Syslog-NG. It has a number of improvements over traditional syslog and Syslog-NG. As an example, it is able to buffer messages that need to be sent to a remote server, when that server is down. It can also log messages directly into a number of popular databases, such as MySQL.

Rsyslog is the default system logger in RHEL6. Its configuration file is /etc/rsyslog.conf.

***Instructor notes:***

**Purpose** — Introduce rsyslog.

**Details** —

**Additional information** —

**Transition statement** — Let's look at the configuration file.

## /etc/rsyslog.conf example

```
$ModLoad imuxsock.so      # provides support for local system logging
$ModLoad imklog.so        # provides kernel logging support

$WorkDirectory /var/spppl/rsyslog # where to place spool files

# Use default timestamp format
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

kern.*                      /dev/console
*.info;mail.none;authpriv.none;cron.none  /var/log/messages
authpriv.*                   /var/log/secure
mail.*                       /var/log/maillog
cron.*                       /var/log/cron
*.emerg                      *
*.*                          @@192.168.2.1:514
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-8. /etc/rsyslog.conf example

LX158.0

### Notes:

The visual shows an example /etc/rsyslog.conf file. You can see that the file, to a large extent, contains syntax directly lifted from the traditional BSD syslog.conf file. But at the start of the file there are a number of directives that all start with a dollar sign.

Rsyslog is, to an extent, modularized. Several Open Source modules are included in the package, but there are also commercial, closed source modules available.

Depending on the modules loaded, several further directives are available. The visual shows two of them:

- The **\$WorkDirectory** directive identifies a directory where syslog messages should be stored in case a remote server is not available. Several related directives (not shown in the visuals) allows you to further configure the behavior of rsyslog in this case. You can for instance set a maximum size of this directory, and configure the behavior of rsyslog in case it is forced to shut down.

- The **\$ActionFileDefaultTemplate** defines the template for the messages that is used when writing messages in a file. With the RSYSLOG\_TraditionalFileFormat setting, these messages will be compatible with BSD syslogd messages.

**Instructor notes:**

**Purpose** — Discuss rsyslog.conf

**Details** —

**Additional information** —

**Transition statement** — Okay, enough talk about the daemons. Let's take a look at the client side of things too.

## **logger command**

- Logs messages to system logger
- Syntax: `logger -p facility.priority message`

```
# logger -p daemon.info This is a test
# tail -1 /var/log/messages
Feb 18 16:34:32 pentium logger: daemon.info This is a test
```

```
$ logger -p kern.panic Kernel panic! Please log off NOW!
$
Message from syslogd@host at Fri Feb 18 16:42:38 2006 ...
host logger: Kernel panic! Please log off NOW!
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-9. logger command

LX158.0

### **Notes:**

#### **Introduction**

Logging is usually built-in into the system daemons. But, you might also want to do some logging ourselves, especially if you are writing complex scripts. That is what the **logger** command is for.

#### **Operation**

The logger command is really simple. The only thing you need to do is specify the facility, priority and the message itself, and it will be sent to the **syslogd** daemon. Refer to the example above.

Note that the logger command is not a privileged command; every user can make use of this command to log any message to the **syslogd** daemon. It is important to be able to recognize messages coming from the **logger** command since users might try to fool you into panicking.

**Instructor notes:**

**Purpose** — Introduce the logger command.

**Details** — Discuss student notes.

**Additional information** — Point out to the students that the logger command is really useful in scripts. But also point out that it is not really secure and that every user can trick you into believing things that are not there.

**Transition statement** — You can imagine that after a while your log file fills up. Let's see how to clean them.

## **logrotate command**

- **logrotate** automatically "rotates" logs:
  - Copies the current log to archive log
  - Can compress archive log
  - Can mail archive log
  - Cleans the current log
  - Deletes old archive logs
  - Usually run from cron
- Criteria for rotation:
  - Time
  - Size
- Config file: /etc/logrotate.conf

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-10. logrotate command

LX158.0

### **Notes:**

#### **Introduction**

When a log file grows, there comes a point in time where you might want to clean it out.

If you do not do that, you end up with a full /var file system, and you are not able to tell from the log file what is wrong with your system.

#### **Clean up log files**

**logrotate** is designed to ease administration of systems that generate large numbers of log files. It allows automatic rotation, compression, removal, and mailing of log files.

Each log file can be handled daily, weekly, monthly, or when it grows too large.

The **logrotate** command is normally run from cron. It cleans out all the specified log files, based on the information in the /etc/logrotate.conf file. It can do any of the following things with the log file:

- Copy the contents of the log file to an archive log file. This file is usually named the same as the log file, with a number appended.

- Compress the archive log file so that it uses less space on your file system.
- Mail the log file to someone.
- Clean the current log.
- Delete old archive logs, ensuring that only a limited amount of archive logs are being saved.

The decision when to rotate a log can be based on two criteria: size of the log file (for instance, rotate when the file size exceeds 50 kilobytes) or the time of day (for instance, rotate at midnight).

***Instructor notes:***

**Purpose** — Introduce the logrotate command.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Let's see how we configure logrotate.

## Sample /etc/logrotate.conf

```
# Global options (may be overwritten by local options)
weekly
rotate 4
errors root
create
# Include several config files in the given directory
include /etc/logrotate.d
# local options for some logfiles
/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 1
}
/var/log/messages {
    size 500k
    postrotate
        /usr/bin/killall -HUP syslogd
    endscript
}
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.

US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-11. Sample /etc/logrotate.conf

LX158.0

### Notes:

#### Introduction

The /etc/logrotate.conf file starts with a section that describes global options, options that apply to all files that need to be rotated. In the sample above, the following global options are defined:

- Rotate all files weekly
- Only keep four archive logs around
- Send all errors to root
- Create a new, empty log file after rotation
- The compress function is commented out, so no compression is being done

The next line, “include /etc/logrotate.d”, tells the logrotate command to read all files in the /etc/logrotate.d directory and to add the contents of those files to this file.

This way programs (and thus, log files that need to be rotated) can be added to the system without the need for the install program (rpm) to change existing files.

---

The next couple of lines each define a log file that needs to be rotated. If no options are given, the default options are used.

For a complete list of possible options, consult the manual page for logrotate.

***Instructor notes:***

**Purpose** — Define /etc/logrotate.conf file and role.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — You are not keeping all these logs for no reason. Let's last take a look at how to analyze them.

# Analyzing log files

- Analyze log files regularly
  - Preferably through a cron job, every day
- Possible strategies:
  - Read through whole log file
  - Search for interesting things (positive search)
  - Discard uninteresting things (negative search)
  - Use automated tools for analysis
- Automated tools
  - Simple: `grep`, `grep -v`, `logcheck`, `logdigest`
  - Intermediate: `logwatch`, `logsurfer`
  - Advanced: `swatch`  
<http://sourceforge.net/projects/swatch/>
- Automated tools typically send email with results.
  - Do not work if your email subsystem is broken or disabled

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
 US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-12. Analyzing log files

LX158.0

## Notes:

### Introduction

Log files are not collected for fun. They contain valuable information about the overall health of your system, and things that went wrong. It is therefore a good idea to analyze your log files regularly.

There are several strategies for analyzing a log file:

- You can read through the whole log file. With short log files, this generally is not a problem, but it quickly becomes tedious when your log files are longer than a few hundred lines. Nevertheless, in case of strange problems, it might be necessary anyway, so that you can correlate different log file entries.
- You can search through the log file (using `grep` or `vi`'s search capability) for interesting items. This is typically done when you are looking for something specific, such as all the actions of a particular user. Searching for specific items like this is called a positive search.

- You can perform a negative search through the log file. A negative search typically uses a list of non-interesting items. Using, for instance, the **grep -v** command, the log file is analyzed and all non-interesting items are filtered out. This, in theory, leaves you with only the interesting items to look at.

(This will not immediately work correctly. The list of non-interesting items therefore changes a lot over time.)

- You can use automated tools for log file analysis. These tools analyze the log file line by line, and are capable of doing both positive and negative searches. Some tools are even capable of correlating different log lines with each other.

Several automated tools exist for log file analysis:

- The easiest tool for log file analysis is **grep**. It can be used for on-the-fly analysis, or can be put into a logrotate postrotate script for positive and negative searches (with the **-v** option), of which the results are then e-mailed to the administrator. grep allows you to list the expression to search for on the command line, but the expression to search for can also be stored in a file, which is then referenced using the **-f** option.

With this last option, you can create a file (for instance `/etc/ignore`), containing regular expressions that match the lines of the log file that you are not interested in. You then run the command "`grep -v -f /etc/ignore /var/log/messages`" and this will list the messages that you are (hopefully) interested in exclusively.

- **logcheck** is a simple script which checks your log files from a cron job. It uses grep and grep **-v** extensively in a smart combination. Another advantage of logcheck over plain grep is that logcheck keeps track of what it has analyzed already, so it does not present results twice.
- **logdigest** is based on logcheck, and works generally the same. All configuration files are in `/etc/logdigest`. It is available on SLES, although it is not installed by default.
- **logwatch** is a series of perl scripts that are able to check different log files and services. logwatch itself knows the default behavior of just about every service that might be running on your Linux system and filters the interesting log items automatically. Therein lies its weakness too: it can be difficult to configure logwatch for a specific situation or service. The logwatch configuration directory, `/etc/log.d`, is a myriad of scripts, configuration files and symbolic links which can make it difficult to figure out where to make a change to get a certain thing to be reported or not. logwatch is installed on a RHEL/Fedora system by default.
- **logsurfer** again uses positive and negative matches to browse through a log file, but it uses a slightly more elaborate pattern file, `/etc/logsurfer.conf`. logsurfer is available on a SLES system by default, although it is not automatically installed.
- **swatch** is a heavy-duty log file analysis tool which is really popular in the UNIX network administrator's world. It is highly configurable and is capable of performing real-time log file analysis: you will hear of any problems only a few seconds after the log lines are added to the log file instead of having to wait for a scheduled log file analysis.

---

The “hear” in the last sentence can be taken literally: If your pager or cell phone has a scriptable interface, then swatch can send the relevant log entries to your pager or cell phone automatically.

Depending on your distributions, one or more of these tools might already be installed by default or might need to be installed separately.

A last note: most automated tools submit their results by email and do not submit a report if there is nothing to report. That means that not receiving a report can have two causes:

- There is nothing to report
- Your email subsystem is broken or disabled

Beware this last scenario, especially if you use these tools to monitor a large number of systems that do not all send in a report every day.

## **Instructor notes:**

**Purpose** — Discuss log file analysis.

**Details** — This visual discusses a combination of tools that are either included with the distributions or available online from various Web sites. These are provided as an example only and are in no way endorsed by IBM.

**Additional information** — Regarding logcheck, at the time of this writing, the Web site [www.psionic.com/abacus/logcheck/](http://www.psionic.com/abacus/logcheck/) is not available (Psionic has been redirected to [www.cisco.com](http://www.cisco.com)). We will leave the reference to logcheck in the student notes as it should hopefully be available again on another Web site. However, be advised this tool might no longer be available.

**Transition statement** — Okay, that's logging. Let's look at a few other methods of troubleshooting.

## **14.2.Troubleshooting**

### **Instructor topic introduction**

**What students will do —**

**How students will do it —**

**What students will learn —**

**How this will help students on their job —**

# Troubleshooting

- Identifying and fixing problems
- Required:
  - Deep understanding of the system
  - Knowledge of dependencies in the system
  - Knowledge of problem determination tools
  - Knowledge of problem solving methods
  - Experience
- Useful:
  - Documentation
  - Reference systems
  - Internet access
  - No outside distraction
  - Sparring partner



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-13. Troubleshooting

LX158.0

## Notes:

### Introduction

Troubleshooting is a short name for identifying and fixing problems. Most people consider it an art form which takes years to get proficient in. This unit gives you some general techniques and tools that will help you in becoming proficient in it too. Troubleshooting generally requires you to have a deep understanding of the underlying system and its dependencies and of the troubleshooting tools that are available on your system. Also, a lot of experience helps a lot too. Useful things to have include documentation, reference systems, and Internet access. However, there are two things that are most often forgotten:

- Having no outside distraction is really important, especially when solving critical problems on production systems. It is really hard to solve a pressing problem if the phone rings every minute. In fact, large system administrator groups typically have emergency scenarios where one team member is tasked with answering the phone and talking to management so that the others are able to direct their full attention to the problem.

- Having a sparring partner with more-or-less equal knowledge of the system is also indispensable, since he or she might see things or think of things that you did not, and vice versa.

***Instructor notes:***

**Purpose** — Define troubleshooting.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Let's look at the first step, identifying the problem.

## Identifying the problem: Part 1

- Check the system to see if this is an isolated failure (single service not responding) or if it is a broader issue.
  - For example, a http service might fail (not respond) due to networking issues.
- Check log files in /var/log for any sign of failure (generic and application specific).
  - Debugging switch or key might give more information.
- Check configuration files.
  - Use syntax checkers if available.
- Check and see if anything was changed recently.
  - Run a rpm verification on the system.



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-14. Identifying the problem: Part 1

LX158.0

### Notes:

#### Service not responding

Identifying the problem usually starts with determining if the reported problem is an isolated failure (meaning one subsystem) or is it a broader issue. Actions to take:

- Test the server and any services that it relies on for signs of failure.



**Note:** A service failure might be caused by a problem with an underlying service, such as networking, DNS, PAM, full file systems, improper permissions, or the X Font Server (xfs). For example, an NFS mount will fail if either forward or reverse name resolution fails.

- Read the log files for signs of failure, both the generic log files (such as /var/log/messages) and the application-specific log files, which are usually located in or under /var/log as well. Most services have a debugging switch which greatly increases

the output to the log file, especially if you reconfigured your /etc/syslog.conf file to log debug output too.

- If your log files do not give you a clue, read the configuration files for the service that you are debugging. Most services provide a method to check the configuration file for syntax errors. For example, Apache provides the apachectl **configtest** command that checks for errors in the configuration file.
- Check and see if anything was changed recently:
  - Run a rpm verification:  
`# rpm -Va`
  - From Rescue Mode, use the -root option to the **rpm** command:
    - For example, on SUSE with the root file system mounted onto the /mnt mount point, enter:  
`# rpm -root /mnt -Va`
    - For example, on RHEL/Fedora with the root file system mounted onto the /mnt/sysimage mount point, enter:  
`# rpm -root /mnt/sysimage -Va`
- If Tripwire<sup>1</sup> is configured on the system, run a tripwire check.

<sup>1</sup> Tripwire is a security package that is included with the SUSE Linux distribution, available from <http://sourceforge.net/projects/tripwire> or from [www.tripwire.com](http://www.tripwire.com) (Tripwire Enterprise/Server) for purchase.

***Instructor notes:***

**Purpose** — Introduce students to actions to take when a problem is reported.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Next, let's look at some additional steps to consider when troubleshooting a problem.

## Identifying the problem: Part 2

- Check the change documentation for the system.
- Compare with reference system.
- Check the Web:
  - Google: <http://www.google.com/linux>
  - The Linux Documentation Project: <http://www.tldp.org>
  - Red Hat Bugzilla: <https://bugzilla.redhat.com>
  - Novell Support: <http://www.novell.com/support>



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-15. Identifying the problem: Part 2

LX158.0

### Notes:

#### Additional things to consider

When dealing with a failure of a service or a system-wide outage, the following items should be also considered:

- Are there multiple system administrators managing the system?
- How does the failing system compare to a working system?
- Has anyone else seen this type of failure before?

#### Multiple administrators

For systems that are being administered by multiple system administrators, it is recommended that a system change log be kept. Doing so will help reduce the time to identify the cause of a service/system failure. Checking this type of log will provide hints as to what has changed and how the changes might be rolled back, if needed.

## Reference system

During times of a service/system failure, it might be useful to compare the actual situation with a working reference system, another system running the same services.

## Web resources

Finally, it can be useful to check the Web. Various Web sites, including the one from your distributor, include bug tracking databases which can greatly help you if you use them properly. Sites of interest include:

- Google: <http://www.google.com/linux>
- The Linux Documentation Project: <http://www.tldp.org>
- Red Hat Bugzilla: <https://bugzilla.redhat.com/>
- Novell: <http://www.novell.com/support>

**Instructor notes:**

**Purpose** — Introduce students to other things to consider when dealing with problem determination.

**Details** — Discuss the student notes.

**Additional information** — The Web address for Google shown in the text/visual is called a google hack. It will lock down all searches to the area of Linux.

**Transition statement** — Another source of debugging information might be a core dump. Let's look at what that is.

## Core dumps

- Due to programming/compilation errors, programs might misbehave.
  - Access memory outside the assigned memory area
  - Perform illegal instructions
  - Divide by zero
  - And so on
- In these cases, the kernel will detect this behavior and *dump core*, which saves the current program state in a *core* file.
  - A core dump can usually be forced with **Ctrl+\**
- This file can be read by debuggers such as **gdb**.
  - Only useful for the programmer
  - Core dumps can generally be deleted without consequence
- Saving core dumps can be prevented with `ulimit -c 0`

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-16. Core dumps

LX158.0

### Notes:

#### Introduction

Due to programming or compilation errors, programs might misbehave in certain ways:

- They might try to access memory outside their assigned memory area
- They might try to perform illegal instructions
- They might try to divide by zero

In most cases, the kernel will detect this behavior<sup>1</sup>, interrupt the program and dump the current state of the program to a core file. This file is usually called *core* and might be several megabytes in size.

It is also possible to force the creation of a core dump by sending the program the SIGQUIT signal. This is usually done with **Ctrl+\** keys. Note, however, that it is possible for the programmer to write a signal handler that assigns a different meaning to this signal.

<sup>1</sup> In fact, it is usually the CPU that detects these illegal instructions. The CPU will then suspend the program and start the error handler of the kernel.

This core file can then be used by the programmer to figure out what went wrong in his or her program. For this, the programmer typically uses a debugger such as gdb to read the core file. For other users, a core file is normally not interesting and can safely be deleted. In fact, most system administrators will run a cron job which deletes all core files older than three days automatically.

If you do not want core dumps to be saved, you can set the maximum size of them to zero with the command **ulimit -c 0**. Most distributions will set a soft core dump limit of zero by default.

***Instructor notes:***

**Purpose** — Introduce students to core dump files.

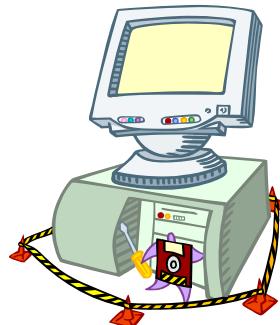
**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Once you have found the error, it is usually really easy to fix it.

## Fixing the problem

- Fixing the problem is usually obvious once you find the error.
- When repairing a problem, change one item at a time.
- Can be more complicated if system refuses to boot normally.
- Solutions:
  - Boot from boot floppy
  - Boot into single user mode and/or with special kernel parameters such as init=/bin/bash
  - Boot into Rescue Mode



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-17. Fixing the problem

LX158.0

### Notes:

#### Introduction

Once the error has been found, it needs to be fixed. This is typically a trivial task, but might become more complicated if the system refuses to boot properly because of that error. In that case, there is a number of things you can do:

- Boot from the boot floppy disk that was created during the installation process. This boot disk usually consists of a boot loader (GRUB or LILO), a Linux kernel, and (if needed) an Initial RAM Disk. This allows you to bypass any problem that might exist in your master boot record or in your /boot partition/directory (that is, corrupt or missing kernel image, initrd), but does not help you if the problem is in your root file system or further along in the boot process. A boot disk is typically created with the mkbootdisk shell script and is system-specific to a certain degree:
  - The boot loader configuration contains the device name of your root partition, typically something like /dev/hda5. If your root partition has moved, you need to specify a new one at the GRUB or LILO boot prompt with linux root=/dev/hda6.

- The kernel on the boot disk is optimized for your processor. This means that you cannot use a boot disk created on an older Pentium-II machine to boot a regular Pentium machine.
- The initial root disk on the boot disk only contains the modules that are needed on your system.

Since the kernel and initrd combined are typically larger than a single floppy, and since a lot of modern systems are not equipped with floppy drives anymore anyway, most distributions have stopped creating boot floppy disks during the installation.

- Boot into single user mode. This requires the boot process, up to and including the /etc/rc.sysinit file, to be in full working order, but might help you if you have a problem starting certain services.

On a RHEL/Fedora system, the single user mode does not prompt for the root password. Therefore, it can be used to recover the root password if that was lost. A SUSE system does require the root password before booting in single-user mode and thus cannot be used to recover the root password.

You can also use a boot parameter such as **init=/bin/bash** or **init=/bin/sh**. This uses **/bin/bash** or **/bin/sh** as first program to start, instead of **/sbin/init**, and thus gives you a shell prompt immediately. The only disadvantage of this, compared to the single user mode, is that your boot scripts have not yet been executed. This means that only the root file system is mounted, read-only. Before you can do anything useful, you probably have to mount this read-write with the command **mount -o remount,rw /**, and you might also need to mount other file systems with **mount -a**.

- Boot into a Rescue Mode. In this case, the full boot process is done from CD-ROM or the network. This allows you to fix virtually any problem on disk.

***Instructor notes:***

**Purpose** — Introduce students to how to fix a problem.

**Details** — Discuss the student notes.

**Additional information** — Remind students to only change one thing at a time!

**Transition statement** — Next, let's look at what Rescue Mode is.

## 14.3. Rescue Mode

### Instructor topic introduction

**What students will do —**

**How students will do it —**

**What students will learn —**

**How this will help students on their job —**

## What is Rescue Mode?

- The ability to boot a small Linux environment entirely from an external source.
- Why Rescue Mode?
  - You are unable to boot Linux
  - You are having hardware or software problems
  - You want to get a few important files off your system's hard drive
- Rescue Mode is available from CD #1 of distribution or from network server.



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-18. What is Rescue Mode?

LX158.0

### Notes:

#### Small Linux environment entirely from an external source

If you have worked with any UNIX-like operating system, you have probably needed to recover or repair that system at some time. Even though today's operating systems, like Linux, are robust and stable, they are still susceptible to possible corruption. In the case of Linux, this is handled by a process called Rescue Mode.

Linux Rescue Modes generally fall into two classes. The first class of these are rescue disks that are provided with or produced by a specific Linux distribution and are therefore targeted toward correcting problems encountered on a machine running that distribution. The second class of Rescue Mode is distribution-independent, single-CD rescue disks that are designed to help you recover any Linux system, regardless of the distribution on which it is based.

## Why Rescue Mode?

Some classic examples of things that can induce boot problems are:

- Missing or damaged disk blocks (such as the master boot record (MBR))
- Missing files required by the boot loader
- Bad or incorrectly updated boot loader configuration information
- Bad or missing kernel
- Bad or missing initrd

Rescue Mode is used in the situation where your system will not boot because the root file system is corrupted and you cannot even boot to the point where you can access the fsck utility on the system itself. The following are possible reasons for using Rescue Mode:

- File system mis-configuration
- Bootloader problems
- Damaged kernel

In the worst case, you might find that your file systems are so damaged that it is easier to reinstall your system in its entirety. In this case, you can boot into Rescue Mode and then use backup utilities to back up files to supported removable media or over the network.

**Instructor notes:**

**Purpose** — Introduce Rescue Mode.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Next, let's discuss the use of the **chroot** command as it relates to the Rescue Mode environment.

## Use available rescue tools (1 of 3)

- The first task is to see if your partition table exists.

- # fdisk -l
- Disk /dev/sda: 80.0 GB, 80026361856 bytes
- 255 heads, 63 sectors/track, 9729 cylinders
- Units = cylinders of 16065 \* 512 = 8225280 bytes

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1		1	4152	33342907	7	HPFS/NTFS
/dev/sda2	*	4152	4283	105227	83	Linux
/dev/sda3		4283	4414	1052257+	82	Linux swap / Solaris
/dev/sda4		4414	9729	42700769+	f	W95 Ext'd (LBA)
/dev/sda5		4414	5066	1574463	83	Linux
/dev/sda6		5067	5719	252459	83	Linux
/dev/sda7		5720	6372	252459	83	Linux
/dev/sda8		6373	7025	1574463	83	Linux

- Can you pick out your root partition?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-19. Use available rescue tools (1 of 3)

LX158.0

### Notes:

One of the first thing you need to do in rescue mode is to figure out where your root and boot and usr file systems exist. The **fdisk -l** command will show you your partition table – or not, if it is corrupted.

You then need to discover which of the partitions show is your “root” partition, as well as your boot and usr partitions. This can be a daunting task if this happens to *not* be your machine – or one that you setup.

There are some hints, such as size or type. If there is no documentation, you just have to take your best guess and “mount and see” one at a time.

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Use available rescue tools (2 of 3)

- Next, did you choose the correct partition?
  - # ls -l /mnt/resq
  - total 264
  - drwxr-xr-x 3 root root 4096 Dec 18 2007 X11R6
  - drwxr-xr-x 2 root root 65536 Dec 19 2007 bin
  - drwxr-xr-x 2 root root 4096 Oct 10 2006 etc
  - drwxr-xr-x 2 root root 4096 Oct 10 2006 games
  - drwxr-xr-x 119 root root 12288 Dec 18 2007 include
  - drwxr-xr-x 6 root root 4096 Sep 11 2007 kerberos
  - drwxr-xr-x 117 root root 69632 Dec 19 2007 lib
  - drwxr-xr-x 13 root root 4096 Dec 19 2007 libexec
  - drwxr-xr-x 11 root root 4096 Dec 18 2007 local
  - drwxr-xr-x 2 root root 20480 Dec 19 2007 sbin
  - drwxr-xr-x 232 root root 12288 Dec 19 2007 share
  - drwxr-xr-x 4 root root 4096 Dec 18 2007 src
  - lrwxrwxrwx 1 root root 10 Dec 18 2007 tmp -> ../../var/tmp
- Can you prove this is or is not root?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-20. Use available rescue tools (2 of 3)

LX158.0

### Notes:

Let's say you chose this partition because it was the largest or the smallest or lowest/highest partition number, or one of many other reasons. After you mount it, do you like what you see?

If you remember from earlier in this course, there are five directories that must exist in the root directory (and we are *not* speaking of “/root/” here). Those five directories are /dev, /etc, /bin, /sbin and /lib. The display above does not have these so it cannot be the one we are looking for.

Can you figure out which partition this is? One hint that it is not “/” is the entry for “tmp” – it is a symbolic link to somewhere “up”. One hint that this display is from “/usr” are the entries for “X11” and “games”.

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Use available rescue tools (3 of 3)

- Maybe this is the correct partition?
  - ls -l /mnt/resq
  - total 176
  - drwxr-xr-x 2 root root 4096 Dec 19 2007 bin
  - drwxr-xr-x 5 root root 4096 Dec 19 2007 boot
  - drwxr-xr-x 11 root root 4220 Aug 4 15:45 dev
  - drwxr-xr-x 106 root root 12288 Aug 4 15:46 etc
  - drwxr-xr-x 3 root root 4096 Dec 18 2007 home
  - drwxr-xr-x 14 root root 4096 Dec 19 2007 lib
  - drwxr-xr-x 2 root root 4096 Aug 4 13:19 media
  - drwxr-xr-x 3 root root 16384 Dec 31 1969 mnt
  - dr-xr-xr-x 147 root root 0 Aug 4 09:14 proc
  - drwxr-x--- 21 root root 4096 Jan 14 2008 root
  - drwxr-xr-x 2 root root 12288 Dec 19 2007 sbin
  - drwxr-xr-x 2 root root 4096 Oct 10 2006 srv
  - drwxr-xr-x 11 root root 0 Aug 4 09:14 sys
  - drwxrwxrwt 4 root root 4096 Aug 4 14:35 tmp
  - drwxr-xr-x 14 root root 4096 Dec 18 2007 usr
  - drwxr-xr-x 26 root root 4096 Dec 18 2007 var
- Can you prove this is or is not root?

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-21. Use available rescue tools (3 of 3)

LX158.0

### Notes:

This display looks like the right partition. First, it contains all five necessary directories. Now you might need to find the “boot” and “usr” partitions to continue your challenge to fix the system.

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

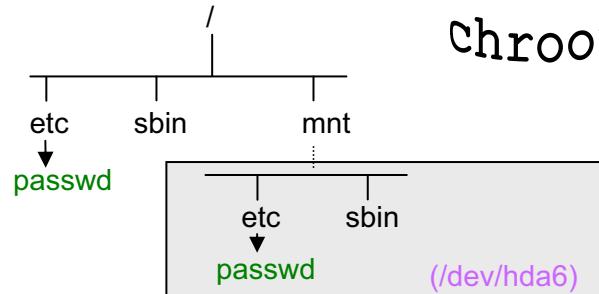
**Transition statement —**

## chroot command

- chroot command reassigned your root path.

**Don't Forget**

chroot



```

Rescue: ~ # df
Filesystem      1K-blocks   Used Available Use% Mounted on
/dev/root       57576      57576        0 100% /
udev            192796      72    192724     1% /dev
/dev/sda6       10490040  2934416  7555624   28% /mnt
/dev/sda5       17510244  38808  17471436     1% /mnt/home

Rescue: ~ # chroot /mnt
Rescue: ~ # df
Filesystem      1K-blocks   Used Available Use% Mounted on
/dev/sda6       10490040  2934416  7555624   28% /
proc             10490040  2934416  7555624   28% /proc
udev            10490040  2934416  7555624   28% /dev

```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-22. chroot command

LX158.0

### Notes:

The **chroot** command reassigned your root path to the disk file system instead of the rescue file system. If you do not use the **chroot** command, any other command will perform an action on the RAM disk, not the target disk you actually want to fix!

Looking at the example in the visual (SLES), we see the following:

- The root file system is stored in a RAM disk called /dev/root
- The root file system of the SLES installation on the hard disk is stored on /dev/hda6
- /dev/hda6 is mounted on /mnt

Without the use of the **chroot** command, any operation will be relative to the root node of /dev/root. For example, when recovering the root password on a SLES installation:

- If the **passwd** command is issued without chroot being invoked, the /etc/passwd file of the root file system stored in the RAM disk (/dev/root) will be modified

With the use of the **chroot** command invoked with the mount point of /mnt, any operation will be relative to the root node of /dev/hda6:

- If the **passwd** command is issued with chroot being invoked, the /etc/passwd file of the root file system stored in the hard disk (/dev/hda6) will be modified.

Note that just mounting the root filesystem (on, for instance /mnt) and then **chrooting** to /mnt might not be sufficient to solve your problem. This is because the full filesystem structure of your system is not complete. As an example, SLES stores the **passwd** program in /usr/bin. So if you need to reset the root password, you need to mount the filesystem containing /usr on /mnt/usr as well, before doing the **chroot** to /mnt.

Another issue is the fact that the virtual filesystems, particularly /dev, are not mounted in the **chroot** environment. This may cause issues in several ways. For instance, if you try a **mount -a** in the chroot environment, the nodes /dev/sda1, /dev/sda2 and so forth are simply not there. Also, when resetting a password, the **passwd** program needs to retrieve some random data from /dev/random or /dev/urandom.

To prevent these kinds of problems, you can "bind mount" the virtual filesystems that are mounted in the rescue environment, into the chroot environment as well:

```
# mount -o bind /dev /mnt/dev  
# mount -o bind /proc /mnt/proc  
# mount -o bind /sys /mnt/sys
```

These commands are entered after the root filesystem has been mounted on /mnt, but before the chroot command is issued.

**Instructor notes:**

**Purpose** — Introduce students to the role/use of the **chroot** command in the Rescue Mode environment.

**Details** — Discuss student notes.

**Additional information** —

**Transition statement** — Let's look at how to access Rescue Mode in RHEL/Fedora.

## Booting Rescue Mode: RHEL

- Step 1: Boot with **rescue** option, or use proper menu option

- Step 2: Decide to mount existing installation to /mnt/sysimage

- Step 3: Acknowledge mount on /mnt/sysimage

```
boot: linux rescue
```

```
+-----+ Rescue +-----+
| The rescue environment will now attempt to      # |
| find your Red Hat Linux installation and mount  # |
| it under the directory /mnt/sysimage. You can   # |
| . . .
+-----+     +-----+     +-----+
| Continue | | Read-Only | | Skip |
+-----+     +-----+     +-----+
```

```
+-----+ Rescue +-----+
| Your system has been mounted under           #
| /mnt/sysimage.
|
| Press <return> to get a shell. If you
| would like to make your system the
| root environment, run the command:
|
| chroot /mnt/sysimage
|
| The system will reboot automatically
| when you exit from the shell.
|
+-----+
| OK |
+-----+
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-23. Booting Rescue Mode: RHEL

LX158.0

### Notes:

#### Introduction

The Rescue Mode environment for RHEL/Fedora is accessed by either booting from the first CD-ROM of the distribution or over the network. When booting from the CD-ROM, issue the command **linux rescue** at the boot: prompt. If you boot from the original RHEL media, you can also use the "Rescue Mode" menu option.

#### Automatic mount of Linux installation

During the loading of the Rescue Mode environment, you will be prompted if you want Rescue to find any existing installations and mount them to the directory structure /mnt/sysimage. The options presented are:

- Continue:** Find and mount the discovered installation to the mount point /mnt/sysimage with read-write
- Read Only:** Find and mount the discovered installation to the mount point /mnt/sysimage as read-only
- Skip:** Do not attempt to discovered the existing installation

In the example shown in the visual, the **Continue** button was selected, and the system administrator is prompted to acknowledge that the mount was successful.

***Instructor notes:***

**Purpose** — Introduce students to the steps needed to boot the Rescue Mode environment.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Next, let's discuss what to do once Rescue Mode has been booted.

## Using Rescue Mode: RHEL

- At this point, take administrative action to fix problem or recover data.
- When using the **chroot** command, the first **exit** will leave the **chroot** environment.
  - Second **exit** will leave the rescue environment.

```

sh-3.00# df
Filesystem      1K-blocks      Used Available Use% Mounted on
rootfs          7163        4944       1810    74% /
/dev/root.old   7163        4944       1810    74% /
/tmp/loop0      183580     183580         0 100% /mnt/runtime
/dev/VolGroup00/LogVol00
                  16320424    6876448    8614948  45% /mnt/sysimage
/dev/sda1        101105     13834      82050   15% /mnt/sysimage/boot
/dev/root.old   7163        4944       1810    74% /mnt/sysimage/dev
sh-3.00# chroot /mnt/sysimage
sh-3.00# df
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/VolGroup00/LogVol00
                  16320424    6876448    8614948  45% /
/dev/sda1        101105     13834      82050   15% /boot
sh-3.00# exit
sh-3.00# exit

```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-24. Using Rescue Mode: RHEL

LX158.0

### Notes:

#### Now what?

Once the Rescue Mode environment has been booted, it is now time to take administrative action to fix or recover the system. If you selected the mounting of the existing installation to the /mnt/sysimage mount point, you can now dive in to resolve the problem. Remember, that you might need to use the **chroot** command to change the roots path to the /mnt/sysimage mount point.

#### Exiting chroot/Rescue

Exiting chroot/Rescue relies on the use of the **exit** command. If the **chroot** command was invoked, the first exit will leave the chroot environment. The second exit will leave the Rescue Mode environment and reboot the system.

***Instructor notes:***

**Purpose** — Introduce students to what they should do next once the Rescue Mode environment has been booted.

**Details** — Discuss the student notes.

**Additional information —**

**Transition statement** — Let's look at how to access Rescue Mode in SLES.

## Booting Rescue Mode: SUSE

- Step 1: Boot with **rescue** option.
- Step 2: Log in as root.
- Step 3: Find existing file systems.
- Step 4: Mount existing installation to /mnt

```

Boot from Hard Disk
Installation
Installation—ACPI Disabled
Installation—Safe Settings
Rescue System
Memory Test

Boot Options _____
F1 Help F2 Language F3 Other Options

```

```

Rescue login: root
Rescue: ~# guessfstype /dev/hda3
/dev/sda3 *appears* to be: swap
Rescue: ~# guessfstype /dev/sda5
/dev/sda5 *appears* to be: reiserfs
Rescue: ~# guessfstype /dev/sda6
/dev/sda6 *appears* to be: reiserfs
Rescue: ~# mount /dev/sda6 /mnt
Rescue: ~# mount /dev/sda5 /mnt/home

```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-25. Booting Rescue Mode: SUSE

LX158.0

### Notes:

#### Introduction

The Rescue Mode environment for SLES is accessed by either booting from the first CD-ROM of the distribution or over the network. When booting from the CD-ROM, the boot splash screen as shown in the visual will appear. Select the **Rescue System** options.

**Note:** Rescue requires you to login as root, and no password by default is required.

#### Accessing Linux installation

Unlike RHEL/Fedora, the SLES Rescue Mode environment does not prompt you to mount the existing installation to a given mount point. It is up to you as the system administrator to know the disk layout of your system. However, if you do not know the layout, you can use the following commands:

- **fdisk -l** (To determine the partitioning of the system)
- **guessfstype** (To guess the type of file system utilized in a given partition)

In the example shown in the visual, the devices sda5/6 contain reiserfs file systems.

Using the **fdisk** command, enter:

```
# fdisk /dev/sda

Disk /dev/sda: 30.0 GB, 30020272128 bytes
255 heads, 63 sectors/track, 3649 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot Start End Blocks Id System
/dev/sda1 * 1 15 120456 83 Linux
/dev/sda2 16 3519 28145880 f W95 Ext'd (LBA)
/dev/sda3 3520 3649 1044225 82 Linux swap /
Solaris
/dev/sda5 16 2195 17510818+ 83 Linux
/dev/sda6 2196 3501 10490413+ 83 Linux
```

Given the size of the device /dev/hsa6, it is possible to guess that it is the root file system and mount it on /mnt.

**Instructor notes:**

**Purpose** — Introduce students to the steps needed to boot the Rescue Mode environment.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — Next, let's discuss what to do once Rescue Mode has been booted.

## Using Rescue Mode: SUSE

- At this point, take administrative action to fix problem or recover data.
- When using the **chroot** command, the first **exit** will leave the chroot environment.
  - **halt** will leave the rescue environment.

```
Rescue: ~ # df
Filesystem      1K-blocks   Used Available Use% Mounted on
/dev/root        57576       0  100% /
udev             192796      72  192724   1% /dev
/dev/sda6        10490040  2934416  7555624  28% /mnt
/dev/sda5        17510244  38808 17471436   1% /mnt/home
Rescue: ~ # chroot /mnt
Rescue: ~ # df
Filesystem      1K-blocks   Used Available Use% Mounted on
/dev/sda6        10490040  2934416  7555624  28% /
proc              10490040  2934416  7555624  28% /proc
udev              10490040  2934416  7555624  28% /dev
Rescue: ~ # exit
exit
Rescue: ~ # halt
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-26. Using Rescue Mode: SUSE

LX158.0

### Notes:

#### Now what?

Once the Rescue Mode environment has been booted, it is time to take administrative action to fix or recover the system. Once the existing installation is mounted, you can now dive in to resolve the problem. Remember, that you might need to use the **chroot** command to change the roots path to the /mnt mount point.

#### Exiting chroot/Rescue

Exiting chroot relies on the use of the **exit** command. If the **chroot** command was invoked, the exit will leave the chroot environment. To exit Rescue, use the **halt** command. The **halt** command will leave the Rescue Mode environment and reboot the system.

**Instructor notes:**

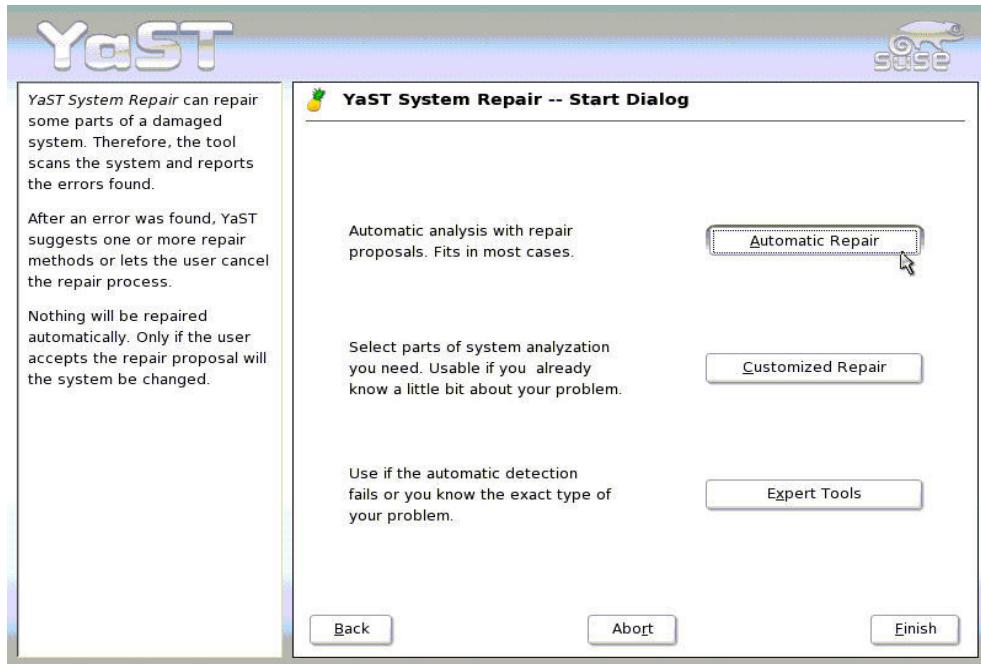
**Purpose** — Introduce students to what they should do next once the Rescue Mode environment has been booted.

**Details** — Discuss the student notes.

**Additional information** —

**Transition statement** — Let's look at another tool available under SLES.

# Repair installed system: SUSE



© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-27. Repair installed system: SUSE

LX158.0

## Notes:

### Introduction

Because it cannot be assumed that a damaged system can boot by itself and a running system cannot be easily repaired, the YaST System Repair utility is run from the SUSE Linux installation CD-ROM or network.

**Note:** Because the test and repair procedure is loaded from CD-ROM or network, it is essential to run it from an installation medium that exactly corresponds to your installed version of SLES.

### Automatic repair

This method is best suited to restoring a damaged system with unknown cause. Selecting it starts an extensive analysis of the installed system, which takes quite some time due to the large number of tests and examinations. The progress of the procedure is displayed at the bottom of the screen with two progress bars. The upper bar shows the progress of the currently running test. The lower bar shows the overall progress of the analysis process.

The log window above allows tracking of the currently running activity and its test result. The following main test runs are performed with every run.

They contain, in turn, a number of individual sub-tests:

### **Partition tables of all hard disks**

The validity and coherence of the partition tables of all detected hard disks are checked.

### **Swap partitions**

The swap partitions of the installed system are detected, tested, and offered for activation where applicable. The offer should be accepted for the sake of a higher system repair speed.

### **File systems**

All detected file systems are subjected to a file system-specific check.

### **Entries in the file /etc/fstab**

The entries in the file are checked for completeness and consistency. All valid partitions are mounted.

### **Boot loader configuration**

The boot loader configuration of the installed system (GRUB or LILO) is checked for completeness and coherence. Boot and root devices are examined, and the availability of the initrd modules is checked.

### **Package database**

This checks whether all packages necessary for the operation of a minimal installation are present. While it is optionally possible also to analyze the base packages, this takes a long time because of their vast number.

Whenever an error is encountered, the procedure stops and a dialog opens, offering details and possible solutions. It is not possible to describe all these cases. Read the messages on the screen carefully and choose the desired action from the list options. It is also possible to decline the offered repair action in cases of doubt. The system remains unaltered in this case, and no repair is ever performed automatically without prompting the user.

### **Customized repair**

If you already know what part of the system is affected, the range of the applied tests can be narrowed. Choosing Customized Repair shows a list of test runs that are all marked for execution at first. The total range of tests matches that of automatic repair. If you already know where no damage is present, unmark the corresponding tests.

Clicking **Continue** then starts a narrower test procedure that probably has a significantly shorter running time.

Not all test groups are applicable individually. The analysis of the fstab entries is always bound to an examination of the file systems, including existing swap partitions. YaST

automatically satisfies such dependencies by selecting the smallest number of necessary test runs.

## Expert tools

If you are knowledgeable with SUSE Linux and already have a very clear idea of what needs to be repaired in your system, directly apply the tools necessary for repairing it by choosing Expert tools.

### Install new boot loader

This starts the YaST boot loader configuration module.

### Run partitioning tool

This starts the expert partitioning tool in YaST.

### Fix file system

This checks the file systems of your installed system. You are first offered a selection of all detected partitions and can then choose the ones to check.

### Restore lost partitions

It is possible to attempt a reconstruction of damaged partition tables. A list of detected hard disks is presented first for selection. Clicking **OK** starts the examination. This can take a while depending on the processing power and size of the hard disk.

**Instructor notes:**

**Purpose** — Introduce system repair tool under SUSE Linux.

**Details** — There are a lot of student notes here. They are taken almost completely from the SUSE Linux administration guide. They are included only as a more detailed reference to the visual, and you are not expected to discuss each in great detail. The idea here is to show how SUSE Linux has bundled a repair system that does not require extensive knowledge of the system.

**Additional information —**

**Transition statement** — Let's check our progress.

## Checkpoint (1 of 2)

---

1. The \_\_\_\_\_ receives all logging requests and forwards it to the right destination, depending on priority and facility.
2. What does the **logger** command do?
3. The **logrotate** command:
  - a. Creates new log files
  - b. Rotates the log files
  - c. Deletes log files

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 14-28. Checkpoint (1 of 2)

LX158.0

### Notes:

**Instructor notes:****Purpose —****Details —**

## Checkpoint solutions (1 of 2)

1. The [system log daemon](#) receives all logging requests and forwards it to the right destination, depending on priority and facility.

The answer is [system log daemon](#).

2. What does the **logger** command do?

The answer is [it sends log messages to the syslogd daemon](#).

3. The **logrotate** command:

- a. Creates new log files
- b. [Rotates and cleans up log files](#)
- c. Deletes log files

The answer is [rotates and cleans up log files](#).

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —****Transition statement —**

## Checkpoint (2 of 2)

---

4. True or false: Internet access is required for troubleshooting.
5. If your X server does not start, then the problem might also be:
  - a. The network
  - b. The font server
  - c. A full file system
  - d. All of the above
6. SUSE Linux provides a \_\_\_\_\_ from the Installation menu.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure 14-29. Checkpoint (2 of 2)

LX158.0

### Notes:

**Instructor notes:****Purpose —****Details —**

## Checkpoint solutions (2 of 2)

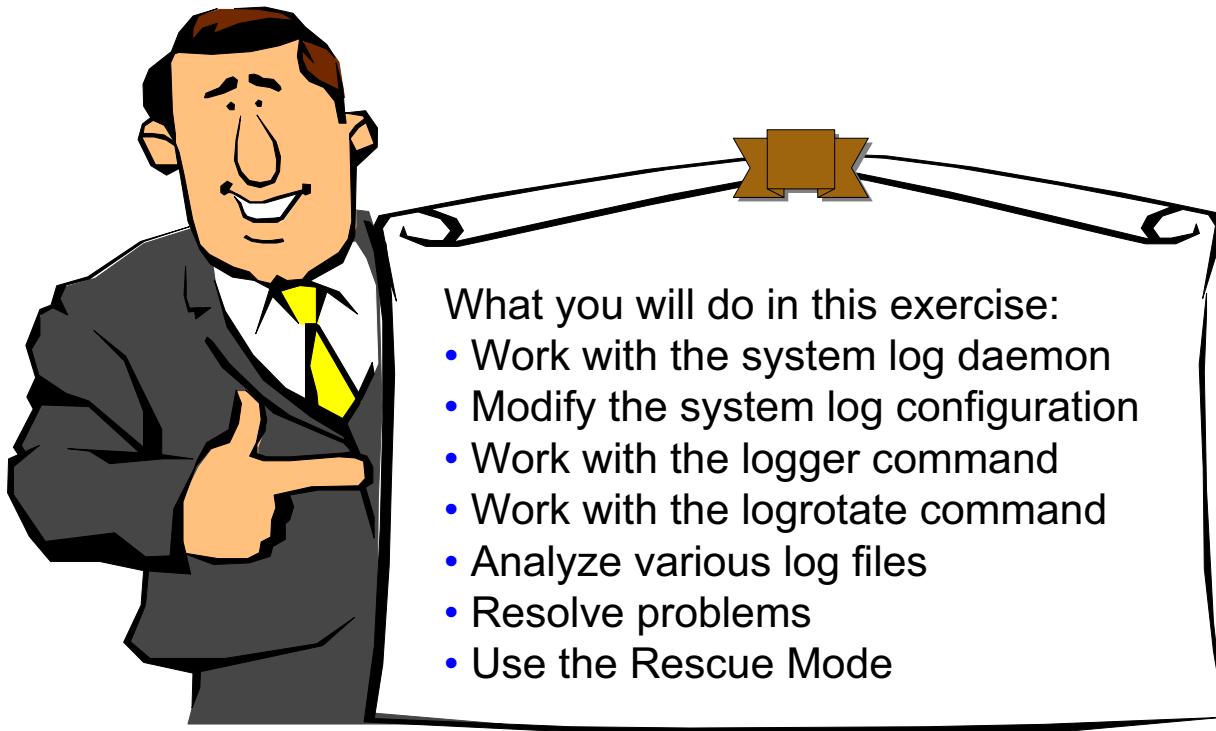
4. True or false: Internet access is required for troubleshooting.  
The answer is true.
  
5. If your X server does not start, then the problem might also be:
  - a. The network
  - b. The font server
  - c. A full filesystem
  - d. All of the above  
The answer is all of the above.
  
6. SUSE Linux provides a repair tool from the Installation menu.  
The answer is repair tool.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

**Additional information —****Transition statement —**

## Exercise: Logging and troubleshooting

---



- What you will do in this exercise:
- Work with the system log daemon
  - Modify the system log configuration
  - Work with the logger command
  - Work with the logrotate command
  - Analyze various log files
  - Resolve problems
  - Use the Rescue Mode

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-30. Exercise: Logging and troubleshooting

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Unit summary

Having completed this unit, you should be able to:

- Describe logging concepts
- Configure the **syslog**, **syslog-ng** and **rsyslog** daemons
- Use the **logger** program
- Use the **logrotate** program
- Discuss log analysis
- Perform basic problem determination
- Utilize Rescue Mode to perform system recovery

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure 14-31. Unit summary

LX158.0

### Notes:

Having completed this unit, you should understand:

- Nearly all logging on a Linux system is done through the **system logger** daemon
- The **system logger** daemon handles the log items according to facility and priority
- Modern **system loggers** can also filter on the log message itself
- The **logger** command allows you to submit log items manually
- The **logrotate** command automatically cleans up old logs
- Always check your logfiles; use debugging switches if available
- Always check proper operation of underlying services
- If a system will not boot, you can use the boot disk, single user mode or the Rescue Mode to fix the system

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**



# Appendix A. Checkpoint solutions

## Unit 1, "Introduction to Linux"

### Solutions for Figure 1-11, "Checkpoint," on page 1-29

## Checkpoint solutions

1. True or false: Linus Torvalds wrote the Linux operating system all by himself.

The answer is false.

2. Which of the following statements is not true about software licensed under the GNU GPL?

- a. You have the right to obtain and review the source code.
- b. You cannot charge any money for the software.
- c. You cannot change the license statement.
- d. You can modify the source code and subsequently recompile it.

The answer is you cannot charge any money for the software.

3. Who is the mascot of Linux?

The answer is Tux.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 2, "Installing Linux"

Solutions for Figure 2-21, "Checkpoint (1 of 2)," on page 2-51

### Checkpoint solutions (1 of 2)

---

1. True or false: Linux can coexist with Windows on the same hard disk.

The answer is true.

2. Which of the following steps is not essential in the installation process:

- a. Create partitions for Linux.
- b. Configure your printer.
- c. Select your keyboard type.
- d. Identify the packages to install.

The answer is configure your printer.

3. What is the best source of information about your hardware?

The answer is a currently installed and configured operating system, such as Windows.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

## Solutions for Figure 2-22, "Checkpoint (2 of 2)," on page 2-53

---

### Checkpoint solutions (2 of 2)

4. True or false: A network install server needs to be a Linux system.  
The answer is false. It can be any operating system that provides NFS, FTP, or HTTP services.
  
5. Which of the following install methods does not require a network server?
  - a. NFS
  - b. SMB
  - c. FTP
  - d. CD-ROMThe answer is CD-ROM.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 3, "Linux documentation"

Solutions for Figure 3-14, "Checkpoint," on page 3-29

### Checkpoint solutions

---

1. True or false: A HOWTO document is the best source of documentation if you want up-to-date information about a specific command.

The answer is false.

2. The main Linux documentation Web site is:

- a. <http://www.tldp.org>
- b. http://www.linux.org
- c. http://www.lwn.net
- d. http://www.kernel.org

The answer is <http://www.tldp.org>.

3. In which sections are manual pages divided?

The answers are user commands, system calls, library calls, devices, file formats and protocols, games, conventions, macro packages, and so forth, system administration, and Linux kernel.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 4, "Startup and shutdown"

Solutions for Figure 4-20, "Checkpoint," on page 4-56

### Checkpoint solutions

1. Name the four steps that form the startup order of a Linux system.

The answers are firmware, boot loader, kernel, and init.

2. How would you select a graphical login screen (**xdm**, **kdm**, or **gdm**)?

The answer is by setting runlevel 5 as the default runlevel in /etc/inittab.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 5, "System administration tools"

Solutions for Figure 5-9, "Checkpoint," on page 5-21

### Checkpoint solutions

---

1. The RHEL [setup](#) tool provides a menu-based interface for various tools used during a text-based installation.  
The answer is [setup](#).
2. [True](#) or false: RHEL provides separate tools that start with system-config to administer the system with a GUI interface.  
The answer is [true](#).
3. SUSE provides a tool called [YaST](#) as a GUI interface/text menu tool to be used for various system administration tasks.  
The answer is [YaST](#).
4. What is the default port number to connect with the Webmin administration tool using a Web browser?  
The answer is [10000](#).

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 6, "Package management"

### Solutions for Figure 6-19, "Checkpoint," on page 6-51

## Checkpoint solutions

1. Which basic modes of operation does rpm have?  
The answers are [install, freshen and upgrade, uninstall, query, and verify.](#)
2. Which command can I use to verify that the permissions of /etc/sendmail.cf are still correct?  
The answer is [rpm -V -f /etc/sendmail.cf.](#)
3. Which software maintenance operations does the rpm command provide?
  - a. [Installation of a RPM package](#)
  - b. Installation of a tar ball archive
  - c. Removal of seldom used packages
  - d. [Updating a package](#)
  - e. [Verification of package installation](#)

The answers are [installation of a RPM package, updating a package, and verification of package installation.](#)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 7, "X Window System and VNC"

Solutions for Figure 7-17, "Checkpoint," on page 7-49

### Checkpoint solutions

---

1. What is the function of X.org?

The answer is it is the graphical user interface for UNIX/Linux.

2. What is the function of a window manager?

The answer is it allows more control for your windows space.

3. How do you run an individual X application over a network?

The answer is set the DISPLAY variable (or option) and enable authentication with xauth or ssh.

4. What port number do you need to connect to VNC server instance :4 via VNC?

The answer is 5904.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 8, "User administration and security"

Solutions for Figure 8-37, "Checkpoint (1 of 2)," on page 8-91

### Checkpoint solutions (1 of 2)

1. What is a User Private Group?
  - a. A group for users who need privacy.
  - b. A group which has the same name as the user; this user has this group as its primary group.
  - c. A group which is used for sharing files between the members of this group.
  - d. The "staff" group.

The answer is a group which has the same name as the user; this user has this group as its primary group.

2. Where are the passwords of users stored?

The answer is /etc/shadow.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Solutions for Figure 8-38, "Checkpoint (2 of 2)," on page 8-93

### Checkpoint solutions (2 of 2)

---

3. What is the difference between authentication and authorization?

The answer is authentication is how you identify yourself to the system, and authorization specifies what you can do once logged in.

4. True or false: The user root can log in anywhere, anytime.

The answer is false.

5. True or false: PAM is the subsystem responsible for user authentication.

The answer is true.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 9, "Disk management, LVM, and RAID"

Solutions for Figure 9-28, "Checkpoint," on page 9-74

### Checkpoint solutions

1. True or false: RAID volumes can be used as physical volumes in an LVM setup.

The answer is true.

2. Mirroring is offered by RAID level:

- a. Linear
- b. Zero
- c. One
- d. Four
- e. Five

The answer is one.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 10, "File systems and file system quota"

Solutions for Figure 10-27, "Checkpoint," on page 10-69

### Checkpoint solutions

1. Assuming a blocksize of 1024, how many i-nodes and data blocks do you need for a file on an ext2 file system?
  - a. With size 0? [The answer is 1 i-node and 0 data blocks.](#)
  - b. With size 1? [The answer is 1 i-node and 1 data block.](#)
  - c. With size 2000? [The answer is 1 i-node and 2 data blocks.](#)
  - d. With size 12289 (12 K+1)? [The answer is 1 i-node and 12 data blocks directly from the i-node, an indirect block, and an extra data block. Total 14 data blocks.](#)
2. What are the two methods of copying a file to a (not yet mounted) MS-DOS floppy?  
[The answer is mount file system, and use cp command and use mcopy \(from mtools\).](#)
3. What files are important with respect to quotas?  
[The answer is /etc/fstab to specify file systems and /quota.users and /quota.groups.](#)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 11, "Kernel services, kernel configuration and device management"

Solutions for Figure 11-25, "Checkpoint (1 of 2)," on page 11-63

### Checkpoint solutions (1 of 2)

1. The Linux kernel is:

- a. a monolithic kernel
- b. a monolithic, modularized kernel
- c. a microkernel
- d. a modularized microkernel

The answer is a monolithic, modularized kernel.

2. What information can be found in the /proc virtual filesystem?

The answer is kernel, process and device information.

3. Name three ways to configure the kernel.

The answers are with a command line parameter (via the bootloader), when loading a module (via /etc/modprobe.conf), and using the /proc/sys virtual filesystem (through sysctl).

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Solutions for Figure 11-26, "Checkpoint (2 of 2)," on page 11-65

### Checkpoint solutions (2 of 2)

---

4. True or false: A character device allows random seeks.  
The answer is true.
  
5. What is the difference between /dev/random and /dev/urandom?  
The answer is /dev/urandom doesn't block when the entropy pool runs out.
  
6. Name a few commands that allow you to retrieve device information.  
The answers are **lspci**, **lsusb** and **udevadm info**.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 12, "Memory management"

### Solutions for Figure 12-16, "Checkpoint," on page 12-48

## Checkpoint solutions

1. Which file in /proc shows current system-wide memory performance statistics?

The answer is [meminfo](#).

2. List two commands that provide system memory status.

The answer is [procinfo, free, top, vmstat, ps](#). (Answers may vary.)

3. What is the difference between a paging partition and a paging file? Which is more efficient?

The answers are [a paging partition is directly written in the partition table and to disk, while a paging file has to go through the file system. A paging partition is more efficient](#).

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 13, "Virtualization"

Solutions for Figure 13-21, "Checkpoint," on page 13-51

### Checkpoint solutions

---

1. Which open source solutions for virtualization are supported in Linux?

The answer is Xen and QEMU-KVM.

2. What is KVM?

The answer is Kernel Virtualization Module - the module that provides support for the VT-x (Intel) or AMD-V (AMD) processor feature. This is a requirement for full virtualization (HVM).

3. What is libvirt?

The answer is a library that provides a solution-agnostic API, daemon, GUI, CLI and tools for virtualization on Linux.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Unit 14, "Logging and troubleshooting"

### Solutions for Figure 14-28, "Checkpoint (1 of 2)," on page 14-78

#### Checkpoint solutions (1 of 2)

1. The system log daemon receives all logging requests and forwards it to the right destination, depending on priority and facility.

The answer is system log daemon.

2. What does the **logger** command do?

The answer is it sends log messages to the syslogd daemon.

3. The **logrotate** command:

- a. Creates new log files
- b. Rotates and cleans up log files
- c. Deletes log files

The answer is rotates and cleans up log files.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

## Solutions for Figure 14-29, "Checkpoint (2 of 2)," on page 14-80

### Checkpoint solutions (2 of 2)

---

4. True or false: Internet access is required for troubleshooting.  
The answer is true.
  
5. If your X server does not start, then the problem might also be:
  - a. The network
  - b. The font server
  - c. A full filesystem
  - d. All of the aboveThe answer is all of the above.
  
6. SUSE Linux provides a repair tool from the Installation menu.  
The answer is repair tool.

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

# Appendix B. Setting up a network installation server

## Estimated time

00:30

## What this unit is about

This appendix describes the procedure to setup a network installation server on Linux.

## What you should be able to do

After completing this unit, you should be able to:

- Setup a network install server.

## Unit objectives

---

After completing this unit, you should be able to:

- Set up a network install server

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure B-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

## Network install server: Overview

---

- Should be a Linux/UNIX server
- Content of all relevant DVDs copied to disk
  - Use a naming scheme that allows multiple versions/distributions to be exported.
    - For example: /export/rhel64, /export/sles11, ...
- Export Method
  - NFS
  - (Anonymous) FTP
  - HTTP
- Optional: PXEBoot

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure B-2. Network install server: Overview

LX158.0

### Notes:

#### Image server

A network install server is typically a Linux/UNIX server, although Windows servers can sometimes also be used, provided that they support a network protocol that the distribution also supports. The content of all relevant CDs or DVDs is copied to disk and made available. It is a good idea to use a naming scheme that allows multiple versions of multiple distributions to be copied to disk.

Almost all network install servers export the installation data through NFS. You can also configure a server to use FTP or HTTP.

#### NFS

If you decide to use NFS, be aware of the fact that the newer distributions typically use NFS version 3, while older distributions typically use NFS version 2. This might lead to

---

compatibility problems, which can be solved easily by forcing the NFS server to always use version 2.

## Anonymous FTP

If you decide to offer anonymous FTP installs, then you need to create your directory structure somewhere in the /var/ftp directory, since the FTP daemon will perform a chroot to this directory when anonymous FTP is requested.

## HTTP

If you decide to offer HTTP installs, you can simply create a symbolic link from your DocumentRoot directory to the directory where your CDs are copied into, as long as FollowSymLinks is set in your web server configuration.

## PXE boot

Setting up a PXEboot server on the network install server, or on another server on the network will allow you to serve the files that are needed for the installer to boot, via the network too. This means you don't need any media such as CDs, DVDs or USB storage to initiate an installation.

***Instructor notes:***

**Purpose** — Discuss the configuration of the network install server.

**Details** — Discuss student notes.

**Additional information —**

**Transition statement** — The installation server must contain the installation data (RPM packages) for the network installation to work.

## Network install server: Copy Data

- Server fileset configuration
  - RHEL: Copy DVD1 completely; **createrepo .**
  - SUSE: Copy DVD 1 completely.
- Other Distros might have specific instructions.
- Server configuration
  - Red Hat
    - Manual (**cp -a /media/dvd/\* /mnt/dvd/.[!.]\*/export/rhel64;**  
**createrepo -g comps.xml .**)
  - SUSE Linux
    - Manual (**cp -a /media/dvd/\* /export/sles11**)  
or with **/sbin/yast2 instserver**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
 US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure B-3. Network install server: Copy Data

LX158.0

### Notes:

### Introduction

We mentioned on the previous visual that you should select a naming convention when setting up your installation server. You should always check the documentation and release notes for a distribution to verify if there is a specific naming convention you must follow as well, or your network installation might not function correctly.

### Important files

After creating the installation directory, you need to copy the contents of the relevant DVDs to that directory. This needs to be done with all preservations of permissions, user names and so forth intact and can usually be done with the **cp -a** command.

The **cp -a** command does not copy hidden files by default. To copy these as well, use **.[!.]\*** in addition to the **\*** wildcard. This will match all hidden files (starting with a dot)

except the .. file (which identifies the parent directory). Red Hat requires this because of the presence of a .discinfo file.

## RHEL configuration

After copying the files of RHEL, you might want to run the Red Hat tool createrepo in the directory where you copied the files to. This allows you to use this fileset as a local installation tree and yum repository. The complete command is:

```
[/export/rhel61]# createrepo -g checksum-comps-rhel6-Server.xml .
```

## SLES configuration

SUSE Linux Enterprise Server (SLES) also provides a tool under Yet another Setup Tool (YaST) to configure your installation server. This menu-driven interface will configure a directory tree with the proper sub-directories and files so that a system running SLES can successfully be used as an installation server.

**Instructor notes:**

**Purpose** — Discuss the copying of data to the network install server.

**Details** —

**Additional information** —

**Transition statement** — The last required step is to setup the network server itself. In this case, we're going to use NFS.

## Network install server: Setup NFS

- Create /etc/exports file with list of directories to export

```
/export/rhel64 * (ro,insecure)
/export/sles11 * (ro,insecure)
/export/files *(ro,insecure)
```

- Enable NFS

```
# service nfs start
# service nfslock start
# chkconfig nfs on
# chkconfig nfslock on
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure B-4. Network install server: Setup NFS

LX158.0

### Notes:

The last required step to setup a network install server is to setup the server program itself. In this particular case, we are going to use NFS.

NFS requires that we create an /etc/exports file containing the list of directories to export, including their security settings and options. The example in the visual exports three directories to all systems on the network, read-only, and also if the request comes from an insecure (higher than 1024) port.

After creating the /etc/exports file you need to start the nfs and nfslock service, and make sure they're started after a reboot.

**Instructor notes:**

**Purpose** — Discuss the final NFS configuration

**Details —**

**Additional information** — Stress once again that it is typically also possible to use (anonymous) FTP and/or HTTP. NFS however normally gives the best performance on a local network. Configuration of FTP and HTTP is outside the scope of this course but is not all that hard to achieve.

The /export/files directory that's in the visual will later be used to contain our Kickstart/AutoYaST files.

**Transition statement** — Once NFS is up and running we're done with all the required bits, and can boot the client. This can be done using physical media, such as a CD, DVD or USB key, but it's a lot more convenient to use PXEboot.

## PXEboot overview

- Intel “Preboot eXecution Environment” standard
- Implemented in PC BIOS
- Two step process:
  - DHCP Request; Reply should include “filename” and “next-server”
  - Retrieve “filename” from “next-server” via TFTP and execute
- In a Linux environment, “filename” normally points to the pxelinux.0 boot loader from the syslinux RPM
- pxelinux.0 checks for pxelinux.cfg directory via TFTP, retrieves configuration file based on GDID, MAC address, IP address or “default”
- PXELinux configuration file may list multiple distributions to boot, boot from hard disk and so on

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure B-5. PXEboot overview

LX158.0

### Notes:

PXEboot is an Intel standard introduced in 1999. It is implemented in virtually any PC BIOS on the market today, and is the default protocol that's being used when the PC is forced to boot from a network card.

When the PC boots from the network card it will first try to obtain an IP address for itself using the DHCP protocol. But there are some variations to the base DHCP protocol:

- The DHCP Discover and Request package will include some specific PXEboot information. This can be used by the DHCP server to craft a specific response package.
- The client will only accept DHCP offers and replies that contain, at least, a “filename” option. In most cases, a “next-server” option is required as well, unless your DHCP server happens to be the network install server.

Once the PXEboot client has obtained a valid IP address, filename and next-server from the DHCP server, it will use the TFTP protocol to contact this next-server, and download that file from it. This file is then possibly checked, stored in RAM and executed.

---

In a Linux environment, the file to be downloaded is normally pxelinux.0, which is part of the syslinux RPM. Syslinux is a series of tools that allows for booting via PXE, from CD/DVD, floppies and a few other methods.

PXELinux also contacts the TFTP server and checks for a configuration file. This configuration file then lists the various boot options that are available for that particular machine.

***Instructor notes:***

**Purpose** — Give an overview of the PXEboot process

**Details** —

**Additional information** —

**Transition statement** — Let's look at the details. First, the DHCP server configuration.

## PXEboot DHCP considerations

- Sample /etc/dhcp/dhcpd.conf file:

```

subnet 192.168.2.0 netmask 255.255.255.0 {
    range 192.168.2.100 192.168.2.199;
    default-lease-time 3600;
    max-lease-time 4800;
    option routers 192.168.2.254;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.2.254;

    host linux {
        hardware ethernet 00:c0:29:60:e7:9d;
        fixed-address 192.168.2.201;
        filename "pxelinux.0";
        next-server 192.168.2.254;
    }
}

```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure B-6. PXEboot DHCP considerations

LX158.0

### **Notes:**

As said earlier, the DHCP server needs to provide the PXEboot client with a “filename” and a “next-server” option, in addition to normal network parameters like IP address, netmask, router addresses and so forth.

The visual above shows the two lines that are needed in the default (ISC) DHCP server that comes with Linux. For other DHCP servers, the configuration may be different, but as long as these two options are added, things should work.

## **Instructor notes:**

**Purpose** — Discuss the DHCP server configuration.

### **Details —**

**Additional information** — This is a very simple example. The ISC DHCP server allows for much more complex configurations. As an example, you could configure the ISC DHCP server to only supply these two options when certain other PXEboot options are present in the DHCP clients DISCOVER and REQUEST packets.

You don't have to use the ISC DHCP server. You can get the exact same results by including these options in, for instance, a Microsoft DHCP server.

**Transition statement** — Next step is that this filename needs to be downloaded via TFTP from the next-server. Let's see.

## PXEboot TFTP considerations

- TFTP server should be installed and configured
  - Usually runs under xinetd
- Store all relevant files in /var/lib/tftpboot
  - /var/lib/tftpboot/pixelinux.0
  - /var/lib/tftpboot/pixelinux.cfg/default
  - /var/lib/tftpboot/boot.txt
  - /var/lib/tftpboot/rhel64/vmlinuz
  - /var/lib/tftpboot/rhel64/initrd.img
  - /var/lib/tftpboot/sles11/linux
  - /var/lib/tftpboot/sles11/initrd

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure B-7. PXEboot TFTP considerations

LX158.0

### **Notes:**

Once the client successfully obtained the DHCP parameters, it will try to download the “filename” from the “next-server” via TFTP. The server therefore needs to have the TFTP server installed, and all relevant files made available via TFTP.

If you install the TFTP server product, it typically allows read-only access to all files in the directory /var/lib/tftpboot by default.

This directory should be filled with:

- The files needed for PXELinux to work. At the very least, you will need:
  - pixelinux.0
  - pixelinux.cfg/default
  - Any files that are referenced by pixelinux.cfg/default
- The “kernel” and “initrd” files from the distributions you’d like to be able to install. These files are taken from the distribution CD or DVD.

The easiest way of locating these files is to look on the distribution CD or DVD for a file called isolinux.cfg. This is the configuration file that manages the boot process if a system is booted from this CD or DVD. In this file (which has the same format, by the way, as pxelinux.cfg/default, since pxelinux and isolinux are direct cousins) you will find references to the kernel and initrd files, and possibly any append statements that are required.

Simply copy over these kernel and initrd files to a directory structure within the /var/lib/tftpboot directory structure, and copy over the stanzas from the isolinux.cfg file to pxelinux.cfg/default and you're done.

**Instructor notes:**

**Purpose** — Discuss the TFTP server configuration.c

**Details** —

**Additional information** —

**Transition statement** — Let's take a closer look at the PXELinux configuration file.

## PXELinux Configuration file

- Sample /var/lib/tftpboot/pixelinux.cfg/default file:

```

DISPLAY boot.txt
PROMPT 1
TIMEOUT 0
DEFAULT localboot

LABEL localboot
LOCALBOOT 0

LABEL rhel64
KERNEL rhel64/vmlinuz
INITRD rhel64/initrd.img

LABEL sles11
KERNEL sles11/linux
INITRD sles11/initrd

```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure B-8. PXELinux Configuration file

LX158.0

### **Notes:**

Once pxelinux.0 starts, it will try to download a configuration file from the TFTP server.

Since it should be possible to use different configuration files per machine, the pxelinux.0 process will look for a large number of files. All these files should be located in the pxelinux.cfg directory:

1. pxelinux.0 will first look for a file named after the GUID, such as pxelinux.cfg/0b90b601-45c8-11cb-807b-a21d22c3e426. The GUID or Globally Unique Identifier is a number which is uniquely assigned to the clients BIOS.
2. If that file named after the GUID is not available, it will look for a file named after the clients MAC address, such as pxelinux.cfg/01-00-d0-59-b5-5a-12.
3. If that file is not available, it will look for a file named after the clients DHCP IP address, in hexadecimal format, such as pxelinux.cfg/C0A802CA. If that file is not available, it will look for the same file, but with an increasingly large subnet size: pxelinux.cfg/C0A802C, pxelinux.cfg/C0A802 and so forth.

4. If none of these files are available, it will download a file called pxelinux.cfg/default. If that file is not available, PXELinux will generate an error.

All these files have the same syntax. The details of this syntax can be found in the documentation that comes with the syslinux package. The file as shown in the visual means the following:

- Download the file boot.txt via TFTP and display it.  
This file should be stored on the server as /var/lib/tftpboot/boot.txt and is essentially a text file. Several control characters, for instance to erase the screen or change the font colors, are allowed.
- Prompt the user for the boot choice to make.
- Do not time out.
- If the user presses Enter without making a specific choice, execute the commands associated with the label localboot.
- If the user enters localboot, perform a local boot (from the first local hard disk).
- If the user enters rhel61, load the kernel file rhel61/vmlinuz and initrd file rhel61/initrd.img via TFTP, and execute the kernel file.
- If the user enters sles11, load the kernel file sles11/linux and initrd file sles11/initrd via TFTP, and execute the kernel file.

**Instructor notes:**

**Purpose** — Discuss the pxelinux.cfg configuration files.

**Details** —

**Additional information** —

**Transition statement** — As the last part of PXEboot, let's see how we start a PXEboot installation on the client.

## Starting a PXEboot installation

- Boot system normally
  - Power on
  - In BIOS menu select to boot from Network Adapter
- Wake-On-Lan (WOL)
  - Special packet sent over the network to the adapter when the system is powered off
  - Linux tool to send WOL packet: **ether-wake <MAC>**
  - Will normally initiate a PXEboot (depends on BIOS settings)

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure B-9. Starting a PXEboot installation

LX158.0

### Notes:

There are two ways in which you can start a PXEboot installation.

The most common method is to power on the client with the power-on button. When the BIOS prompt appears, you interrupt the boot process (typically with F1, Del or F12) and force the BIOS to boot from the network adapter.

The second method is by noting down the MAC address of the ethernet adapter and sending a specially crafted Wake-On-LAN (WOL) packet to this MAC address. If WOL-enabled, the network adapter will now power-on the system and the BIOS will enter a special “network boot” boot list, where the network adapter is usually the first device listed.

WOL packets can be sent from a Linux system using the ether-wake tool.

***Instructor notes:***

**Purpose** — Discuss client startup in a PXEboot environment.

**Details** —

**Additional information** —

**Transition statement** — That's it for this appendix.

## Unit summary

Having completed this unit, you should be able to:

- Set up a network install server

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure B-10. Unit summary

LX158.0

### Notes:

Having completed this unit, you should be able to:

- Understand network install servers are convenient means of software distribution for doing both upgrades and installs
- Understand a network install server typically exports multiple versions of multiple distributions through NFS, FTP, or HTTP

***Instructor notes:***

**Purpose —**

**Details —**

**Additional information —**

**Transition statement —**

# Appendix C. Networking

## Estimated time

01:00

## What this unit is about

This unit describes the Linux networking stack, and the basic commands and files to configure the most common networking setups.

## What you should be able to do

After completing this unit, you should be able to:

- Configure ethernet adapters
- Configure Wifi adapters
- Configure bridging, bonding and VLANs
- Configure IPv4 and IPv6
- Setup static and dynamic routing
- Setup DNS clients
- Discuss NetworkManager

## Unit objectives

---

After completing this unit, you should be able to:

- Configure ethernet adapters
- Configure Wifi adapters
- Configure bridging, bonding and VLANs
- Configure IPv4 and IPv6
- Setup static and dynamic routing
- Setup DNS clients
- Discuss NetworkManager

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

---

Figure C-1. Unit objectives

LX158.0

### Notes:

***Instructor notes:***

**Purpose** — Define unit objectives.

**Details** —

**Additional information** —

**Transition statement** — Let's start with an overview of Linux Networking.

## Linux networking

- Very versatile networking stack, covering all layers of the OSI model
- Hardware: Ethernet, Token Ring, serial/modem, ISDN, frame relay, ATM, X.25 and many more
- Protocols: TCP/IP, DECnet, IPX and many more
- Client and server applications: DNS, telnet, FTP, SSH, WWW, SMTP and many others
- See the kernel source "net/" directory for an overview of hardware and protocol support

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-2. Linux networking

LX158.0

### Notes:

The Linux Networking stack is very versatile and complete, covering all layers of the OSI model.

A lot of networking hardware is supported: ethernet adapters, token ring, serial/modem connections, ISDN, frame relay, ATM, X.25 and many more.

Likewise, a whole series of networking protocols are supported. Obviously this includes the full TCP/IP suite, but if necessary Linux will also support DECnet, IPX and many others.

Finally, Linux has support for practically all client and server applications that are currently in use, such as DNS, telnet, FTP, SSH, WWW, SMTP and many more.

For an example of the protocols that are supported by the kernel, take a look at the net/ directory in the kernel sources.

In this unit we will cover the most common client and server setups: A machine with one or more ethernet or Wifi adapters, IPv4 and/or IPv6. We will also look at a few protocols that are integral to the operation of TCP/IP, particularly DHCP and DNS.

**Instructor notes:**

**Purpose** — Introduce the Linux networking stack.

**Details** — Stress that we will never be able to cover everything in this unit. We will limit ourselves to the most common configurations for both clients and servers.

**Additional information —**

**Transition statement** — Okay, let's look at configuring an ethernet adapter first.

## Ethernet configuration

- Low-level configuration done through **ethtool**
  - Link state
  - Link speed, duplex autonegotiation parameters
  - Firmware dumps, updates
  - Low-level performance tuning
  - Statistics

```
# ethtool eth0
Settings for eth0:
    Supported ports: [ TP ]
    Supported link modes:  10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full
    Supported pause frame use: No
    Supports auto-negotiation: Yes
    Advertised link modes:  10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full
    Advertised pause frame use: No
    Advertised auto-negotiation: Yes
    Speed: 1000Mb/s
    Duplex: Full
    ...
...
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-3. Ethernet configuration

LX158.0

### Notes:

Once an ethernet adapter is detected in the system, some configuration needs to be done. At the link level, the adapter has to detect that a link is present and then negotiate the speed and duplex settings with the machine or switch at the other end of the cable. Furthermore you may want to update adapter firmware, perform some low-level performance tuning and retrieve statistics from the adapter. All these tasks, and more, are done through the **ethtool** program.

The **ethtool** program is particularly useful if you suspect link state or negotiation issues. It will tell you whether the link is up, and what speed/duplex setting was negotiated with the machine or switch at the other end. A very common issue is that one of the machines or switches is set to "auto/auto", while the other end is set to a fixed value such as 100/Full Duplex. In this case, autonegotiation fails. In that case, an ethernet adapter set to "auto/auto" will be able to detect the proper link speed, but will not be able to detect the duplex setting. Because of the latter, it will fall back to Half Duplex. This is incredibly bad for performance, leading to an effective loss of 75-90% of theoretical throughput.

Practically all modern switches and ethernet cards, even at the low end of the market, are Full Duplex capable. So any time you see a Half Duplex setting in the output of **ethtool**, this is a subject for investigation.

### Red Hat Configuration

For adapters that need a permanent deviation from the default settings (typically "auto/auto"), you can add the ETHTOOL\_OPTS line to the corresponding /etc/sysconfig/network-scripts/ifcfg-<interface> file. For example:

```
ETHTOOL_OPTS="autoneg off speed 100 duplex full"
```

### SUSE Configuration

The SUSE configuration is very similar to Red Hat, apart from the fact that the option is called ETHTOOL\_OPTIONS:

```
ETHTOOL_OPTIONS="autoneg off speed 100 duplex full"
```

***Instructor notes:***

**Purpose** — Discuss ethernet adapter configuration.

**Details** —

**Additional information** —

**Transition statement** — Next, wifi.

# Wifi configuration

- Low-level configuration done through **iwconfig**
  - ESSID
  - Channel selection, speed
  - Transmit power, sensitivity
  - Encryption settings
- Other Wifi tools: **iwlist**, **iwspy**

```
# iwconfig wlan0
wlan0      IEEE 802.11abg  ESSID:"IBMTRAINING"
           Mode:Managed  Frequency:2.412 GHz  Access Point: 6C:FD:B9:33:FA:BA
           Bit Rate=1 Mb/s  Tx-Power=15 dBm
           Retry  long limit:7    RTS thr:off    Fragment thr:off
           Encryption key:off
           Power Management:off
           Link Quality=70/70  Signal level=-34 dBm
           Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
           Tx excessive retries:0  Invalid misc:24    Missed beacon:0
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-4. Wifi configuration

LX158.0

## Notes:

Wifi adapters need to be configured, at the very least, with an ESSID, so that they associate themselves with the correct Access Point. But depending on the network settings, you may also need to supply parameters for channel selection, speed, protocol, transmit power, receive sensitivity and so forth. All this is done with the **iwconfig** tool.

Furthermore, you will most likely also need to setup security/encryption, to prevent others from joining your network, and to prevent snooping on the network. This is covered on the next page.

Other useful tools are:

- **iwlist**, which gives you detailed information from a wireless interface. Particularly the **scanning** option is useful, to detect all APs in range.
- **iwspy**, which shows current signal strength and other dynamic parameters.

## Red Hat and SUSE Configuration

There is some support in Red Hat and SUSE for permanently storing parameters such as the mode, SSID and WEP key in the /etc/sysconfig/network-scripts/ifcfg-<interface> files

but as the majority of Wifi users do so from a (mobile) laptop, it is preferable to use NetworkManager instead. NetworkManager will be covered later in this unit.

**Instructor notes:**

**Purpose** — Discuss basic Wifi configuration

**Details** —

**Additional information** —

**Transition statement** — Let's look at Wifi security.

## Wifi security setup

- WEP (Wired Equivalent Privacy)
  - Old standard (deprecated), easy to setup but also easy to break
  - Configured with **iwconfig wlan0 key <key>**
- WPA, WPA2
  - Managed through **wpa\_supplicant** daemon
  - Configuration file: /etc/wpa\_supplicant/wpa\_supplicant.conf
  - Configuration file may contain keys
  - wpa\_cli: command line interface to **wpa\_supplicant**

```
# wpa_cli status
Selected interface 'wlan0'
bssid= 6C:FD:B9:33:FA:BA
ssid=IBMTTRAINING
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.2.3
```

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-5. Wifi security setup

LX158.0

### Notes:

The first available protocol to secure a Wifi connection was the Wired Equivalent Protocol (WEP). WEP uses a secret key that is common across all participants within an SSID domain. This secret key is configured with the command **iwconfig wlan0 key <key>**. However, the WEP protocol turned out to be riddled with bugs, and several tools are currently available on the Internet that are able to crack this encryption within minutes. WEP is considered deprecated and should no longer be used.

Far better security is offered by the Wifi Protected Access (WPA) and particularly the WPA2 protocol. WPA2 cannot be configured on the adapter alone, but requires the **wpa\_supplicant** daemon to be running. This daemon is configured through a configuration file /etc/wpa\_supplicant/wpa\_supplicant.conf, and this file will contain the keys to be used for the various SSID domains that you will want to connect to.

**wpa\_cli** is a tool that allows you to manage the **wpa\_supplicant** daemon from the command line. It will allow you to retrieve the status of the connection, connect to a new SSID domain, supply the necessary passwords (so you don't have to store them in wpa\_supplicant.conf) and a lot of other things.

Note that when using **NetworkManager** (which will be discussed later in this unit), **wpa\_supplicant** should NOT be started through the normal SysV init scripts (or upstart or systemd for that matter). Instead, **wpa\_supplicant** will be started by **NetworkManager** if and when required. In such a situation, you will also find that **wpa\_cli** will not have access to the socket to communicate with **wpa\_supplicant**. If you need to manage **wpa\_supplicant**, you need to do this through **NetworkManager**.

**Instructor notes:**

**Purpose** — Discuss Wifi security

**Details** —

**Additional information** —

**Transition statement** — We can also use Linux to create access points.

## Creating a Wifi access point

- Done through **hostapd** daemon
- Configuration file /etc/hostapd/hostapd.conf
  - Interface to use
  - SSID
  - Security settings
- Runs as a daemon
  - **service hostapd start**
  - **chkconfig hostapd on**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-6. Creating a Wifi access point

LX158.0

### Notes:

It is also possible for a Linux system to function as an access point, and to provide network access to other clients this way. This functionality is implemented by the **hostapd** daemon.

You should first configure this daemon through the configuration file /etc/hostapd/hostapd.conf. In this file, amongst other things, you will need to identify the adapter to use, the ESSID name to use, and how security should be configured. You can then start the daemon.

An example hostapd.conf file is shown below:

```
ctrl_interface=/var/run/hostapd  
ctrl_interface_group=wheel
```

```
interface=wlan0
```

```
hw_mode=b
```

```
channel=1
```

```
ssid=IBMTRAINING
```

```
wpa=3
```

```
wpa_key_mgmt=WPA-PSK
```

```
wpa_pairwise=TKIP
```

```
rsn_pairwise=CCMP
```

```
wpa_passphrase=IBM4YOU
```

Note that **hostapd** only provides the Wifi link-level access point feature. Most likely you will also want to configure a DHCP server, routing and DNS to make your access point useful.

**Instructor notes:**

**Purpose** — Discuss how to create a Linux-based Wifi Access Point

**Details** —

**Additional information** —

**Transition statement** — Okay, that's enough about Wifi. Let's look at a slightly higher level in the protocol stack, and see what we can do with these ethernet and Wifi adapters.

## Bonding ethernet adapters

- Bonding: "Gluing" multiple adapters into one
  - Load balancing; bandwidth aggregation
  - Redundancy
- Various types available; some depend on switch capability to support 802.1ad standard; see student notes
- Create bonding interface bond0:
  - **modprobe bonding mode=<mode> miimon=100**
- Add physical adapters to bond0:
  - **ifenslave bond0 eth0 eth1**
- Can now use bond0 as a regular ethernet interface
- View bonding status with **cat /proc/net/bonding/bond0**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-7. Bonding ethernet adapters

LX158.0

### Notes:

"Bonding" is the technical term for combining two or more independent ethernet adapters into a single logical adapter, allowing those adapters to act as one. Depending on the settings, this can be done to increase bandwidth, increase redundancy or both.

There are seven bonding methods available:

- 0: **balance-rr**
- 1: **active-backup**
- 2: **balance-xor**
- 3: **broadcast**
- 4: **802.3ad**
- 5: **balance-tlb**
- 6: **balance-alb**

For a full explanation of these modes, see the documentation on channel bonding, for instance at:

[https://access.redhat.com/site/documentation/en-US/Red\\_Hat\\_Enterprise\\_Linux/6/html/Deployment\\_Guide/sec-Using\\_Channel\\_Bonding.html](https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/sec-Using_Channel_Bonding.html)

Note that some bonding methods use the 802.1ad protocol. When using these methods it is vitally important that the switches used also support this protocol, and are configured to enable it on the ports where your Linux system attaches.

Bonding is implemented by the "bonding" kernel module. If you only need a single bonding interface, all you need to do is load the bonding module. This will create a device "bond0" which can then be configured and used as the bonding interface.

If you need multiple bonding interfaces, you need to load the module for each of them. This is done by creating an /etc/modprobe.d/bonding.conf file, which then contains the following line for each bonding interface:

```
alias bond0 bonding
```

For a full manual configuration you would also want to add a line like the following to this file:

```
options bond0 mode=<mode> ...
```

However, both Red Hat and SUSE prefer that you add the bonding mode and options to the ifcfg-<device> file, which is covered below.

In addition to the bonding mode, you also need to specify the method how Linux determines that a certain physical adapter is usable. There are essentially two methods supported:

- MII: With this method Linux uses **mii link monitoring** to determine if an adapter is usable, by looking at the link state of the adapter: If the link is active (the cable is inserted and the device at the other end is alive), then the adapter can be used.

This method has very minimal impact and generates no network traffic whatsoever. The disadvantage is that it will not detect an "upstream" error in the path.

Some network switches have the ability to detect "upstream" errors in the path, and if an error is detected, bring the link to your Linux machine down. If you are using mii link monitoring, this is a very good idea to use.

When using mii link monitoring, the only parameter to configure is the **miimon** parameter, which sets the monitoring interval in milliseconds. 100 is a very reasonable value.

- ARP: With this method Linux will actively do an ARP request to up to 16 stations via each adapter. Any adapter that subsequently receives ARP replies is considered active.

This method has a little more impact compared to mii link monitoring, and will generate some network traffic. It has the advantage that it can detect errors that are slightly more

"upstream", compared to mii link monitoring. But as ARP requests and replies cannot traverse a router, they can only detect link errors up to and including the first router.

Once the bonding interface has been setup, you can add the adapters to the bonding interface. This is done with the **ifenslave** command. Note that the ethernet adapters in a bonding configuration should not be configured with their own IP address. Instead, it is the bonding adapter which will eventually get an IP address.

You can view the state of your bonding adapters with **cat /proc/net/bonding/bond0**.

## Red Hat Configuration

On a Red Hat system, bonding is configured through the use of the /etc/sysconfig/network-scripts/ifcfg-<device> files. Here are the important lines in a bonding scenario where eth0 and eth1 together form a bond0 device:

### **ifcfg-eth0:**

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
USERCTL=no
```

### **ifcfg-eth1:**

```
DEVICE=eth1
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
USERCTL=no
```

### **ifcfg-bond0:**

```
DEVICE=bond0
ONBOOT=yes
BONDING_OPTS="mode=<mode> miimon=100"
USERCTL=no
BOOTPROTO=none
IPADDR=192.168.1.1
NETMASK=255.255.255.0
```

## SUSE Configuration

In SLES11, bonding can be configured through YaST2. See the following URL for a description:

<http://www.novell.com/support/kb/doc.php?id=3815448>

Under the covers, SUSE is configured very similar to Red Hat, but some of the options in the ifcfg-bond0 file are named slightly differently, and in SUSE the list of slaves is

enumerated in the ifcfg-bond0 file, instead of listing the master in the ifcfg-eth0 and ifcfg-eth1 files.

***Instructor notes:***

**Purpose** — Discuss ethernet bonding.

**Details** —

**Additional information** —

**Transition statement** — Next, let's look at bridging.

## Bridging ethernet adapters

- Bridging: Layer 2 forwarding of packets (based on MAC address)
- Typically used in virtualized environment
- Create bridge: **brctl addbr br0**
- Add interfaces: **brctl addif br0 eth0**
- Show configuration: **brctl show [<bridge>]**
- Bridging may introduce loops in your network
  - Linux bridging by default participates in 802.1d Spanning Tree Protocol
  - Various **brctl** commands modify STP parameters

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-8. Bridging ethernet adapters

LX158.0

### Notes:

"Bridging" is the technical term that is used to forward network packets towards their destination based on the MAC address (layer-2 address) contained in the packet. This in contrast to "routing", where forwarding is done based on the IP address (layer-3) address.

Bridging in a Linux context is most often used in a virtualized environment, where virtual machines running on a Linux host, need transparent access to the physical network. But Linux can also function as a physical bridge, linking two physical networks together.

Bridging is implemented through the use of a "bridging device", typically called br0, br1 and so forth. The physical adapters that need to be used, are then attached to this bridging device, and this enables the bridging functionality.

A bridge device is created using the **brctl addbr** command, and interfaces are added to the bridge with the **brctl addif** command. Similarly, you can remove interfaces from the bridge with **brctl delif** and remove the bridge device with **brctl delbr**. The **brctl show [<bridge>]** command will give you current information.

Bridging between two physical networks may create loops in your network, and the original Ethernet standard was not able to cope with these loops. For this reason, the Spanning Tree Protocol was added to the Ethernet standard. STP detects these loops and severs a strategic link somewhere in the topology so that the loops are broken, leaving you with a tree network that spans the whole topology - hence the name "Spanning Tree".

There are several **brctl** commands that deal with the STP parameters. However, bridging on Linux is only rarely used to bridge physical networks. In most cases bridging will be used to support virtualized environments, and in these situations adding a Linux bridge will not create loops in your network topology. So all STP parameters can be ignored in these situations.

However, if you do use Linux to bridge between physical networks, make sure you take a good look at the STP settings of both your Linux bridge and any other bridges/switches in your network.

Bridging and bonding, as seen on the previous slide, can be configured together. First, create the bonding device bond0, and then use the bond0 device to bridge the br0 bridge to the outside world.

## Red Hat Configuration

To create a bridge br0 on a Red Hat system, create the file /etc/sysconfig/network-scripts/ifcfg-br0, with the following content:

```
DEVICE=br0
TYPE=Bridge
DELAY=0
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=static
IPADDR=192.168.1.1
NETMASK=255.255.255.0
```

Furthermore, if any bridging options (such as those for STP) are required, you can add them to this file in the BRIDGING\_OPTS variable.

For any physical interface that will be attached to this bridge, modify the ifcfg-<device> file as follows:

```
DEVICE=eth0
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
NM_CONTROLLED=no
BRIDGE=br0
```

Once again, note that it is the "master" device, in this case the bridge, that gets the IP address, not any of the "slave" devices. In the example above the IP address was configured statically, but using DHCP is also a legitimate solution.

## SUSE Configuration

In SUSE, a bridging configuration is created through YaST2. The configuration of the files that are created by YaST2 is very similar to the Red Hat setup, although some of the names of the variables may be subtly different. And furthermore, SUSE uses the "BRIDGE\_PORTS" variable in the ifcfg-br0 file to enumerate the physical adapters that make up the bridge, instead of listing the bridge name in the ifcfg-eth0 files.

***Instructor notes:***

**Purpose** — Discuss bridging.

**Details** —

**Additional information** —

**Transition statement** — Next, VLANs.

## Virtual LAN (VLAN) support

- Linux supports 802.1q VLAN technology.
- VLANs are disabled by default: packets arriving on the "base" adapter are supposed to be untagged; packets sent by way of the "base" adapter will not be tagged.
  - The switch will add/remove the VLAN tag, if any, based on the "Port VLAN ID" (PVID) setting for the port
- Add a VLAN device: **vconfig add eth0 5**
  - Creates network adapter **eth0.5**; can be managed as a normal ethernet adapter
  - Packets arriving on eth0 with a VLAN 5 tag will be untagged and sent to eth0.5
  - Packets sent through eth0.5 will be tagged with VLAN 5 tag and sent by way of eth0
  - Requires switch to add VLAN 5 for this port
- Remove a VLAN device: **vconfig rem eth0.5**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-9. Virtual LAN (VLAN) support

LX158.0

### Notes:

Virtual LAN, or VLAN technology, is a standard that allows you to split a single physical network into well over 4000 virtual networks. To each of these virtual networks you can connect a different set of hosts, and the traffic between those hosts will not pass from one virtual LAN to the other, effectively giving the appearance that each set of hosts is connected to a different, isolated physical network.

VLANs are typically implemented in switches, and the hosts connected to these switches are then VLAN-unaware: They do not have to add or remove VLAN headers, and have no idea to which VLAN they are connected.

However, there are several situations where it is advantageous that the host is VLAN aware. Consider for instance the case of a Linux-based router or firewall that needs to forward traffic from one VLAN to another. Or in case of virtualization, where one Linux host supports multiple VM guests, who need to connect to different VLANs.

For these reasons, Linux supports the 802.1q VLAN standard. This standard essentially adds an extra header to the Ethernet frame. The most important field in this header is a 12-bit VLAN ID, which identifies the VLAN number that this packet belongs to.<sup>1</sup>

VLANs in Linux are implemented as follows:

The "base" adapter, such as ent0, is always an "untagged" or VLAN-unaware adapter. This means that any untagged packet (an Ethernet packet without a VLAN header) that is received on the physical adapter, is forwarded to applications connected to this adapter/IP address. Similarly, any packet that is sent out through this adapter will leave your Linux system "untagged". If VLAN technology is used, it is the responsibility of the network switch to tag these packets with the proper VLAN ID before forwarding it. (The VLAN ID that is used for these packets is typically called the "Port VLAN ID" or PVID.)

On top of this "base" adapter you can create VLAN-aware adapters. This is done with the **vconfig add** command. For example, **vconfig add eth0 5** will create a VLAN adapter "eth0.5", which is an ethernet adapter for VLAN 5. Any "tagged" packets arriving on the base adapter with VLAN ID 5 will be stripped from their VLAN header and then forwarded to eth0.5. Similarly, any traffic sent through the eth0.5 adapter will be tagged with a VLAN header and VLAN ID 5, before sending it out through the base adapter ent0.

Tagged packets arriving on eth0, with a VLAN ID for which no VLAN adapter is configured, will be silently discarded.

You can create as many VLAN adapters as you want on top of a single ethernet adapter. Furthermore, VLAN adapters can also be created on top of bridged and/or bonded adapters.

## Red Hat Configuration

To configure a VLAN interface in Red Hat, you leave the base configuration (e.g. eth0) alone. After all, even in a VLAN environment you can still use the base adapter for untagged packets, so that adapter needs to be used normally, and needs to get an IP address.

To add a VLAN adapter eth0.5, create a file /etc/sysconfig/network-scripts/ifcfg-eth0.5 as follows:

```
DEVICE=eth0.5
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.1.1
NETMASK=255.255.255.0
USERCTL=no
VLAN=yes
```

You can create as many of these files as required.

## SUSE Configuration

On a SUSE system VLANs are configured through YaST2. YaST2 will configure the ifcfg-files for you, very similar to what Red Hat does.

<sup>1</sup> Note that  $2^{12}$  equals 4096, so theoretically you can create up to 4096 VLANs. However, some of these numbers (including 0, 1 and 4095) are reserved, or have a special meaning to some network switches, so these cannot be used.

Furthermore, the 802.1q header adds 4 bytes in total to the Ethernet packet. If your network infrastructure is limited to an MTU size of 1500, you may want to reduce the MTU of the VLAN adapter to, for instance 1496 bytes with the command **ifconfig eth0.5 mtu 1496**.

## Tracing VLAN Traffic

VLANs are notoriously hard to setup properly, and it may help to run a network trace on both the "base" adapter (e.g. eth0) and the VLAN adapter (e.g. eth0.5) while troubleshooting.

Using **tcpdump** with the **-e** flag will show extended header information, including the VLAN header (if any). This will allow you to see if incoming and outgoing packets are properly tagged.

***Instructor notes:***

**Purpose** — Discuss VLANs

**Details** —

**Additional information** —

**Transition statement** — Okay, that's it for OSI layers 1 and 2. Let's now look at TCP/IP configuration. First, IPv4.

# Configuring IPv4 addresses

- Static configuration:
  - **ifconfig eth0 192.168.1.1 netmask 255.255.255.0**
- DHCP client:
  - **dhclient eth0**
  - Various options available to modify selection of DHCP offers, set client identifier and so forth
- Activating an adapter with an IPv4 address is included in the startup scripts of the distribution; see student notes.
- View current configuration:
  - **ifconfig eth0**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-10. Configuring IPv4 addresses

LX158.0

## Notes:

Once all your network devices have been setup with the proper link speed, wireless parameters, bridging, bonding and VLANs, it's time to setup TCP/IP. Linux allows for both static and dynamic configuration of IPv4 addresses.

Static configuration is done with the **ifconfig** command. You need to specify the IP address and the netmask, and may also specify parameters such as the MTU size.

Linux also comes with a DHCP client, which can be started on the interface and will request a DHCP address from a DHCP server. It will then run as a client-side daemon, renewing the lease when required.

Note that there are several DHCP client code bases available, and not all distributions use the same code base. In some cases the syntax is **dhclient eth0** while others use **dhclient -i eth0**.

The current configuration of a network adapter can be viewed with **ifconfig [<adapter>]**.

## Red Hat Configuration

Basic IPv4 addressing is usually configured during the installation. The **system-config-network** tool will allow you to modify your configuration afterwards, assuming you're not using NetworkManager.

The configuration ends up in the file /etc/sysconfig/network-scripts/ifcfg-<device>. The BOOTPROTO variable determines whether DHCP or Static addressing is used.

For a DHCP configuration, your file needs to contain this:

```
BOOTPROTO="dhcp"
```

For a static configuration, your file needs to contain this:

```
BOOTPROTO="static"  
IPADDR=192.168.1.1  
NETMASK=255.255.255.0
```

You can also set BOOTPROTO="none" for adapters that do not need to be configured with an IP address, for instance because they will become part of a bonding or bridging solution.

Three more parameters can be useful in the ifcfg-<device> file. ONBOOT determines if the device is activated during system boot, USERCTL determines if a non-root user can activate/deactivate this device, and NM\_CONTROLLED determines if this device can be controlled through NetworkManager.

## SUSE Configuration

IP addresses are configured through YaST2. The resulting files are very similar to the files used by Red Hat.

***Instructor notes:***

**Purpose —** Discuss IPv4 addresses.

**Details —**

**Additional information —**

**Transition statement —** Let's also look at IPv6.

# Configuring IPv6 addresses

- When enabled, all adapters will automatically receive a "link local" address
  - Prefix `fe80::/64`, followed by MAC address
- Further addresses can be configured with **ifconfig add**
- Remove addresses with **ifconfig del**
- View current configuration with **ifconfig**
- **dhclient** also has IPv6 support
- Other useful programs when working with IPv6:
  - **ping6**
  - **tracepath6**
  - **traceroute6**

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-11. Configuring IPv6 addresses

LX158.0

## Notes:

If IPv6 is enabled on an adapter, that adapter will automatically receive a "link local" address. This is an address that can only be reached from the local network, and is not routable. It consists of the prefix "fe80::/64", followed by what is essentially the MAC address of the adapter, with some bits added. This way, a local IPv6 network will come into existence without any form of configuration whatsoever, while still maintaining uniqueness across the IPv6 addresses on a network.

Further IPv6 addresses, for instance site-local or globally unique addresses, can be added with **ifconfig add**, and removed with **ifconfig del**. **ifconfig** by itself shows the current configuration, and the DHCP client, **dhclient** also has support for IPv6.

Other useful programs in an IPv6 environment are **ping6**, **tracepath6** and **traceroute6**. These perform the same function as their IPv4 counterparts.

## Red Hat Configuration

On a Red Hat system, there is limited support for configuring IPv6 in Anaconda. After installation, IPv6 can be enabled and disabled globally with the `NETWORKING_IPV6`

variable in /etc/sysconfig/network. If enabled there, IPv6 can further be enabled and disabled per adapter with the IPV6INIT variable in the adapter configuration file /etc/sysconfig/network-scripts/ifcfg-<adapter>.

In that same file, you can add an IPv6 address as follows:

```
IPV6ADDR="fec0::ce52:afff:fece:8903/64"
```

If you need to add more than one IPv6 address, you can alternatively use the IPV6ADDR\_SECONDARIES variable to list multiple addresses.

IPv6 does a lot of auto configuration by default. This is managed by various shell variables in the ifcfg-<adapter> file as well. These variables are explained in the header of the file /etc/sysconfig/network-scripts/ifup-ipv6.

## SUSE Configuration

On a SUSE system, IPv6 is configured through YaST2. The resulting files are very similar to Red Hats setup.

**Instructor notes:**

**Purpose** — Discuss IPv6 configuration.

**Details** —

**Additional information** —

**Transition statement** — Okay, next is routing.

# Routing

---

- Enable IPv4 and/or IPv6 forwarding:
  - `echo 1 > /proc/sys/net/ipv4/ip_forward`
  - `echo 1 > /proc/sys/net/ipv6/conf/all/forwarding`
  - Permanent configuration in `/etc/sysctl.conf`
- Add static routes:
  - `route add default gw 192.168.2.254`
  - `route add 192.168.3.0 netmask 255.255.255.0 gw 192.168.2.254`
- Delete static routes with `route del`
- Show static routes with `route, netstat -r`
- Dynamic routing requires routing daemons:
  - `routed`: RIP only
  - `BIRD, zebra, quagga`: RIP, RIPv2, OSPF, BGP4 and many others

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-12. Routing

LX158.0

## Notes:

Linux can be used as a router in a network, and Linux can use other routers to route traffic to its destination. Most Linux systems will get by with just a single "default router" or "default gateway", which is typically configured during the installation, or whose IP address is obtained from the DHCP server.

For more complex situations you may want to setup Linux as a router itself, and configure multiple static routes.

Configuring Linux as a basic routers, where traffic that is received but does not have that Linux system as the destination, is forwarded, can be done on the fly:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

You can also setup IP forwarding on a per-adapter basis in a similar way.

To make these changes permanent, add the following lines to your `/etc/sysctl.conf` file:

```
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
```

Note however that distributions have their own method of turning IP forwarding on, usually contained in one of the ifcfg-<device> scripts. In that case, do not modify the sysctl.conf file.

Static routes are added with the **route add** command. Similarly, static routes can be deleted with the **route del** command. The current routing table can be viewed with the **route** or **netstat -r** commands. Use the **-n** option to prevent reverse DNS lookups.

Linux supports various routing daemons, through which dynamic routing can be done. Examples include **routed**, **zebra** and **quagga**.

## Red Hat Configuration

If you use dynamic addressing (DHCP) then the default route and any supplementary routes will be configured on-the-fly by the DHCP client.

For static addressing and static routing, you can set a global gateway in /etc/sysconfig/network as follows:

```
GATEWAY=192.168.1.254
```

You can also configure this variable in the adapter files ifcfg-<adapter>. This will configure that gateway if and only if that adapter is "up". If multiple gateways are specified this way, then the "GATEWAYDEV" device listed in /etc/sysconfig/network will take precedence.

For additional static routes, you will need to create files /etc/sysconfig/network-scripts/route-<interface>. These files then contain the list of routes to activate when that particular interface is brought up. These files will look like this:

```
ADDRESS0=10.10.10.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.1
ADDRESS1=172.16.1.0
NETMASK1=255.255.255.0
GATEWAY1=192.168.0.2
```

## SUSE Configuration

In SUSE, static routing is configured through YaST2. The configuration eventually ends up in the files /etc/sysconfig/network/routes or /etc/sysconfig/network/ifroute-<interface>. This file has the following syntax:

```
DESTINATION          GATEWAY NETMASK   INTERFACE [ TYPE ] [ OPTIONS ]
DESTINATION          GATEWAY PREFIXLEN INTERFACE [ TYPE ] [ OPTIONS ]
DESTINATION/PREFIXLEN GATEWAY -           INTERFACE [ TYPE ] [ OPTIONS ]
```

***Instructor notes:***

**Purpose** — Discuss routing

**Details** —

**Additional information** —

**Transition statement** — Next, DNS.

## Name resolution

- RHEL: Hostname configured in /etc/sysconfig/networking

```
HOSTNAME="system.example.com"
```

- SLES: Hostname configured in /etc/HOSTNAME

```
system.example.com
```

- List of hostnames configured in /etc/hosts

```
127.0.0.1    loopback.locaLdomain loopback localhost
192.168.1.1  system.example.com system
```

- DNS client configuration in /etc/resolv.conf

```
search example.com
nameserver 192.168.2.254
nameserver 192.168.2.253
```

- DNS server configuration done through BIND (named) server

- Configuration file /etc/named.conf
- Zone files stored in /var/named or /var/lib/named

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-13. Name resolution

LX158.0

### Notes:

The system hostname is stored in /etc/sysconfig/network (Red Hat) or /etc/HOSTNAME (SUSE).

The file /etc/hosts contains a list of IP addresses, followed by the corresponding hostname(s) of other systems that are important enough that you don't want to rely on a DNS server for name resolution.

The Linux DNS client configuration is stored in /etc/resolv.conf.

All these files can be created during the installation, manually, or through tools such as Red Hat system-config-network or SUSE YaST2.

You can also use Linux as a DNS server. There are multiple DNS server products available for Linux, but the most commonly used is the BIND (named) server. The configuration file for this daemon is in /etc/named.conf, and the zone files will be stored in /var/named (Red Hat default) or /var/lib/named (SUSE default).

***Instructor notes:***

**Purpose** — Discuss DNS

**Details** — Configuring a DNS server is outside the scope of this course.

**Additional information** —

**Transition statement** — Let's look at firewalls next.

# Firewalling

- Packet filtering, NAT done through **iptables** kernel module
  - Modular design
  - Includes modern features such as stateful filtering, deep packet inspection, header rewrite, port knocking and many others
- Alter kernel tables with **iptables** tool
  - `iptables -P INPUT DROP`
  - `iptables -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT`
- Save and restore kernel tables with **iptables-save** and **iptables-restore**
  - Typically built into distribution startup scripts

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-14. Firewalling

LX158.0

## Notes:

Linux has very extensive firewalling capabilities, implemented through the **iptables** framework. **iptables** is a modular series of kernel modules that provide for stateless and stateful filtering, deep packet inspection, header rewrite, port knocking and many other features.

The tool that allows you to interface with the iptables kernel modules is called **iptables** as well. The visual shows a very simple example where a default policy of "DROP" is set on any incoming network packets. An exception to this policy is then made to "ACCEPT" any packets that have a destination of TCP port 22.

The current configuration of your iptables kernel setup can be saved to a text file using **iptables-save**. This can then be restored later on with **iptables-restore**.

## Red Hat Configuration

Red Hat installs some very simple firewall rules as part of the installation. These rules are stored in /etc/sysconfig/iptables, in iptables-save format. You can edit this file by hand, or use the **iptables** tool to modify the actual kernel tables. You can then save the actual,

---

current kernel tables to this file by using **iptables-save > /etc/sysconfig/iptables**, or with **service iptables save**.

The System V init script /etc/init.d/iptables will ensure your /etc/sysconfig/iptables file is read on startup, and otherwise allows you to activate and deactivate the iptables rulesets as if they were a normal daemon.

### SUSE Configuration

SUSE has an extensive script, SuSEfirewall2, which sets up an elaborate iptables configuration based on information in /etc/sysconfig/SuSEfirewall2. This latter file is typically edited by YaST2, after which the SuSEFirewall2 script is run.

***Instructor notes:***

**Purpose** — Discuss firewalling

**Details** — Coverage of iptables is not part of this course.

**Additional information** —

**Transition statement** — Last, let's look at something called NetworkManager.

# NetworkManager

- Client-side daemon for automatic configuration of TCP/IP
- Communicates with user via Network Manager Applet
- Ethernet:
  - Will start DHCP client if link detected
- Wifi:
  - Will detect all SSIDs in range
  - Will communicate with keymanager in the desktop environment for available SSIDs
  - Will configure WEP/WPA/WPA2 and other Wifi security protocols
  - Will start DHCP client once Wifi setup is complete
- Will configure client-side DNS
- Can manage VPNs
  - Cisco OpenConnect, OpenVPN, PPTP

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-15. NetworkManager

LX158.0

## Notes:

NetworkManager is a series of tools used to perform automatic configuration of the TCP/IP stack. The most important components are the NetworkManager daemon, which runs as root, and the NetworkManager Applet, which runs in the desktop environment of the user. There is also a command-line interface **nmcli**.

The aim of this suite of tools is to make network configuration on a desktop/laptop or otherwise "mobile" system as easy and painless as possible, while at the same time maintaining proper security. The latter is important since you might want to automatically connect to multiple WPA2-protected Wifi networks, without storing all their passwords in plain text on your laptop.

NetworkManager is highly customizable, but the usual setup is as follows:

- During the boot phase of your system, the NetworkManager daemon will start up and will run as root.
- During the login phase of a user, the desktop NetworkManager applet will be started.

- If a link is detected on a wired (Ethernet) network, NetworkManager will start a DHCP client on this network to try and obtain an IP address and other configuration items from a DHCP server.
- If a Wifi adapter is present, NetworkManager will perform a scan for all available Access Points. If an AP is found that is known to NetworkManager, NetworkManager will try to associate with it and then run a DHCP client to configure TCP/IP networking.

If the Wifi network is protected through WEP, WPA or WPA2, then the NetworkManager applet will communicate with the desktop keychain application to retrieve the proper key.

The desktop keychain manager will store any key it needs to store (WEP/WPA/WPA2, but also, for instance SSH keys) in an encrypted format. Through the use of PAM, this keystore is typically unlocked when the user logs in.

- If a Wifi adapter is present, but no known APs are found, then the user can click on the NetworkManager applet icon, view the list of available APs and select the one that the user wants to connect to. If necessary, the applet will ask for the appropriate password and use that to connect to the AP. The password will also be stored in the keychain.

NetworkManager will also configure routing and the client-side DNS setup, based on the information obtained from the DHCP server.

NetworkManager is also able to setup VPNs on behalf of the user. At this time, Cisco Openconnect, IPSec and PPTP are supported.

### **Red Hat Bug**

At the time of this writing, if you install Red Hat Enterprise Linux using the "Basic Server" profile, then all detected adapters are configured so that NetworkManager may handle them, and they are not activated automatically on boot (`NM_CONTROLLED="yes"`, `ONBOOT="no"`). However, in the "Basic Server" profile, NetworkManager is not installed by default. This leaves you without a network configuration after the first boot.

To activate the network adapter on boot, modify the file  
`/etc/sysconfig/network-scripts/ifcfg-eth0` to show:

```
NM_CONTROLLED="no"
```

```
ONBOOT="yes"
```

After a reboot, the system should now have a working network configuration. You can also activate an adapter manually using the `ifup <adapter>` script.

### **SUSE Configuration**

The SUSE installer allows you to choose between "traditional networking" (using the `ifup/ifdown` scripts) or NetworkManager as part of the installation. Traditional networking is most suitable for fixed-location systems (servers and wired desktops), while NetworkManager is most suitable for mobile systems (laptops) and desktops that require wireless connections.

**Instructor notes:**

**Purpose** — Discuss NetworkManager

**Details** —

**Additional information** —

**Transition statement** — Okay, that's it.

## Unit summary

---

Having completed this unit, you should be able to:

- Configure ethernet adapters
- Configure Wifi adapters
- Configure bridging, bonding and VLANs
- Configure IPv4 and IPv6
- Setup static and dynamic routing
- Setup DNS clients
- Discuss NetworkManager

© Copyright IBM Corporation 2002, 2013. All Rights Reserved.  
US Government Users Restricted Rights - Use duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp

Figure C-16. Unit summary

LX158.0

### Notes:

Having completed this unit, you should understand:

- How ethernet adapters are configured, including link parameters, bonding, bridging and VLANs
- How Wifi adapters are configured, including Wifi security
- How IPv4 and IPv6 addresses are configured
- How routing is configured
- How name resolution is configured
- How firewalling is performed
- The function of NetworkManager

**Instructor notes:**

**Purpose** — Summarize unit material.

**Details** —

**Additional information** —

**Transition statement** —





**IBM**  
®